ગુજરાત માધ્યમિક અને ઉચ્ચતર માધ્યમિક શિક્ષણ બોર્ડના પત્ર-ક્રમાંક
મઉમશબ/સંશોધન/2026/306-307, તા.21-1-2026થી મંજૂર

કમ્પ્યૂટર-અધ્યયન

ધોરણ 11

પ્રતિજ્ઞાપત્ર

ભારત મારો દેશ છે.
બધાં ભારતીયો મારાં ભાઈબહેન છે.
હું મારા દેશને ચાહું છું અને તેના સમૃદ્ધ અને
વૈવિધ્યપૂર્ણ વારસાનો મને ગર્વ છે.
હું સદાય તેને લાયક બનવા પ્રયત્ન કરીશ.
હું મારાં માતાપિતા, શિક્ષકો અને વડીલો પ્રત્યે
આદર રાખીશ અને દરેક જણ સાથે સભ્યતાથી વર્તીશ.
હું મારા દેશ અને દેશબાંધવોને મારી નિષ્ઠા અર્પું છું.
તેમનાં કલ્યાણ અને સમૃદ્ધિમાં જ મારું સુખ રહ્યું છે.

રાજ્ય સરકારની વિનામૂલ્યે યોજના હેઠળનું પુસ્તક

ગુજરાત રાજ્ય શાળા પાઠ્યપુસ્તક મંડળ
'વિદ્યાયન' સેક્ટર 10 - એ, ગાંધીનગર - 382 010
तमसो मा ज्योतिर्गमय

© ગુજરાત રાજ્ય શાળા પાઠ્યપુસ્તક મંડળ, ગાંધીનગર
આ પાઠ્યપુસ્તકના સર્વ હક ગુજરાત રાજ્ય શાળા પાઠ્યપુસ્તક મંડળને હસ્તક છે.
આ પાઠ્યપુસ્તકનો કોઈપણ ભાગ કોઈપણ રૂપમાં ગુજરાત રાજ્ય શાળા
પાઠ્યપુસ્તક મંડળના નિયામકશ્રીની લેખિત પરવાનગી વગર પ્રકાશિત કરી શકાશે નહિ.

લેખન
પ્રો. મનસુખ ટી. સાવલિયા (કન્વીનર)
ડૉ. હર્ષલ એ. આરોલકર
ડૉ. કુંતલ પી. પટેલ
ડૉ. બ્રિજેશ ભટ્ટ
શ્રી સાકેત એ. દવે

અનુવાદ
પ્રો. મનસુખ ટી. સાવલિયા
શ્રી સાકેત એ. દવે
ડૉ. રજનીકાંત એ. પંડ્યા

સમીક્ષા
શ્રી ઉરેશ એન. પરમાર
શ્રી રમણિકલાલ એલ. ગીલાતર
શ્રી રાજશ્રીબહેન એન. પડીયા
શ્રી પંકજકુમાર આર. શુક્લ
શ્રી પુનિત એસ. પુજારા
શ્રી રાજેશ આર. નાગપુરે
શ્રી સેજલબહેન પી. પરમાર

ભાષાશુદ્ધિ
શ્રી સંજય ત્રિવેદી

સંયોજન
શ્રી આશિષ એચ. બોરીસાગર
(એકેડેમિક સેક્રેટરી)

નિર્માણ-સંયોજન
ડૉ. કમલેશ એન. પરમાર
(નાયબ નિયામક : શૈક્ષણિક)

મુદ્રણ-આયોજન
ડૉ. કમલેશ એન. પરમાર
(નાયબ નિયામક : ઉત્પાદન)

વિતરણ-આયોજન
શ્રી હર્ષદ એચ. ચૌધરી
(નાયબ નિયામક : વહીવટ-વિતરણ)

પ્રસ્તાવના

માધ્યમિક અને ઉચ્ચત્તર માધ્યમિક શિક્ષણ બોર્ડ ઓપન સોર્સ ઓપરેટિંગ સિસ્ટમ અને તેને સુસંગત વિવિધ મુદ્દાઓ માટેના કમ્પ્યૂટર-અધ્યયનને લગતા ઓપન સોર્સ સાફ્ટવેર આધારિત નવો અભ્યાસક્રમ તૈયાર કર્યો છે. આ અભ્યાસ ગુજરાત સરકાર દ્વારા મંજૂર કરવામાં આવ્યો છે.

ગુજરાત સરકાર દ્વારા મંજૂર થયેલા **ધોરણ 11 ના કમ્પ્યૂટર-અધ્યયન** વિષયના નવા અભ્યાસક્રમ અનુસાર તૈયાર કરવામાં આવેલું આ પાઠ્યપુસ્તક વિદ્યાર્થીઓ સમક્ષ મૂકતાં પાઠ્યપુસ્તક મંડળ આનંદ અનુભવે છે.

આ વિષયનું અંગ્રેજી માધ્યમનું પાઠ્યપુસ્તક પ્રસિદ્ધ કરતાં પહેલાં એની હસ્તપ્રતની આ સ્તરે શિક્ષણકાર્ય કરતા શિક્ષકો અને તજજ્ઞો દ્વારા સર્વાંગી સમીક્ષા કરાવવામાં આવે છે. અને આ ઉપરાંત, આ હસ્તપ્રતની ચકાસણી ગુજરાત માધ્યમિક અને ઉચ્ચત્તર માધ્યમિક શિક્ષણ બોર્ડ દ્વારા પસંદ કરવામાં આવેલ નિષ્ણાંતોની પેનલ દ્વારા પણ કરવામાં આવી છે. તેમનાં સૂચનો અનુસાર હસ્તપ્રતમાં યોગ્ય સુધારા-વધારા કર્યા પછી આ પાઠ્યપુસ્તક પ્રસિદ્ધ કરવામાં આવ્યું છે. અંગ્રેજી માધ્યમના પાઠ્યપુસ્તકનું આ ગુજરાતી અનુવાદ છે.

પ્રસ્તુત પાઠ્યપુસ્તકને વિષયવસ્તુલક્ષી, રસપ્રદ અને ક્ષતિરહિત બનાવવા માટે મંડળે પૂરતી કાળજી લીધી છે, તેમ છતાં શિક્ષણમાં રસ ધરાવનાર વ્યક્તિઓ પાસેથી પુસ્તકની ગુણવત્તા વધારે તેવાં સૂચનો આવકાર્ય છે.

વિનયગિરિ ગોસાઈ	**મનુભાઈ બી. પાવરા**
નિયામક	કાર્યવાહક પ્રમુખ
તા. 29-01-2026	ગાંધીનગર

પ્રથમ આવૃત્તિ : 2026
પ્રકાશક : ગુજરાત રાજ્ય શાળા પાઠ્યપુસ્તક મંડળ, 'વિદ્યાયન', સેક્ટર 10-એ, ગાંધીનગર વતી
વિનયગિરિ ગોસાઈ, નિયામક
મુદ્રક :

મૂળભૂત ફરજો

ભારતના દરેક નાગરિકની ફરજો નીચે મુજબ રહેશે :*

(ક) સંવિધાનને વફાદાર રહેવાની અને તેના આદર્શો અને સંસ્થાઓનો, રાષ્ટ્રધ્વજનો અને રાષ્ટ્રગીતનો આદર કરવાની;

(ખ) આઝાદી માટેની આપણી રાષ્ટ્રીય લડતને પ્રેરણા આપનારા ઉમદા આદર્શોને હૃદયમાં પ્રતિષ્ઠિત કરવાની અને અનુસરવાની;

(ગ) ભારતનાં સાર્વભૌમત્વ, એકતા અને અખંડિતતાનું સમર્થન કરવાની અને તેમનું રક્ષણ કરવાની;

(ઘ) દેશનું રક્ષણ કરવાની અને રાષ્ટ્રીય સેવા બજાવવાની હાકલ થતાં, તેમ કરવાની;

(ચ) ધાર્મિક, ભાષાકીય, પ્રાદેશિક અથવા સાંપ્રદાયિક ભેદોથી પર રહીને, ભારતના તમામ લોકોમાં સુમેળ અને સમાન બંધુત્વની ભાવનાની વૃદ્ધિ કરવાની, સ્ત્રીઓના ગૌરવને અપમાનિત કરે તેવા વ્યવહારો ત્યજી દેવાની;

(છ) આપણી સમન્વિત સંસ્કૃતિના સમૃદ્ધ વારસાનું મૂલ્ય સમજી તે જાળવી રાખવાની;

(જ) જંગલો, તળાવો, નદીઓ અને વન્ય પશુપક્ષીઓ સહિત કુદરતી પર્યાવરણનું જતન કરવાની અને તેની સુધારણા કરવાની અને જીવો પ્રત્યે અનુકંપા રાખવાની;

(ઝ) વૈજ્ઞાનિક માનસ, માનવતાવાદ અને જિજ્ઞાસા તથા સુધારણાની ભાવના કેળવવાની;

(ટ) જાહેર મિલકતનું રક્ષણ કરવાની અને હિંસાનો ત્યાગ કરવાની;

(ઠ) રાષ્ટ્ર પુરુષાર્થ અને સિદ્ધિનાં વધુ ને વધુ ઉન્નત સોપાનો ભણી સતત પ્રગતિ કરતું રહે એ માટે, વૈયક્તિક અને સામૂહિક પ્રવૃત્તિનાં તમામ ક્ષેત્રે, શ્રેષ્ઠતા હાંસલ કરવાનો પ્રયત્ન કરવાની;

(ડ) માતા-પિતાએ અથવા વાલીએ 6 વર્ષથી 14 વર્ષ સુધીની વયના પોતાના બાળક અથવા પાલ્યને શિક્ષણની તકો પૂરી પાડવાની.

*ભારતનું સંવિધાન : કલમ 51-क

પાઠ્યપુસ્તક વિષે...

શિક્ષક મિત્રો,

ડિજિટલ સાક્ષરતાને મજબૂત કરવાની વધતી જતી જરૂરિયાત સાથે, ગુજરાત સરકારે વિદ્યાર્થીઓને આધુનિક તકનીકી પ્રગતિ માટે તૈયાર કરવા માટે જરૂરી પગલાં લીધાં છે, જેમાં ઓપન-સોર્સ ટેક્નોલોજીનો ઉપયોગ કરીને નવીનતમ વિકાસ સાથે કમ્પ્યૂટર શિક્ષણને પ્રોત્સાહન આપવાનો સમાવેશ થાય છે. આ ઓપન-સોર્સ સોલ્યૂશન શાળાઓને લાઇસન્સની ચિંતાઓ વિના સોફ્ટવેરને મુક્તપણે ઇન્સ્ટોલ કરવા, ઉપયોગ કરવા અને શેર કરવા સક્ષમ બનાવે છે, જે કમ્પ્યૂટર શિક્ષણને વધુ સુલભ અને ભવિષ્ય માટે ઉપયોગી બનાવે છે. કમ્પ્યૂટર ક્ષેત્રે - ખાસ કરીને આર્ટિફિશિયલ ઇન્ટેલિજન્સ (AI), સાયબર સુરક્ષા (cybersecueriy) અને ડેટા હેન્ડલિંગ જેવા ક્ષેત્રોમાં - ઝડપી પ્રગતિને ઓળખીને, આજની શીખવાની જરૂરિયાતોને પહોંચી વળવા માટે અભ્યાસક્રમને સુધારવો જરૂરી બન્યો. તેથી, ધોરણ 9 થી 12 માટે નવો આધુનિક અભ્યાસક્રમ તબક્કાવાર રીતે રજૂ કરવામાં આવ્યો છે, જે નીચેના વિષયો પર ધ્યાન કેન્દ્રિત કરે છેઃ ઓપન-સોર્સ ઓપરેટિંગ સિસ્ટમ્સ (Open-source operataing systems), જરૂરી ઉત્પાદકતા સાધનો (Essential productivity tools), ડેટાબેઝ મેનેજમેન્ટ, પ્રોગ્રામિંગ, AI નો પરિચય, સુરક્ષિત ઇન્ટરનેટ પ્રથાઓ (Safe internet practices), મૂળભૂત સાયબર સુરક્ષા (cybersecurity) જાગૃતિ વગેરે. આ સુધારાનો હેતુ વિદ્યાર્થીઓને ઉચ્ચ વર્ગો અને ઊભરતી તકનીકો માટે કમ્પ્યૂટર અભ્યાસમાં એક મજબૂત અને સુસંગત પાયો પૂરો પાડવાનો છે.

ધોરણ 11 માટેનું આ પાઠ્યપુસ્તક 'કમ્પ્યૂટર-અધ્યયન' વિષયની શ્રેણીમાં ત્રીજું પુસ્તક છે. આ પાઠ્યપુસ્તકનો મુખ્ય ઉદ્દેશ ડેટાબેઝનું સૈદ્ધાંતિક જ્ઞાન પૂરું પાડવાનો છે અને ત્યારબાદ આધુનિક ડેટાબેઝ સોફ્ટવેર લીબ્રેઓફિસ બેઝ (LibreOffice Base) નો પરિચય કરાવવાનો છે. આ સોફ્ટવેર દ્વારા માહિતીનું સંચાલન કરીને પછીથી ક્વેરી (Queries) નો ઉપયોગ કરીને ડેટામાંથી જરૂરી સમજ મેળવી શકાય છે. તે ડેટા યુઝર પાસેથી મેળવવા માટે ફોર્મ (Forms) અને વિવિધ હેતુઓ માટે ઉપયોગી રિપોર્ટ (Reports) તૈયાર કરવાનું પણ આવરી લે છે. આગળનો ભાગ ફ્લોચાર્ટ (flowchart) અને એલ્ગોરિધમ (algorithm) દ્વારા સમસ્યા ઉકેલવાની શરૂઆત કરે છે અને વિવિધ સમસ્યાઓ માટે પ્રોગ્રામ લખવા માટે C પ્રોગ્રામિંગને આવરી લે છે. અંતે, આધુનિક અભિગમને ધ્યાનમાં રાખી ઓબ્જેક્ટ-ઓરિએન્ટેડ ખ્યાલો (Object-Oriented Concepts) ને આવરી લેવામાં આવ્યા છે. આ પાઠ્યપુસ્તક ડેટાનું સંચાલન કરવા અને પ્રોગ્રામિંગ કૌશલ્યો વિકસાવવા માટે મજબૂત પાયો નાખે છે, જે આધુનિક સોફ્ટવેર ડેવલપમેન્ટના ક્ષેત્રમાં વધુ પ્રગતિ કરવા માટે ઉપયોગી છે.

અમે આશા રાખીએ છીએ કે આ સામગ્રી વિદ્યાર્થીઓ માટે ઉપયોગી થશે અને તમને ઉબુન્ટુ ઓપરેટિંગ સિસ્ટમ અને જરૂરી સોફ્ટવેર સાથે સિસ્ટમ પર શિક્ષણ આપવાની અને પ્રેક્ટિકલ કરાવવાની મજા આવશે.

વ્હાલા વિદ્યાર્થીઓ

કમ્પ્યૂટર, ઓપરેટિંગ સિસ્ટમ, ઓફિસ ટૂલ્સ, ઇન્ટરનેટ અને વેબ ડિઝાઇનિંગના પાયાના ખ્યાલો તમારા અગાઉના વર્ગોમાં શીખ્યા પછી, હવે તમે કમ્પ્યૂટર સાયન્સના વધુ ઊંડા અને રોમાંચક ક્ષેત્રો વિષે જાણવા માટે તૈયાર છો. ધોરણ 11નું આ પુસ્તક તમને કમ્પ્યૂટર ડેટાનું વ્યવસ્થાપન કેવી રીતે કરે છે અને વાસ્તવિક દુનિયાની સમસ્યાઓ હલ કરવા માટે સોફ્ટવેર કેવી રીતે વિકસાવવામાં આવે છે તેના મૂળમાં લઈ જાય છે. તમે ડેટાબેઝ મેનેજમેન્ટ

સિસ્ટમ (DBMS) થી શરૂઆત કરશો, જે મોટી માત્રામાં ડેટાને અસરકારક રીતે સંગ્રહિત કરવા, વ્યવસ્થિત કરવા અને સંચાલિત કરવામાં મદદ કરે છે. આ પુસ્તક સમજાવે છે કે શાળાઓ, બેંકો અને ઉદ્યોગોમાં માહિતીને સુરક્ષિત રીતે અને ઝડપથી સંચાલિત કરવા માટે ડેટાબેઝનો ઉપયોગ કેવી રીતે થાય છે. તમે તમારા પોતાના ડેટાબેઝની રચના અને સંચાલન કરવા માટે એક ઓપન-સોર્સ ડેટાબેઝ ટૂલ, લીબ્રેઓફિસ બેઝ સાથે કામ કરવાનું શીખશો. ક્વેરીનો ઉપયોગ કરીને ડેટા પુનઃપ્રાપ્ત કરવો અને ફોર્મ તથા રિપોર્ટ બનાવવાં જેવા વિષયો તમને ડેટાને સંભાળવાનું વ્યવહારિક જ્ઞાન આપશે - જે આજે દરેક ક્ષેત્રમાં ઉપયોગી કૌશલ્ય છે.

પુસ્તકનો બીજો ભાગ સમસ્યાનું નિરાકરણ અને પ્રોગ્રામિંગ પર કેન્દ્રિત છે, જે કમ્પ્યૂટર સાયન્સના મૂળમાં છે. તમે ફ્લોચાર્ટ અને અલ્ગોરિધમ બનાવીને સમસ્યાઓનો તાર્કિક રીતે સામનો કરવાનું શીખશો, જે તમને પગલું-દર-પગલાં ઉકેલોનું આયોજન અને વિઝ્યુલાઇઝ (દૃશ્યમાન) કરવામાં મદદ કરશે. ત્યારબાદ, તમે C પ્રોગ્રામિંગ ભાષા સાથે તમારી સફર શરૂ કરશો, જે સૌથી જૂની છતાં સૌથી શક્તિશાળી પ્રોગ્રામિંગ ભાષાઓમાંની એક છે. તમે ડેટા પ્રકારો, ઓપરેટર અને પદાવાલીઓનો ઉપયોગ સમજશો, અને પ્રોગ્રામને ઇન્ટરેક્ટિવ (સંવાદાત્મક) બનાવતી ઈનપુટ અને આઉટપુટ ક્રિયાઓ કરવાનું શીખશો. તમે નિર્ણય લેવાની (decision-making) અને નિયંત્રણ રચનાઓ (control structures) પણ શીખશો, જે કમ્પ્યૂટરને શરતોના આધારે પસંદગીઓ કરવા અને અલગ-અલગ કાર્યો કરવા મદદ કરે છે. આ કૌશલ્યો એડવાન્સ પ્રોગ્રામિંગ માટેનો આધાર બનાવે છે અને તમને ભવિષ્યના સોફ્ટવેર ડેવલપમેન્ટ માટે તૈયાર કરે છે.

અંતિમ ભાગ તમને ઑબ્જેક્ટ-ઓરિએન્ટેડ પ્રોગ્રામિંગ (OOP) ના ખ્યાલોનો પરિચય કરાવે છે, જે આધુનિક સોફ્ટવેરની રચના કેવી રીતે થાય છે તે સમજવા માટે આવશ્યક છે. તમે ક્લાસ (Class), ઓબ્જેક્ટ (Object), ઈન્હેરિટન્સ (Inheritance), એન્કેપ્સ્યુલેશન (Encapsulation) અને પોલિમોર્ફિઝમ (Polymorphism) જેવા મુખ્ય વિચારો શીખશો, જે પ્રોગ્રામરને કોડને અસરકારક રીતે ગોઠવવા અને પુનઃઉપયોગ કરવાની મંજૂરી આપે છે. આ વિષયો તમારા પ્રોગ્રામિંગ જ્ઞાનને વાસ્તવિક દુનિયાની એપ્લિકેશન સાથે જોડે છે, જેનાથી ગેમ્સ, એપ્લીકેશન અને વેબસાઇટ જેવી મોટી સિસ્ટમ્સ કેવી રીતે બનાવવામાં આવે છે તે સમજવું સરળ બને છે. આ પુસ્તકના અંત સુધીમાં, તમે ડેટા હેન્ડલિંગ અને પ્રોગ્રામિંગમાં મજબૂત કૌશલ્યો ધરાવતા હશો. તમે તાર્કિક રીતે વિચારવા, સમસ્યાઓનું અસરકારક રીતે નિરાકરણ કરવા અને કમ્પ્યૂટર સાયન્સ, આર્ટિફિશિયલ ઈન્ટેલિજન્સ અને સોફ્ટવેર એન્જિનિયરિંગના એડવાન્સ અભ્યાસ માટે એક નક્કર પાયો બનાવવામાં સક્ષમ બનશો.

અનુક્રમણિકા

1.	ડેટાબેઝ મેનેજમેન્ટ સિસ્ટમનો પરિચય	1
2.	લિબ્રેઑફિસ બેઝનો પરિચય	17
3.	ક્વેરીનો ઉપયોગ કરીને ડેટાની પુનઃપ્રાપ્તિ	38
4.	ફોર્મ અને રિપોર્ટ સાથે કાર્ય	54
5.	સમસ્યા ઉકેલ : ફ્લોચાર્ટ અને અલ્ગોરિધમ	75
6.	C પ્રોગ્રામિંગનો પરિચય	92
7.	C ભાષામાં ડેટા પ્રકારો	102
8.	ઓપરેટર અને પદાવલીઓ	113
9	ઈનપુટ આઉટપુટ ઓપરેશન	129
10.	નિર્ણય લેવાની નિયંત્રણ સંરચનાઓ	146
11.	ઑબ્જેક્ટ ઓરિએન્ટેડ અભિગમ અને પ્રોગ્રામિંગ	165

ડેટાબેઝ મેનેજમેન્ટ સિસ્ટમનો પરિચય

પરિચય

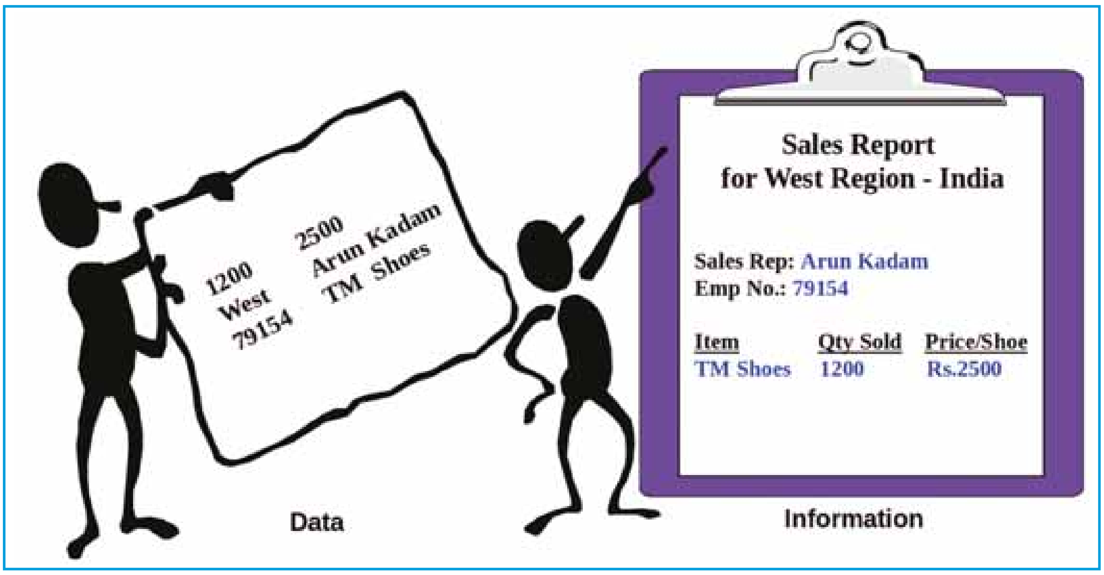
આજના ડિજિટલ યુગમાં, સંસ્થાઓ દર સેકન્ડે મોટા પ્રમાણમાં ડેટા ઉત્પન્ન કરે છે. આ ડેટાને યોગ્ય રીતે સંગ્રહ, પ્રોસેસ અને સુવ્યવસ્થિત (Organize) કરવાની જરૂર પડે છે જેથી તેનો ઉપયોગ ભવિષ્યમાં નિર્ણયો લેવા માટે થઈ શકે. આથી, અસરકારક ડેટા મેનેજમેન્ટ માટે ડેટા અને ઇન્ફોર્મેશન વચ્ચેનો તફાવત સમજવો મહત્ત્વપૂર્ણ બને છે. ડેટાબેઝ એક સંરચિત ભંડાર (Structured Repository) તરીકે કાર્ય કરે છે જે ડેટાને કાર્યક્ષમ રીતે મેળવવા અને ડેટાને મેનીપ્યુલેટ કરવાની સુવિધા આપે છે. આ પ્રકરણમાં, આપણે ડેટા અને ઇન્ફોર્મેશન વચ્ચેનો તફાવત સમજશું. આપણે ડેટાબેઝ મેનેજમેન્ટ સિસ્ટમ (DBMS) અને તેના ઘટકો વિશે જાણીશુ. આપણે એક નમૂનાનો ડેટાબેઝ પણ બનાવીશું અને ડેટાબેઝમાં જે મહત્ત્વનો ભાગ ભજવે છે તે ડેટા ટાઇપ વિશે અભ્યાસ કરીશુ.

ડેટા વિરુદ્ધ (vs) ઇન્ફોર્મેશન

ઘણીવાર યુઝર ડેટા અને ઇન્ફોર્મેશન શબ્દોનો સમાનાર્થી તરીકે ઉપયોગ કરે છે. તેથી, આ બંને શબ્દોનો ચોક્કસ અર્થ જાણવો જરૂરી છે. ડેટા અને ઇન્ફોર્મેશનને અનેક રીતે વ્યાખ્યાયિત કરી શકાય છે, તો ચાલો આપણે તે શું છે તે સમજવાથી શરૂઆત કરીએ.

ડેટા એ લોકો, સ્થળો, વસ્તુઓ અથવા ઘટનાઓ સંબંધિત અસંગઠિત હકીકતો, આંકડાઓ અને વિગતો છે. તે મોટાભાગે કાચો (raw) ગણવામાં આવે છે. ડેટા કોઈપણ ફોર્મેટમાં હોઈ શકે છે, જેમ કે સંખ્યાઓ, ટેક્સ્ટ, ઈમેજ અથવા વિડિયો. તે લખેલો કે બોલેલો પણ હોઈ શકે છે. કાચા સ્વરૂપમાં ડેટા કદાચ બહુ ઉપયોગી હોતો નથી. નિર્ણય લેવાની પ્રક્રિયામાં ડેટાના મહત્ત્વને જોતાં, ઘણી કંપનીઓ તેને એક મુખ્ય સંપત્તિ માને છે. ડેટાનો મૂળભૂત ગુણધર્મ એ છે કે તે ઘણીવાર અપ્રસ્તુત અને અસંરચિત (unstructured) હોય છે.

ઇન્ફોર્મેશન એ પ્રોસેસ થયેલો ડેટા છે. જ્યારે કાચા ડેટાને કોઈ પ્રકારની પરિવર્તન પ્રક્રિયામાંથી પસાર કરવામાં આવે છે ત્યારે તે ઇન્ફોર્મેશનમાં રૂપાંતરિત થાય છે. ઇન્ફોર્મેશનનો મૂળભૂત ગુણધર્મ એ છે કે તે સંરચિત (structured), સુસંગત અને અર્થપૂર્ણ હોય છે.



આકૃતિ 1.1 : ડેટા અને ઇન્ફોર્મેશન વચ્ચેનો તફાવત

ચાલો આપણે એક ઉદાહરણ લઈને ડેટા અને ઇન્ફોર્મેશન વચ્ચેનો તફાવત સમજવાનો પ્રયાસ કરીએ. 1200, 2500, 79154, ''પશ્ચિમ'', ''અરુણ કદમ'' અને ''TM શૂઝ'' જેવી સંખ્યાઓ અને ટેક્સ્ટ માત્ર ડેટા છે. સ્વતંત્ર રીતે, તેમાંથી કોઈ પણ સુસંગત લાગતું નથી. જોકે, તમે તેને યોગ્ય સંદર્ભમાં મૂકોઃ ''શ્રી અરુણ કદમ, કર્મચારી નંબરઃ 79154, એ પશ્ચિમ ભારતના ક્ષેત્રમાં 2500 રૂપિયાના ભાવે 1200 TM શૂઝનું વેચાણ કર્યું છે'', તો તે ઇન્ફોર્મેશન બની જાય છે, કારણ કે તે અર્થ અને સંદર્ભ પ્રદાન કરે છે. આકૃતિ 1.1 ડેટા અને ઇન્ફોર્મેશન વચ્ચેનો તફાવત દર્શાવે છે.

ડેટાબેઝ (Database) શું છે?

આજે આપણે બધા આપણા રોજિંદા જીવનમાં એક અથવા વધુ પ્રકારના ડેટાબેઝનો ઉપયોગ કરીએ છીએ. ડેટાબેઝનું સૌથી સરળ ઉદાહરણ આપણા મોબાઇલ ફોનમાં રહેલી કોન્ટેક્ટ બુક છે, જેમાં આપણા પરિચિતના નામ, કોન્ટેક્ટ નંબર અને ઈમેલ એડ્રેસ હોય છે. કોઈ વ્યક્તિગત પરિચિતને લગતી વિગતોને રેકોર્ડ કહેવામાં આવે છે.

ડેટાબેઝ એ યોગ્ય રીતે વ્યવસ્થિત સ્ટોર કરેલ સંબંધિત ડેટા આઇટમનો સમૂહ છે. આ વ્યાખ્યા મુજબ, જ્યારે આપણે ડેટાબેઝનો ઉલ્લેખ કરીએ ત્યારે બે મુખ્ય મુદ્દાઓ ધ્યાનમાં લેવા જરૂરી છે. પ્રથમ; તેમાં એકબીજા સાથે સંબંધિત ડેટા આઇટમ હોવી જોઈએ અને બીજું, તે એક વ્યવસ્થિત કલેકશન હોવું જોઈએ.

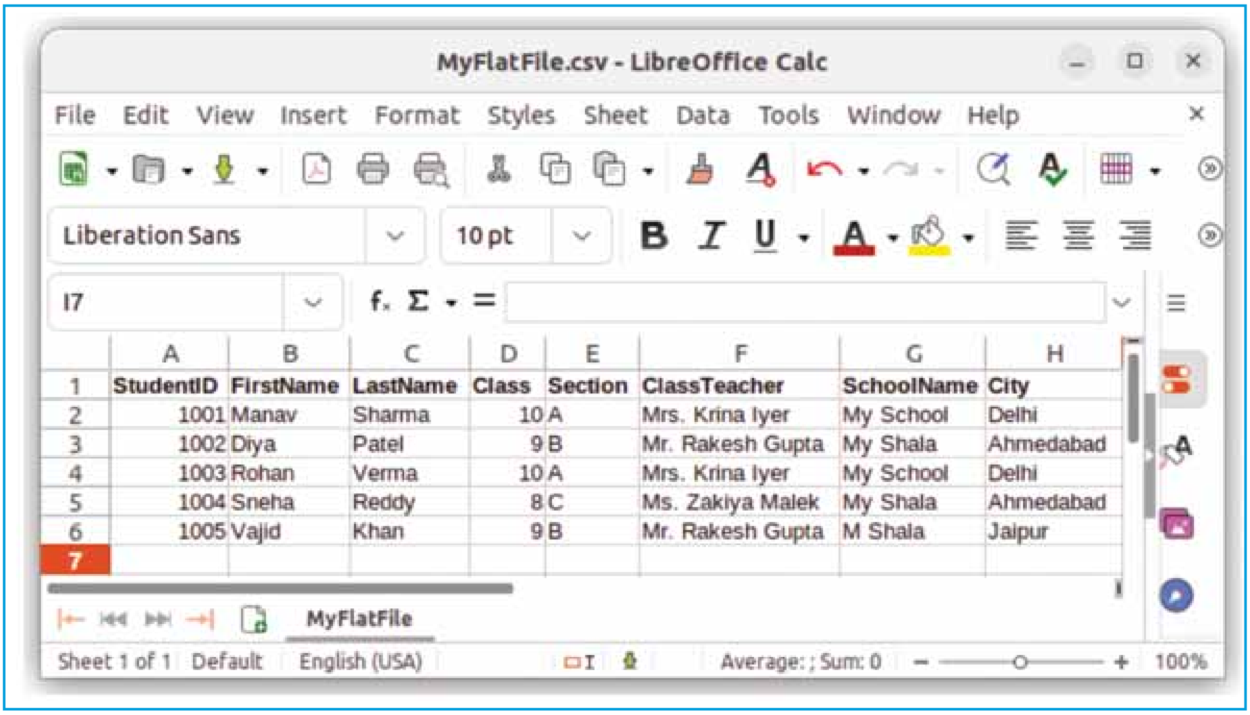
ઘટકોની તાર્કિક ગોઠવણ જ્યારે જરૂર પડ્યે ત્યારે ઘટકો શોધવાનું હંમેશા સરળ બનાવે છે. ડેટાબેઝ ઘણીવાર પ્રિ-ડિફાઇન્ડ નિયમોના સમૂહ અને ડેટા મોડેલ અનુસાર ડિઝાઇન કરવામાં આવે છે. ડેટા મોડેલ એ ડેટાને સ્ટોર કરવા અને ફરીથી મેળવવાની રીતનું વર્ણન કરે છે. DBMSમાં વિવિધ પ્રકારના ડેટા મોડેલ છે જેમ કે હાયરાર્કિકલ, નેટવર્ક, રિલેશનલ, ઓબ્જેક્ટ-ઓરિએન્ટેડ, ફ્લેટ, સેમી-સ્ટ્રક્ચર્ડ, એસોસિયેટિવ અને કન્ટેક્સ્ટ. આમાંના બે મોડેલ, ફ્લેટ અને રિલેશનલ, મહત્ત્વનો ભાગ ભજવે છે. ફ્લેટ મોડેલ ડેટા સ્ટોર કરવાનું પ્રારંભિક મોડેલ હતું, જ્યારે રિલેશનલ મોડેલ આજે સૌથી વધુ વ્યાપકપણે ઉપયોગમાં લેવાતું મોડેલ છે.

ફ્લેટ ડેટા મોડેલ (Flat Data Model)

ફ્લેટ ડેટા મોડેલમાં, ડેટા એક જ, દ્વિ-પરિમાણીય (two-dimensional) ટેબલ તરીકે રજૂ થાય છે જેમાં કોઈ વ્યાખ્યાયિત જોડાણો હોતા નથી. બધા રેકોર્ડમાં ફીલ્ડની સંખ્યા અને પ્રકાર સુસંગત હોય છે. તેની સરળતા અને સમજવામાં સરળતાને કારણે, આ ફોર્મેટ નાના અથવા એક વખતના ડેટાસેટ માટે આદર્શ છે. આ ફોર્મેટનો ઉપયોગ વારંવાર કોમા સેપરેટેડ વેલ્યૂઝ (CSV), લિબ્રેઓફિસ કેલ્સી (LibreOffice Calc) અને એમએસ એક્સેલ (MS Excel) ફાઇલોમાં થાય છે. એક CSV ફાઇલ અને લિબ્રેઓફિસ કેલ્સીનો સેમ્પલ ડેટાસેટ આકૃતિ 1.2 અને 1.3માં દર્શાવેલ છે.

```
MyFlatFile.csv
1 'StudentID','FirstName','LastName','Class','Section','ClassTeacher','SchoolName','City'
2 1001,'Manav','Sharma',10,'A','Mrs. Krina Iyer','My School','Delhi'
3 1002,'Diya','Patel',9,'B','Mr. Rakesh Gupta','My Shala','Ahmedabad'
4 1003,'Rohan','Verma',10,'A','Mrs. Krina Iyer','My School','Delhi'
5 1004,'Sneha','Reddy',8,'C','Ms. Zakiya Malek','My Shala','Ahmedabad'
6 1005,'Vajid','Khan',9,'B','Mr. Rakesh Gupta','M Shala','Jaipur'
```

આકૃતિ 1.2 : ફ્લેટ ડેટા મોડેલનું ઉદાહરણ (CSV ફાઇલ)



	A	B	C	D	E	F	G	H
1	StudentID	FirstName	LastName	Class	Section	ClassTeacher	SchoolName	City
2	1001	Manav	Sharma	10	A	Mrs. Krina Iyer	My School	Delhi
3	1002	Diya	Patel	9	B	Mr. Rakesh Gupta	My Shala	Ahmedabad
4	1003	Rohan	Verma	10	A	Mrs. Krina Iyer	My School	Delhi
5	1004	Sneha	Reddy	8	C	Ms. Zakiya Malek	My Shala	Ahmedabad
6	1005	Vajid	Khan	9	B	Mr. Rakesh Gupta	M Shala	Jaipur
7								

આકૃતિ 1.3 : ફ્લેટ ડેટા મોડેલનું ઉદાહરણ

ફ્લેટ ડેટા મોડેલ ખૂબ જ સરળ અને સમજવામાં સહેલુ છે. જોકે, કેટલીક મર્યાદાઓને કારણે તેનો ઉપયોગ જટિલ અથવા મોટા પાયાના ડેટાસેટ માટે થઈ શકતો નથી. ફ્લેટ ડેટા મોડેલની કેટલીક મર્યાદાઓ નીચે મુજબ છે:

ડેટા રિડન્ડન્સી (Data Redundancy) : ફ્લેટ ડેટાબેઝમાં એક જ ડેટા ઘણી વખત રિપીટ થવાની શક્યતાઓ ઘણી વધારે છે. આકૃતિ 1.3માં ક્લાસ ટીચરનું નામ ઘણી વખત દેખાય છે.

અપડેટ એનોમલીઝ (Update Anomalies) : ડેટા ઘણી વખત રિપીટ થતો હોવાથી, જો ઉદાહરણ તરીકે 'My School' નામ અપડેટ કરવાનું હોય તો આપણે આ ડેટા ધરાવતી દરેક રો (હરોળ)માં ફેરફાર કરવો પડશે. જો રોની સંખ્યા ખૂબ વધારે હોય તો ભૂલો થવાનું જોખમ વધી જાય છે.

મર્યાદિત મોડ્યુલારિટી (Limited Modularity) : આકૃતિ 1.3માં જોઈ શકાય છે કે બધો ડેટા એક જ ટેબલમાં સ્ટોર કરેલ છે. આ કારણે ફેરફારોને અલગ કરવા અથવા ડેટા ઘટકોનો જુદા જુદા સંદર્ભમાં ફરીથી ઉપયોગ કરવો ખૂબ મુશ્કેલ બને છે.

નબળી સ્કેલેબિલિટી (Poor Scalability) : 100 રેકોર્ડ્સ જેવી નાની માત્રાના ડેટા માટે, ફ્લેટ ડેટા મોડેલ બરાબર કામ કરે છે. જેમ જેમ ડેટા વધે છે, તેમ તેનું મેનેજમેન્ટ, અપડેટિંગ અને ક્વેરીંગ બિનકાર્યક્ષમ બને છે. સ્ટ્રક્ચરના અભાવને કારણે પર્ફોર્મન્સ સંબંધિત સમસ્યાઓ પણ ઊભી થાય છે.

ડેટા રિલેશનશિપનો અભાવ (No Data Relationships) : ફ્લેટ ડેટા મોડેલમાં વાસ્તવિક દુનિયાના સ્ટ્રક્ચર્સનું પ્રતિનિધિત્વ કરવું મુશ્કેલ છે, કારણ કે તે વિવિધ એન્ટિટીઝ વચ્ચે રિલેશનશિપ ડિફાઈન કરી શકતું નથી. ઉદાહરણ તરીકે, આકૃતિ 1.3 માં વિદ્યાર્થીઓ-શિક્ષકો અથવા ક્લાસ-વિષયો વચ્ચેનો રિલેશન ઓળખવો ખૂબ મુશ્કેલ છે.

ડેટા ઈન્ટિગ્રિટીની સમસ્યાઓ (Data Integrtiy Issues) : ઓપરેશન દરમિયાન સુસંગતતા જાળવવી મુશ્કેલ બની શકે છે, જેનાથી ડેટા ઈન્ટિગ્રિટીની સમસ્યાઓ ઊભી થાય છે. ઉદાહરણ તરીકે, શાળાનું નામ અપડેટ કરતી વખતે એક ટાઈપોગ્રાફિકલ ભૂલ ધ્યાન બહાર રહી શકે છે.

જટિલ ડેટાને ક્વેરી કરવું મુશ્કેલ (Querying Complex Data is Difficult) : સંસ્થામાં કેટલાક નિર્ણયો લેવા માટે ડેટાબેઝની જરૂર હોય છે. ફ્લેટ ડેટા મોડેલમાં ડેટા વચ્ચે સ્ટ્રક્ચર્ડ લિંક્સના અભાવને કારણે વિશ્લેષણ કરવું અથવા અર્થપૂર્ણ રિલેશનશિપ શોધવી ખૂબ મુશ્કેલ બની જાય છે.

રિલેશનલ ડેટા મોડેલ (Relational Data Model)

રિલેશનલ ડેટા મોડેલ ફ્લેટ ડેટા મોડેલમાં આવતી તમામ સમસ્યાઓનો ઉકેલ લાવે છે. રિલેશનલ ડેટા મોડેલમાં ડેટાને ટેબલના સ્વરૂપમાં ગોઠવવામાં આવે છે. ટેબલ રો અને કોલમથી બનેલા હોય છે. એક ટેબલનો ઉપયોગ ચોક્કસ એન્ટિટીનું પ્રતિનિધિત્વ કરવા માટે થાય છે, અને વિવિધ ટેબલ વચ્ચેના સંબંધોને કી (ટેબલના ફીલ્ડ) નો ઉપયોગ કરીને નિયંત્રિત કરવામાં આવે છે. આથી, આકૃતિ 1.3માં દર્શાવેલ ડેટાબેઝને વિદ્યાર્થી, શિક્ષક, શાળા અને અન્ય એન્ટિટીનું પ્રતિનિધિત્વ કરતા ટેબલમાં વિભાજિત કરી શકાય છે.

રિલેશનલ ડેટા મોડેલ જટિલ ડેટાબેઝને ક્વેરી (પ્રશ્ન) કરવા માટે સ્ટ્રક્ચર્ડ ક્વેરી લેંગ્વેજ (SQL) (જેની વાક્યરચના અંગ્રેજી જેવી છે) નો ઉપયોગ કરે છે. તેની સરળતા, અનુકૂલન ક્ષમતા અને મજબૂત સૈદ્ધાંતિક આધારને કારણે, આ મોડેલનો વ્યાપકપણે ઉપયોગ થાય છે. આપણે આ પ્રકરણના આગામી વિભાગમાં ટેબલ કેવી રીતે બનાવવું તે શીખીશું.

ડેટાબેઝની જરૂરિયાત

આજના સમયમાં વ્યવસાય ડેટા પર આધારિત છે. ડેટા વિના તેમના માટે નવા નવા ઉકેલ અને પ્રોડક્ટ્સ બનાવવાનું અશક્ય છે. તો, ચાલો હવે આપણે જોઈએ કે ડેટાબેઝને આટલો મહત્ત્વપૂર્ણ કોણ બનાવે છે અને તેની શું જરૂરિયાત છે. સંસ્થાની નીચે મુજબની જરૂરિયાતો ડેટાબેઝની આવશ્યકતા પર પ્રકાશ પાડે છે.

ડેટાની ગોઠવણી અને માળખું (Data Organization and Structure)

પહેલા જણાવ્યા મુજબ, ડેટાબેઝ સંસ્થા દ્વારા જનરેટ થતા મોટા પ્રમાણમાં ડેટાને સ્ટ્રક્ચર્ડ ફોર્મેટમાં ગોઠવવા માટે એક વ્યવસ્થિત માર્ગ અથવા અભિગમ પ્રદાન કરે છે. તે સંબંધિત ઇન્ફોર્મેશનના તાર્કિક જૂથ બનાવે છે, જેનાથી કાર્યક્ષમ ડેટા સંગ્રહ થાય છે.

વિદ્યાર્થી, શિક્ષક અને શાળા જેવા અનેક ટેબલ બનાવવાથી કોઈપણ શાળા સંચાલન પ્રવૃત્તિઓ માટે જરૂરી ડેટાને ગોઠવવામાં મદદ મળે છે.

કાર્યક્ષમ ડેટા રીટ્રાઈવલ (Efficient Data Retrieval)

વ્યવસાયોએ તેમની પાસેના ડેટાના આધારે ઝડપી નિર્ણયો લેવાની જરૂર હોય છે. ડેટાને તાર્કિક જૂથમાં સંગ્રહિત કરવામાં આવતો હોવાથી, તે જરૂરી ઇન્ફોર્મેશનને ઝડપથી શોધવા અને રીટ્રાઈવ (પરત મેળવવા) કરવા સક્ષમ બનાવે છે.

ધારો કે આપણે કોઈ વિદ્યાર્થી વિશેની વિગતો શોધવી છે, તો તે વિદ્યાર્થીના ટેબલમાં સ્ટોર કરેલ ડેટામાંથી સરળતાથી મેળવી શકાય છે. જો વધારાની માહિતીની જરૂર હોય, ઉદાહરણ તરીકે વિદ્યાર્થીઓના માર્ક્સ, તો તે માર્ક્સ સ્ટોર કરેલા બીજા સંબંધિત ટેબલના ડેટાનો ઉપયોગ કરીને મેળવી શકાય છે.

ડેટા ઈન્ટેગ્રિટી અને કન્સિસટન્સી (Data Integrity and Consistency)

કોઈપણ વ્યવસાય ડેટા ખોટો હોય તે પસંદ કરશે નહીં. ડેટા પર વેલિડેશન નિયમો લાગુ કરી શકે તેવો ડેટાબેઝ બનાવવાથી વ્યવસાયોને તેમની તમામ કામગીરીમાં સચોટ અને સુસંગત ડેટા જાળવી રાખવામાં મદદ મળે છે.

ડેટાને તાર્કિક અને સ્ટ્રક્ચર્ડ રીતે ગોઠવવામાં આવે છે, જેથી કોઈપણ જરૂરી અપડેટ ફક્ત એક જ જગ્યાએ થશે. આ ફેરફારની અસર પછી સમગ્ર ડેટાસેટમાં પ્રતિબિંબિત થશે. ઉદાહરણ તરીકે, જો આપણે શાળાનું નામ બદલીએ, તો આપણે તેને ફક્ત શાળાના ટેબલમાં જ બદલીશું. અપડેટ થયેલ મૂલ્ય પછી બધા ટેબલમાં પ્રતિબિંબિત થશે.

એકસાથે એક્સેસ (Concurrent Access)

ઘણીવાર ડેટાના એક જ ભાગને અનેક હિસ્સેદારો દ્વારા એક્સેસ (ઉપયોગ) કરવાની જરૂર પડે છે. ડેટાબેઝ ઘણા યુઝર્સને એકસાથે ડેટાને એક્સેસ અને ડેટામાં ફેરફાર કરવાની સવલત પૂરી પાડે છે. આકૃતિ 1.3 ના કિસ્સામાં, ક્લાસ ટીચરના નામને વિદ્યાર્થી તેમજ શાળાના પ્રિન્સિપાલ દ્વારા એક્સેસ કરી શકાય છે.

એક્સેસ નિયંત્રણ (Access Control)

ઘણીવાર ડેટાને વારંવાર એકબીજા સાથે વહેંચવાની કરવાની જરૂર હોય છે, પરંતુ તે સુરક્ષિત અને પ્રતિબંધિત રીતે થવું જોઈએ. ડેટાબેઝ સંવેદનશીલ ઈન્ફોર્મેશનને સુરક્ષિત રાખવા માટે રોલ-આધારિત પરમિશન અને યુઝર-લેવલ સિક્યુરિટી પ્રદાન કરે છે. આકૃતિ 1.3માં જો આપણે દરેક રોમાં કુલ માર્ક્સનું ફીલ્ડ ઉમેરીએ તો, આપણને વિવિધ પ્રકારના એક્સેસ કંટ્રોલની જરૂર પડશે.

ઉદાહરણ તરીકે, એક શિક્ષક ચોક્કસ ક્લાસના બધા વિદ્યાર્થીઓના માર્ક્સ જાણવા માંગશે, તેથી તેને તે ચોક્કસ ક્લાસના તમામ માર્ક્સનો એક્સેસ આપવો જરૂરી છે. બીજી બાજુ, એક વિદ્યાર્થી ફક્ત તેના અથવા તેણીના માર્ક્સ જ જોઈ શકવો જોઈએ.

ડેટા બેકઅપ અને રિકવરી (Data Backup and Recovery)

કોઈપણ વ્યવસાય માટે વૈકલ્પિક વ્યવસ્થા હોવી અત્યંત આવશ્યક છે. ડેટાબેઝ હોવાથી, ડેટાના નુકસાનને રોકવા માટે ઓટોમેટેડ બેકઅપ માટેની સગવડ ઊભી કરી શકાય છે. આમ, બેકઅપ સમયસર રિકવરી ક્ષમતાને સક્ષમ બનાવે છે.

ધારો કે આપણે છેલ્લા દસ વર્ષના વિદ્યાર્થીઓના પરિણામની હાર્ડકોપી સ્ટોર કરી હતી, પરંતુ કોઈ કારણોસર તે રેકોર્ડ્સ નાશ પામ્યા. આ ડેટા પાછો મેળવવો લગભગ અશક્ય હશે, પરંતુ જો તે જ રેકોર્ડ્સ ડેટાબેઝમાં ઉપલબ્ધ હોય, તો માર્કશીટ ફરીથી બનાવવી ખૂબ જ સરળ હશે.

અહેવાલ અને વિશ્લેષણ (Reporting and Analytics)

ડેટાબેઝ હોવાથી, કોઈ વ્યવસાય નિર્ણયો લેવા માટે અહેવાલ અને ડેશબોર્ડ બનાવી શકે છે. તે જટિલ ડેટા વિશ્લેષણ અને બિઝનેસ ઇન્ટેલિજન્સ કાર્યોને સુવિધાજનક બનાવે છે, જેનો ઉપયોગ વ્યવસાયની ક્ષમતાઓને વધારવા માટે થઈ શકે છે.

જો ડેટા ડેટાબેઝનો ભાગ હોય, તો સેંકડો વિદ્યાર્થીઓની માર્કશીટ બનાવવી અને પ્રિન્ટ કરવી ખૂબ જ સરળ હશે. ઉપરાંત, વિષયમાં વિદ્યાર્થીઓની નિષ્ફળતા, શ્રેષ્ઠ પ્રદર્શન કરનાર શિક્ષક અને અન્ય જેવા વિશ્લેષણો કરવાનું કાર્ય પણ ખૂબ સરળ બની જશે.

ડેટાબેઝની શા માટે જરૂર છે તેના ઘણા વધુ કારણો વિશે વિચારી શકાય છે, પરંતુ અત્યારે ઉપરોક્ત કારણો પૂરતા છે.

ડેટાબેઝ મેનેજમેન્ટ સિસ્ટમ શું છે? (What is Database Management System?)

સરળ શબ્દોમાં કહીએ તો, ડેટાબેઝ એ એક સંગ્રહસ્થાન છે. આ સંગ્રહસ્થાન જાતે અથવા કમ્પ્યૂટર દ્વારા બનાવી શકાય છે. ડેટાબેઝ મેનેજમેન્ટ સિસ્ટમ (DBMS) એક એવું સૉફ્ટવેર છે જે યુઝર અને ડેટાબેઝ વચ્ચે સેતુ તરીકે કામ કરે છે. તે કમ્પ્યૂટરનો ઉપયોગ કરીને ડેટાને અસરકારક રીતે સંગ્રહ કરવા, મેનેજ કરવા અને તેમાં ફેરફાર કરવાની સુવિધા પૂરી પાડે છે.

DBMS એ ડેટાને વ્યવસ્થિત રીતે ગોઠવવા માટે એક કેન્દ્રીય પ્રણાલી તરીકે કાર્ય કરે છે. તે ડેટાની ઈન્ટેગ્રિટી અને કન્સિસટન્સી જાળવવા માટે ટેબલ, રિલેશનશીપ, અને સ્કીમાનો ઉપયોગ કરે છે. DBMS ડેટાને સ્ટોર કરવા, રીટ્રાઇવ કરવા, ફેરફાર કરવા અને કાઢી નાખવા જેવી આવશ્યક સુવિધા SQL જેવી પ્રમાણભૂત ક્વેરી લેંગ્વેજ દ્વારા પ્રદાન કરે છે. તે એકસાથે અનેક યુઝરને ઍક્સેસ કરવાની મંજૂરી આપે છે અને સુરક્ષાના પાસાઓ પણ જાળવે છે. તે ટ્રાન્ઝેક્શન મેનેજમેન્ટ, બેકઅપ અને ડેટા રિકવરી જેવા જટિલ કાર્યોનું પણ સંચાલન કરે છે. આ ઉપરાંત, તે ડેટાબેઝના કદને વિસ્તૃત કરીને અને વધુ યુઝરનું સંચાલન કરીને વ્યાપકતા પણ પૂરી પાડે છે.

ઓરેકલ (Oracle), આઈબીએમ ડીબી2 (IBM DB2), માઈક્રોસોફ્ટ એક્સેસ (Microsoft Access) અને માઈક્રોસોફ્ટ એસક્યુએલ સર્વર (Microsoft SQL Server) જાણીતા ખાનગી માલિકીનાં (Proprietary) DBMSના ઉદાહરણો છે. આ ઉપરાંત, ઘણાં ઉત્તમ ઓપન સોર્સ ડેટાબેઝ મેનેજમેન્ટ સિસ્ટમ પણ ઉપલબ્ધ છે, જેમ કે કાઉચડીબી (CouchDB), મારિયાડીબી (MariaDB), પોસ્ટગ્રેએસક્યુએલ (PostgreSQL), લિબ્રેઓફિસ બેઝ (LibreOffice Base) અને માયએસક્યુએલ (MySQL). આ દરેકની પોતાની વિશિષ્ટતાઓ અને ક્ષમતાઓ છે જે તેમને વિવિધ ઉપયોગો માટે આદર્શ બનાવે છે.

સામાન્ય રીતે, DBMS કેટલાક સૉફ્ટવેર એપ્લિકેશનનો એક મહત્વપૂર્ણ ભાગ છે, કારણ કે તે વ્યવસાયોને તેમના રોજિંદા કામકાજ અને નિર્ણય લેવાની પ્રક્રિયાઓ માટે જરૂરી વિશાળ માત્રામાં ડેટાનું સંચાલન કરવા માટે એક મજબૂત, સલામત અને અસરકારક માળખું પૂરું પાડે છે.

ડેટાબેઝ મેનેજમેન્ટ સિસ્ટમના ઘટકો (Components of Database Management System)

મોટા ભાગની DBMS સિસ્ટમમાં ડેટાબેઝ સાથે કામ કરવા માટે ચાર મૂળભૂત ઑબ્જેક્ટ્સ અથવા ઘટકો હોય છે, જેમને ટેબલ (Table), ક્વેરી (Query), ફોર્મ (Form) અને રિપોર્ટ (Report) તરીકે ઓળખવામાં આવે છે. આ ઑબ્જેક્ટ કોઈપણ ડેટાબેઝ બનાવવા, ઉપયોગ કરવા અને મેનેજ કરવા માટેના બિલ્ડીંગ બ્લોક છે. ચાલો આ દરેક ઑબ્જેક્ટ વિશે સંક્ષિપ્તમાં જોઈએ.

ટેબલ (Table)

DBMSમાં ડેટા સંગ્રહ માટેનો મૂળભૂત એકમ ટેબલ તરીકે ઓળખાય છે. ટેબલ એ એક દ્વિ-પરિમાણીય માળખું છે, જેમાં ઘણીવાર અનેક રો અને કોલમ હોય છે. કોલમને 'ફીલ્ડ' અને રો ને 'રેકોર્ડ' કહેવામાં આવે છે. ટેબલ સામાન્ય રીતે કોઈ ચોક્કસ એન્ટિટી સાથે સંબંધિત હોય છે, જેમ કે વ્યક્તિ, સ્થળ, એકાઉન્ટ, પરીક્ષા અથવા અન્ય.

Fields

Records

First Name	Last Name	Birth Date	Birth City	Gender
Vidi	Arolkar	12-12-2000	Ahmedabad	Female
Sunny	Jain	09-11-1999	Jaipur	Male
Sazid	Khan	10-01-2001	Lucknow	Male
Tony	Gomes	13-09-2000	Goa	Male

આકૃતિ 1.4 : વ્યક્તિઓના ડેટાને સ્ટોર કરતા ટેબલનો નમૂનો

ચાલો, આકૃતિ 1.4ના પરિપ્રેક્ષ્યમાં એન્ટિટી (entity), ફીલ્ડ (field) અને રેકોર્ડ (record) શબ્દોને સમજીએ.

એન્ટિટી (Entity) : આપણે ટેબલમાં વ્યક્તિ સંબંધિત ડેટાને સ્ટોર કર્યો છે. અહીં, એક વ્યક્તિ એ એક એન્ટિટી બને છે.

ફીલ્ડ (Field) : તે એવા ડેટાના નાના અને અલગ ભાગનું પ્રતિનિધિત્વ કરે છે, જેને એટ્રીબ્યુટ તરીકે ઓળખવામાં આવે છે. સ્વતંત્ર રીતે તેનું કદાચ બહુ મહત્ત્વ ન પણ હોય. અહીં, First Name , Last Name , Birth Date , Birth City અને Gender ને ટેબલના ફીલ્ડ તરીકે ઓળખવામાં આવે છે.

રેકોર્ડ (Record) : એન્ટિટીના એક જ ઉદાહરણ વિશેની વિગતો દર્શાવતા ફીલ્ડના સમૂહને રેકોર્ડ કહેવાય છે First Name , Last Name , Birth Date , Birth City અને Gender ને સંયુક્ત રીતે એક રેકોર્ડ કહેવાય છે. જે એક વ્યક્તિની વિગતો રજૂ કરે છે.

આકૃતિ 1.4માં આપેલા ટેબલને જોતાં, આપણે કહી શકીએ કે આપણે ચાર અલગ-અલગ વ્યક્તિઓના રેકોર્ડને સ્ટોર કર્યા છે અને દરેક વ્યક્તિના પાંચ એટ્રીબ્યુટ છે.

ફોર્મ (Form)

જરૂરિયાત મુજબ આપણે ટેબલમાં ડેટા ઇન્સર્ટ કરી શકીએ છીએ. એકવાર ડેટા ઇન્સર્ટ થઈ જાય પછી, હાલના રેકોર્ડને એડિટ કરવા, ડીલીટ કરવા અથવા ટેબલમાં ઉપલબ્ધ રેકોર્ડ જોવાની જરૂર પડી શકે છે. ફોર્મ આપણને એક ગ્રાફિકલ યુઝર ઇન્ટરફેસ આપે છે જેને વ્યક્તિગત જરૂરિયાત અથવા પસંદગી મુજબનું બનાવી શકાય છે. ફોર્મમાં ચોક્કસ ડેટા એન્ટ્રી ફોર્મેટ, ડિઝાઇન અથવા લેઆઉટ હોઈ શકે છે જે યુઝરનું કાર્ય સરળ બનાવે છે.

ક્વેરી (Queries)

ટેબલમાં રહેલા ડેટાનો ઉપયોગ યુઝર દ્વારા પૂછાયેલા પ્રશ્નોના જવાબ આપવા માટે થાય છે. ડેટાબેઝમાં પૂછવામાં આવેલા પ્રશ્નને ક્વેરી કહેવામાં આવે છે. "વ્યક્તિઓના ટેબલમાં કેટલા પુરુષો છે?" અથવા "સની જૈનની જન્મ તારીખ શું છે?" જેવા પ્રશ્નોના જવાબ આપવા માટે ક્વેરી બનાવી શકાય છે. ક્વેરીનો ઉપયોગ ડેટાબેઝમાં જુદા જુદા ટેબલમાં સમાવિષ્ટ પસંદગીના ડેટા સેટને દર્શાવવા માટે કરી શકાય છે.

રિપોર્ટ (Reports)

ડેટાબેઝમાં ટેબલ અથવા ક્વેરીનું આઉટપુટ રો અને કોલમના સ્વરૂપમાં દર્શાવવામાં આવે છે. જોકે, યુઝર પૂછેલા પ્રશ્નો માટે વધુ સમજાવટવાળું અને ફોર્મેટ કરેલું આઉટપુટ ઈચ્છે છે જેથી વસ્તુઓને યોગ્ય રીતે સમજી શકાય. રિપોર્ટ એ જરૂરી માહિતીને યોગ્ય રીતે ફોર્મેટ કરેલા, વ્યવસ્થિત અને વાંચી શકાય તેવા સ્વરૂપમાં રજૂ કરે છે.

મેક્રો (Macro)

બેઝનો એક વધારાનો ઘટક મેક્રો (Macro) એ યુઝર માટે ખૂબજ મદદરૂપ છે. મેક્રો એ ક્રમબદ્ધ કમાન્ડનો સમૂહ છે. જે ડેટાબેઝમાં પુનરાવર્તિત કાર્યોને સ્વચાલિત કરે છે. તે એક જ ક્લિકથી ક્રિયાઓની શ્રેણી બનાવી શકે છે, જેમ કે ક્વેરી ચલાવવી, રિપોર્ટ ખોલવા અથવા રેકોર્ડ અપડેટ કરવા. આપણે યુઝર ઈન્ટરફેસ મેક્રો બનાવીને ફોર્મને કન્ટ્રોલ કરવા અથવા ટેબલ ઈવેન્ટ દ્વારા ટ્રિગર થતી ક્રિયાઓને સ્વચાલિત કરી શકીએ છીએ. વધુમાં મેક્રોનો ઉપયોગ ચકાસણી કરવા અને નિયમો લાગુ કરવા માટે થઈ શકે છે, જે ડેટા એન્ટ્રીને વધુ વિશ્વસનીય બનાવે છે.

નમૂનારૂપ ડેટાબેઝ બનાવવો (Sample Database Creation)

કોઈપણ એપ્લિકેશન માટે ડેટાબેઝ બનાવવા માટે એક સુવ્યવસ્થિત અભિગમ જરૂરી છે. શરૂઆતમાં, આપણે એપ્લિકેશનની જરૂરિયાતો, તેની પ્રક્રિયાઓ અને વિવિધ પ્રવૃત્તિઓને જોડતા પ્રવાહોને સમજવાની જરૂર છે. આના આધારે, આપણે જરૂરી મુખ્ય એન્ટિટી અને તેમના એટ્રીબ્યુટને ઓળખવા પડશે.

ચાલો, એક રિલેશનલ ડેટા મોડેલનું માળખું બનાવવાનો પ્રયાસ કરીએ જેનો ઉપયોગ શાળાના સંચાલન સંબંધિત પ્રવૃત્તિઓનો ડેટા સ્ટોર કરવા માટે થઈ શકે. શાળાના સંચાલનમાં વિદ્યાર્થીઓ અને શિક્ષકોનું સંચાલન, પરીક્ષાનું સંચાલન, પગારપત્રક અને બીજી ઘણી બધી પ્રવૃત્તિઓનો સમાવેશ થાય છે, પણ આપણે ફક્ત તેના એક નાના ભાગને જોઈશું, જેમાં આપણે વિદ્યાર્થી, શિક્ષક, વિષય, વર્ગ અને ગ્રેડની વિગતો જોઈશું.

હવે આપણે નક્કી કર્યું કે કોનો ડેટા સ્ટોર કરવામાં આવશે. આ બધા શબ્દો, કે જે તમામનો ડેટા સ્ટોર કરવામાં આવશે, તેને એન્ટિટી તરીકે ઓળખવામાં આવે છે. આપણે દરેક એન્ટિટી માટે એક અલગ ટેબલ બનાવીશું. આમ, આ કિસ્સામાં આપણે પાંચ ટેબલ ડિઝાઈન કરીશું:

Student (વિદ્યાર્થી), Teacher (શિક્ષક), Subject (વિષય), Class (વર્ગ) અને Grade (ગ્રેડ).

પછીનું પગલું એ નક્કી કરવાનું છે કે આ દરેક એન્ટિટીમાં કયા એટ્રીબ્યુટ હશે અને તેમાં કઈ પ્રકારની કિંમતો સ્ટોર થશે. આ એટ્રીબ્યુટને ટેબલની અંદર ફીલ્ડ તરીકે દર્શાવવામાં આવશે. આ તમામ એન્ટિટી અને તેમના એટ્રીબ્યુટનું નમૂનારૂપ સ્કીમા ટેબલ 1.1માં દર્શાવેલ છે

એન્ટિટી (Entity)	મુખ્ય એટ્રીબ્યુટ (Key Attributes)
Student	StudentID, FirstName, MiddleName, LastName, BirthDate, Gender, Address ClassID
Teacher	TeacherID, FirstName, MiddleName, LastName, BirthDate, Gender, Address, SubjectID
Subject	SubjectID, SubjectName, ClassID
Class	ClassID, ClassName, TeacherID
Grade	GradeID, StudentID, SubjectID, GradeDate, Marks

ટેબલ 1.1 : શાળા ડેટાબેઝ માટેનો નમૂનારૂપ સ્કીમા

આ સ્કીમામાં શાળાની જરૂરિયાત મુજબ કોઈપણ વધારાની સુવિધાઓ ઉમેરીને સ્કીમાને વિસ્તૃત કરી શકાય છે. ઉદાહરણ તરીકે, આપણે પગારપત્રક (payroll), તાલીમ (training), અથવા ભરતી (recruitment) જેવી વિગતો ઉમેરી શકીએ છીએ.

ચાલો, ટેબલ 1.2, 1.3, 1.4, 1.5 અને 1.6 માં દર્શાવ્યા મુજબ દરેક ટેબલમાં સ્ટોર થનારા એટ્રીબ્યુટની વિગતો સમજવાનો પ્રયાસ કરીએ.

સ્ટુડન્ટ (STUDENT)		
એટ્રીબ્યુટ	**વિગત**	**સ્ટોર થનાર વેલ્યૂની ટાઈપ**
StudentID	વિદ્યાર્થીનો આઈડી સંગ્રહ કરવા માટે વપરાય છે	સંસ્થાની નીતિ પ્રમાણે ટેક્સ્ટ (text) અથવા સંખ્યા (numeric) હોઈ શકે છે
FirstName	વિદ્યાર્થીનું પહેલું નામ સ્ટોર કરવા માટે વપરાય છે	ટેક્સ્ટ વેલ્યૂ
MiddleName	વિદ્યાર્થીનું મધ્ય નામ સ્ટોર કરવા માટે વપરાય છે	ટેક્સ્ટ વેલ્યૂ
LastName	વિદ્યાર્થીનું છેલ્લું નામ સ્ટોર કરવા માટે વપરાય છે	ટેક્સ્ટ વેલ્યૂ
BirthDate	વિદ્યાર્થીની જન્મ તારીખ સ્ટોર કરવા માટે વપરાય છે	ટેક્સ્ટ વેલ્યૂ
Gender	વિદ્યાર્થીની જાતિ સ્ટોર કરવા માટે વપરાય છે	ટેક્સ્ટ વેલ્યૂ
Address	વિદ્યાર્થીનું સરનામું સ્ટોર કરવા માટે વપરાય છે	આલ્ફા-ન્યુમેરિક વેલ્યૂ (જેમાં લખાણ, સંખ્યા તેમજ ખાસ અક્ષરોનો સમાવેશ થઈ શકે છે)
ClassID	વિદ્યાર્થી જે વર્ગમાં અભ્યાસ કરે છે તેનો આઈડી સ્ટોર કરવા માટે વપરાય છે	ક્લાસ ટેબલ (Class table)માં સ્ટોર થનારા ડેટાની ટાઈપ પર આધાર રાખશે

ટેબલ 1.2 : સ્ટુડન્ટ ટેબલની વિગતો

ટીચર (TEACHER)		
એટ્રીબ્યુટ	**વિગત**	**સ્ટોર થનાર વેલ્યૂની ટાઈપ**
TeacherID	શિક્ષકનો આઈડી સ્ટોર કરવા માટે વપરાય છે	સંસ્થાની નીતિ પ્રમાણે ટેક્સ્ટ (text) અથવા સંખ્યા (numeric) હોઈ શકે છે
FirstName	શિક્ષકનું પહેલું નામ સ્ટોર કરવા માટે વપરાય છે	ટેક્સ્ટ વેલ્યૂ
MiddleName	શિક્ષકનું મધ્ય નામ સ્ટોર કરવા માટે વપરાય છે	ટેક્સ્ટ વેલ્યૂ
LastName	શિક્ષકનું છેલ્લું નામ સ્ટોર કરવા માટે વપરાય છે	ટેક્સ્ટ વેલ્યૂ
BirthDate	શિક્ષકની જન્મ તારીખ સ્ટોર કરવા માટે વપરાય છે	ટેક્સ્ટ વેલ્યૂ
Gender	શિક્ષકની જાતિ સ્ટોર કરવા માટે વપરાય છે	ટેક્સ્ટ વેલ્યૂ
Address	શિક્ષકનું સરનામું સ્ટોર કરવા માટે વપરાય છે	આલ્ફા-ન્યુમેરિક વેલ્યૂ (જેમાં લખાણ, સંખ્યા તેમજ ખાસ અક્ષરોનો સમાવેશ થઈ શકે છે)
SubjectID	શિક્ષક જે વિષય ભણાવે છે તેનો આઈડી સ્ટોર કરવા માટે વપરાય છે	સબ્જેક્ટ ટેબલ (Subject table)માં સ્ટોર થનારા ડેટાની ટાઈપ પર આધાર રાખશે

ટેબલ 1.3 : ટીચર ટેબલની વિગતો

સબ્જેક્ટ (SUBJECT)		
એટ્રીબ્યુટ	**વિગત**	**સ્ટોર થનાર વેલ્યૂની ટાઈપ**
SubjectID	વિષયનો આઈડી સ્ટોર કરવા માટે વપરાય છે	સંસ્થાની નીતિ પ્રમાણે ટેક્સ્ટ (text) અથવા સંખ્યા (numeric) હોઈ શકે છે
SubjectName	વિષયનું નામ સ્ટોર કરવા માટે વપરાય છે	ટેક્સ્ટ વેલ્યૂ
ClassID	જે ક્લાસમાં વિષય ભણાવાય છે, તેનો આઈડી સ્ટોર કરવા માટે વપરાય છે	ક્લાસ ટેબલ (class table)માં સ્ટોર થનારા ડેટાની ટાઈપ પર આધાર રાખશે

ટેબલ 1.4 : સબ્જેક્ટ ટેબલની વિગતો

ક્લાસ (CLASS)		
એટ્રીબ્યુટ	**વિગત**	**સ્ટોર થનાર વેલ્યૂની ટાઈપ**
ClassID	ક્લાસ આઈડી સ્ટોર કરવા માટે વપરાય છે	સંસ્થાની નીતિ પ્રમાણે ટેક્સ્ટ (text) અથવા સંખ્યા (numeric) હોઈ શકે છે
ClassName	ક્લાસનું નામ સ્ટોર કરવા માટે વપરાય છે	ટેક્સ્ટ વેલ્યૂ
TeacherID	શિક્ષકનો આઈડી સ્ટોર કરવા માટે વપરાય છે જે ક્લાસનું સંચાલન કરે છે	ટીચર ટેબલ (teacher table)માં સ્ટોર થનારા ડેટાની ટાઈપ પર આધાર રાખશે

ટેબલ 1.5 : ક્લાસ ટેબલની વિગતો

ગ્રેડ (GRADE)		
એટ્રીબ્યુટ	**વિગત**	**સ્ટોર થનાર વેલ્યૂની ટાઈપ**
GradeID	ગ્રેડ આઈડી સ્ટોર કરવા માટે વપરાય છે	ઈન્ટીજર નંબર
StudentID	જે વિદ્યાર્થીને ગ્રેડ આપવામાં આવ્યો છે, તેનો આઈડી સ્ટોર કરવા માટે વપરાય છે	સ્ટુડન્ટ ટેબલ (student table)માં સ્ટોર થનારા ડેટાની ટાઈપ પર આધાર રાખશે
SubjectID	જે વિષય માટે વિદ્યાર્થીને ગ્રેડ આપવામાં આવ્યો છે, તેનો આઈડી સ્ટોર કરવા માટે વપરાય છે	સબ્જેક્ટ ટેબલ (subject table) માં સ્ટોર થનારા ડેટાની ટાઈપ પર આધાર રાખશે
GradeDate	વિદ્યાર્થીને ગ્રેડ કઈ તારીખે આપવામાં આવ્યો તે સ્ટોર કરવા માટે વપરાય છે	ડેટ વેલ્યૂ
Marks	માર્ક્સ સ્ટોર કરવા માટે વપરાય છે	ન્યુમેરિક વેલ્યૂ

ટેબલ 1.6 : ગ્રેડ ટેબલની વિગતો

આગળના પ્રકરણમાં જ્યારે આપણે લિબ્રેઓફિસ બેઝ (LibreOffice Base)નો ઉપયોગ કરીને ડેટાબેઝ બનાવીશું, ત્યારે આપણે આ ટેબલનો ઉપયોગ કરીશું. આ તબક્કે, ચાલો આપણે રિલેશનલ ડેટા મોડેલમાં કી (Keys)ના ખ્યાલ વિશે વાત કરીએ. અગાઉ જણાવ્યા મુજબ, રિલેશનલ ડેટા મોડેલમાં ડેટાને રો અને કોલમથી બનેલા ટેબલ (રિલેશનશિપ)માં ગોઠવવામાં આવે છે. કી આપણને આ રિલેશનશિપને યોગ્ય રીતે મેન્ટેઈન

કરવામાં મદદ કરે છે. ચાર કી; પ્રાયમરી (primary), ફોરેન (foreign), કેન્ડિડેટ (candidate) અને અલ્ટરનેટ (alternate) કી મહત્વપૂર્ણ ભૂમિકા ભજવે છે.

પ્રાઈમરી કી (Primary Key)

એવું ફીલ્ડ જે ટેબલમાં આવેલી રોને અનન્ય (યુનિક) રીતે ઓળખે છે, તેને પ્રાઈમરી કી કહેવામાં આવે છે. પ્રાઈમરી કી એક અથવા વધુ કોલમની બનેલી હોઈ શકે છે જેમાં વિશિષ્ટ વેલ્યૂ હોય. ટેબલમાં એક કરતાં વધારે રો માટે પ્રાઈમરી કીની વેલ્યૂ સમાન હોઈ શકે નહીં, અને તે ફીલ્ડની વેલ્યૂ ખાલી (empty) પણ છોડી શકાય નહીં. ઉદાહરણ તરીકે, સ્ટુડન્ટ ટેબલમાં દરેક વિદ્યાર્થી પાસે એક વિશિષ્ટ StudentID હોય છે, જેને પ્રાઈમરી કી તરીકે ગણી શકાય. જો, કોઈ ટેબલમાં આપણે રેકોર્ડને ઓળખવા માટે એક કરતાં વધુ ફીલ્ડનો ઉપયોગ કરીએ, તો ફીલ્ડ્સના આ સંયુક્ત સમૂહને કૉમ્પોઝિટ કી (Composite Key) તરીકે ઓળખવામાં આવે છે.

ફોરેન કી (Foreign Key)

ફોરેન કી એ ટેબલમાં આવેલું એવું ફીલ્ડ છે જેનો ઉપયોગ બીજા ટેબલમાંના રેકોર્ડને યુનિક રીતે ઓળખવા માટે કરી શકાય છે, પછી ભલે તે એકલું હોય કે અન્ય ફીલ્ડ સાથે સંયોજનમાં હોય. આ ફોરેન કી બે ટેબલ વચ્ચે રિલેશનશિપ સ્થાપિત કરવામાં મદદ કરે છે. ઉદાહરણ તરીકે, સ્ટુડન્ટ ટેબલમાં આવેલું ClassID ફીલ્ડ ફોરેન કી તરીકે કાર્ય કરે છે; તે આપણને જણાવે છે કે વિદ્યાર્થી કયા ક્લાસમાં અભ્યાસ કરે છે. તે ક્લાસ ટેબલમાં યુનિક રેકોર્ડને પણ ઓળખે છે. આમ, આપણે કહી શકીએ કે ClassID કી Student અને Class ટેબલને જોડે છે.

કેન્ડિડેટ કી (Candidate Key)

ટેબલ માટેની કેન્ડિડેટ કી એ તમામ ફીલ્ડ માટેની વેલ્યૂ છે જે પ્રાઈમરી કી બનવા માટે યોગ્ય છે. આવા ફીલ્ડ્સમાં કોઈ ડુપ્લિકેટ વેલ્યૂ હોવી જોઈએ નહીં, અને તેમને ખાલી છોડી શકાય નહીં. તેથી, ક્લાસ ટેબલમાં ClassID અને ClassName બંને સંભવિત કેન્ડિડેટ કી છે.

અલ્ટરનેટ કી (Alternate Key)

કેન્ડિડેટ કીમાંથી એક અથવા બે કીને ટેબલ માટે પ્રાઈમરી કી તરીકે પસંદ કરવામાં આવે છે. જે કેન્ડિડેટ કી બાકી રહી જાય છે તેને અલ્ટરનેટ કી તરીકે ઓળખવામાં આવે છે. તેથી, જો ClassIDને પ્રાઈમરી કી તરીકે ઉપયોગમાં લેવામાં આવે, તો ક્લાસ ટેબલમાં ClassName એ અલ્ટરનેટ કી છે.

ડેટા અને ડેટા ટાઈપ (Data and Data type)

આપણે ટેબલ 1.2માં પહેલાથી જ જોયું છે કે જુદા જુદા ફીલ્ડમાં સ્ટોર થતો ડેટા અલગ અલગ હોઈ શકે છે. ઉદાહરણ તરીકે, આપણા ડેટા સેટમાં FirstName, LastName અને Gender ફીલ્ડમાં ટેક્સ્ટ ડેટા છે, જ્યારે BirthDate ફીલ્ડમાં ડેટ છે. આપણે Address નામના ફીલ્ડના કિસ્સામાં આલ્ફાન્યુમેરિક ડેટા અને Age નામના ફીલ્ડમાં ન્યુમેરિક ડેટા અને Percentage નામના ફીલ્ડમાં દશાંશ સંખ્યા (decimal number)નો પણ ઉપયોગ કરી શકીએ છીએ.

આમ, ડેટા ટાઈપ એ ચોક્કસ ફીલ્ડમાં સ્ટોર થનારા ડેટાના પ્રકારનો ઉલ્લેખ કરે છે. ડેટા ટાઈપ એ પણ નક્કી કરે છે કે ચોક્કસ ફીલ્ડને કેટલી મેમરી ફાળવવામાં આવશે. એક જ ડેટા ટાઈપની અંદર પણ મેમરીની સાઈઝ અલગ અલગ હોઈ શકે છે. મોટા ભાગના DBMS અમુક સામાન્ય ડેટા ટાઈપને સપોર્ટ કરે છે જેમ કે ટેક્સ્ટ (text), ન્યુમેરિક (numeric), કરન્સી (currency), ડેટ/ટાઈમ (date/time), બૂલીયન (boolean) અને બાઈનરી (binary). આ વિભાગમાં આપણે લિબ્રેઓફીસ બેઝની ડેટા ટાઈપની ચર્ચા કરીશું.

ટેક્સ્ટ ડેટા ટાઈપ (Text Data Type)

ટેક્સ્ટ ડેટા ટાઈપ આપણને ફીલ્ડમાં ડેટા વેલ્યૂ તરીકે આલ્ફાબેટ્સ (મૂળાક્ષરો), સંખ્યાઓ અથવા ખાસ અક્ષરો (spe-

cial characters) ના સંયોજનને સ્ટોર કરવાની મંજૂરી આપે છે. આવા ડેટા પર આપણે કોઈ ગાણિતિક ગણતરીઓ કરી શકતા નથી. જે ફીલ્ડ ટેક્સ્ટ ડેટા ટાઈપનો ઉપયોગ કરી શકે છે તેના કેટલાક ઉદાહરણો નીચે પ્રમાણે છે.

AADHAAR Card Number, FirstName, Delivery Address અને Email. ટેક્સ્ટ ડેટા ટાઈપના ચાર અલગ અલગ પ્રકારો છે, અને તે મુખ્યત્વે સ્પેસનો વપરાશ કેવી રીતે કરે છે તે પ્રમાણે અલગ પડે છે. ટેબલ 1.7 માં લિબ્રેઓફીસ બેઝમાં ઉપયોગમાં લઈ શકાય તેવી વિવિધ ટેક્સ્ટ ડેટા ટાઈપની યાદી દર્શાવવામાં આવી છે.

ટાઈપ	બેઝમાં ફીલ્ડ ડેટા ટાઈપ	મેમરીમાં સ્ટોરેજ
ટેક્સ્ટ	VARCHAR	વેરીએબલ
ટેક્સ્ટ	VARCHAR_IGNORECASE	વેરીએબલ
ટેક્સ્ટ (ફિક્ષ)	CHAR	ફિક્ષ્ડ
મેમો	LONGVARCHAR	વેરીએબલ

ટેબલ 1.7 : ટેક્સ્ટ ડેટા ટાઈપ

VARCHAR : આ શબ્દ વેરીએબલ કેરેક્ટર્સનું પ્રતિનિધિત્વ કરે છે. જ્યારે VARCHARનો ઉપયોગ કરવામાં આવે છે, ત્યારે ફીલ્ડમાં એન્ટર કરી શકાય તેવા કેરેક્ટર્સની મહત્તમ સંખ્યાને વ્યાખ્યાયિત કરવી શક્ય છે. ઉદાહરણ તરીકે, જો કોઈ ફીલ્ડની ડેટા ટાઈપ VARCHAR(30) હોય તો આપણે તેમાં મહત્તમ 30 કેરેક્ટર્સ ડેટા તરીકે એન્ટર કરી શકીએ છીએ. જો યુઝર 30 કેરેક્ટર્સ કરતાં ઓછો ડેટા એન્ટર કરે તો જરૂરી સ્પેસ DBMS દ્વારા આપોઆપ ફાળવવામાં આવશે.

VARCHAR_IGNORECASE : આ VARCHAR જેવું જ છે, પરંતુ તે VARCHARનું કેસ ઇન-સેન્સિટિવ વર્ઝન છે. અહીં ઇગ્નોર-કેસ સર્ચ કરતી વખતે મહત્વપૂર્ણ ભૂમિકા ભજવે છે. ધારો કે યુઝર city નામના ફીલ્ડમાં "AHMEDABAD" ડેટા સ્ટોર કર્યો છે, હવે સર્ચ કરતી વખતે જો યુઝર "Ahmedabad" શોધે તો પણ તેને આઉટપુટ મળશે.

CHAR : આ ડેટા ટાઈપ એક નિશ્ચિત કદના ટેક્સ્ટ ફીલ્ડનો ઉલ્લેખ કરે છે. VARCHAR ડેટા ટાઈપની જેમ જ, ફીલ્ડની સાઈઝ ફીલ્ડ બનાવતી વખતે સેટ કરવામાં આવે છે. જો ફીલ્ડમાં એન્ટર કરાયેલ ટેક્સ્ટ તમામ જગ્યાનો ઉપયોગ ન કરે, તો બાકીના કેરેક્ટરને સ્પેસ વડે ભરી દેવામાં આવે છે. આ ડેટા ટાઈપનો ઉપયોગ ત્યારે શ્રેષ્ઠ છે જ્યારે આપણે PAN અથવા ક્રેડિટ કાર્ડ નંબરની જેમ પહેલેથી નક્કી હોય તેટલી સંખ્યામાં કેરેક્ટરની જરૂર હોય.

LONGVARCHAR : આ ડેટા ટાઈપ ખૂબ મોટા ટેક્સ્ટ બ્લોકને સ્ટોર કરવા માટે રચાયેલ છે. તે યુઝર દ્વારા નિર્દેશિત કેરેક્ટરની મહત્તમ લંબાઈ સુધીનો ડેટા સ્ટોર કરે છે. તેનો ઉપયોગ સામાન્ય રીતે 255 કેરેક્ટરથી વધુ ડેટા સ્ટોર કરવા માટે થાય છે. જેનું સારું ઉદાહરણ એક લેખ છે, જે 64000 કેરેક્ટર સુધીનો હોઈ શકે છે.

ન્યુમેરિક ડેટા ટાઈપ (Numeric Data Type)

આ ડેટા ટાઈપનો ઉપયોગ સંખ્યાત્મક સ્વરૂપના ડેટાને સ્ટોર કરવા માટે થાય છે. ન્યુમેરિક ડેટા પૂર્ણાંક સંખ્યાઓ અથવા દશાંશ સંખ્યાઓના સ્વરૂપમાં હોઈ શકે છે. પૂર્ણાંક અને દશાંશ સંખ્યાઓ બંને સાઇન અથવા અનસાઇન હોઈ શકે છે. આવા ડેટા પર ગાણિતિક ક્રિયાઓ કરવી શક્ય છે. જે ફીલ્ડ્સ ન્યુમેરિક ડેટા ટાઈપનો ઉપયોગ કરી શકે છે તેના કેટલાક ઉદાહરણો Marks, Salary, Interest અને TotalPayment છે. લિબ્રેઓફીસ બેઝમાં ચાર ઇન્ટિજર ડેટા ટાઈપ છે જે સામાન્ય રીતે સાઈઝમાં અલગ હોય છે, અને ચાર ફ્લોટિંગ-પોઇન્ટ ડેટા ટાઈપ છે જે

ચોક્સાઈના સ્તરમાં અલગ હોય છે. ટેબલ 1.8 વિવિધ ન્યુમેરિક ડેટા ટાઈપની યાદી દર્શાવે છે, તેની સાથે તે ઉપયોગમાં લેતા બિટ્સ/બાઈટ્સ ની સંખ્યા અને તેની રેન્જ પણ દર્શાવે છે.

ટાઈપ	બેઝમાં ફીલ્ડ ડેટા ટાઈપ	વેલ્યૂની રેન્જ	મેમરીમાં જરૂરી બાઈટ્સ
ટાઈની ઈન્ટિજર	TINYINT	0 to 255 જો અનસાઈન હોય તો –128 to +127 જો સાઈન હોય તો	1 બાઈટ
સ્મોલ ઈન્ટિજર	SMALLINT	0 to 65535 જો અનસાઈન હોય તો –32768 to +32767 જો સાઈન હોય તો	2 બાઈટ
ઈન્ટિજર	INTEGER	0 to 4294967295 –2147483648 to +2147483647	4 બાઈટ
બીગઈન્ટ	BIGINT	0 to 18446744073709551615	8 બાઈટ
નંબર	NUMERIC	અમર્યાદિત (Unlimited)	વેરીએબલ
ડેસીમલ	DECIMAL	અમર્યાદિત (Unlimited)	વેરીએબલ
ફ્લોટ	FLOAT	$5 \times 10^{(-234)}$ to $1.79 \times 10^{(308)}$	4 બાઈટ
રીઅલ	REAL	ચોક્કસ નહીં (Not exact)	4 બાઈટ
ડબલ	DOUBLE	મહત્તમ 15 દશાંશ સુધી કરી શકાય	8 બાઈટ

ટેબલ 1.8 : ન્યુમેરિક ડેટા ટાઈપ

TINYINT : તે ઈન્ટિજર ડેટા ટાઈપમાં સૌથી નાનું છે. તેનું કદ એક બાઈટ છે, તેથી તે મેમરીમાં 8 બિટ જેટલી જગ્યા રોકે છે.

SMALLINT : તે TINYINTના કદ કરતાં બમણું છે. તેનું કદ 2 બાઈટ્સ છે, તેથી તે મેમરીમાં 16 બિટ જેટલી જગ્યા રોકે છે.

INTEGER અથવા INT : સૌથી વધુ ઉપયોગમાં લેવાતું ન્યુમેરિક ડેટા ટાઈપ INTEGER છે. તેનું કદ 4 બાઈટ્સ છે, તેથી તે મેમરીમાં 32 બિટ જેટલી જગ્યા રોકે છે.

BIGINT : મોટા ભાગના સરળ પ્રોગ્રામમાં તેની જરૂર પડતી નથી, તેથી તેનો ભાગ્યે જ ઉપયોગ થાય છે. તેનું કદ 8 બાઈટ્સ છે, તેથી તે મેમરીમાં 64 બિટ જેટલી જગ્યા રોકે છે.

ફ્લોટિંગ-પોઈન્ટ નંબર્સ દશાંશવાળી સંખ્યાઓ અથવા વાસ્તવિક સંખ્યાઓ છે. તે દશાંશ બિંદુ દ્વારા વિભાજિત સંપૂર્ણ ભાગ અને આંશિક ભાગથી બનેલા હોય છે.

DECIMAL અને NUMERIC : આ ડેટા ટાઈપ અમર્યાદિત રેન્જ ધરાવે છે. તેમને ડિફાઈન કરતી વખતે, આપણે દશાંશ બિંદુ પછી આવનારા ડિજિટ્સની સંખ્યા સાથે, કુલ ડિજિટ્સની સંખ્યાનો ઉલ્લેખ કરવાની જરૂર રહે છે. ઉદાહરણ તરીકે, DECIMAL (10, 2)નો અર્થ એ થશે કે ફીલ્ડમાં મહત્તમ 10 સ્થાન અને દશાંશ બિંદુ પછી બે ડિજિટ્સ હશે. DECIMAL અને NUMERICની ચોકસાઈ લગભગ પૂર્ણ હોય છે.

DOUBLE અથવા REAL : આ ડેટા ટાઈપ એક મર્યાદિત રેન્જ ધરાવે છે અને તેમાં મહત્તમ 15 દશાંશ સ્થાનો હોઈ શકે છે. આ ડેટા ટાઈપની ચોકસાઈ એટલી સારી હોતી નથી.

કરન્સી ડેટા ટાઈપ (Currency Data Type)

કરન્સી ડેટા ટાઈપ આપણને નાણાકીય મૂલ્યો દર્શાવતી ન્યુમેરિક વેલ્યૂ સ્ટોર કરવાની સગવડ આપે છે. આપણે વિવિધ દેશોના ચલણનો ઉપયોગ કરીને ડેટા સ્ટોર કરી શકીએ છીએ. ઉદાહરણ તરીકે, ₹ 1250.50, $ 1500 અથવા £ 700. આ ડેટા ટાઈપ ખાતરી કરે છે કે કરન્સીને લગતી ગણતરીઓ સચોટ છે અને નાણાકીય વેલ્યૂનું ડિસ્પ્લે યોગ્ય રીતે ફોર્મેટ થયેલું છે.

ડેટ/ટાઈમ ડેટા ટાઈપ (Date/Time Data Type)

ડેટ/ટાઈમ ડેટા ટાઈપનો ઉપયોગ વર્ષ, મહિનો, દિવસ, કલાક, મિનિટ, સેકન્ડ અને સેકન્ડના અપૂર્ણાંક જેવી માહિતીને સ્ટોર કરવા માટે થાય છે. જે ફીલ્ડ ડેટ/ટાઈમ ડેટા ટાઈપનો ઉપયોગ કરી શકે છે તેના કેટલાક ઉદાહરણો Date of Joining, Birth Date, In Time અને Out Time છે. લિબ્રેઓફીસ બેઝ માં ડેટ/ટાઈમ ડેટા ટાઈપમાં ત્રણ પ્રકારો છે, જે સ્ટોર કરેલા કન્ટેન્ટ પ્રમાણે અલગ પડે છે. ટેબલ 1.9 વિવિધ ડેટ/ટાઈમ ડેટા ટાઈપ દર્શાવે છે.

ટાઈપ	બેઝમાં ફીલ્ડ ડેટા ટાઈપ	વેલ્યૂની રેન્જ	મેમરીમાં જરૂરી બાઈટ્સ
ડેટ	DATE	-	4 બાઈટ
ટાઈમ	TIME	-	4 બાઈટ
ડેટ/ટાઈમ	TIMESTAMP	એડજસ્ટેબલ (0.5 – 5 with milliseconds)	8 બાઈટ

ટેબલ 1.9 : ડેટ/ટાઈમ ડેટા ટાઈપ

DATE : તે સિસ્ટમની તારીખ સ્ટોર કરે છે. તારીખ એન્ટર કરવા માટેનું ફોર્મેટ YYYY-MM-DD છે. ઉદાહરણ તરીકે, 2025-05-30

TIME : તે ઘડિયાળનો સમય સ્ટોર કરે છે. સમય માટેનું ડિફોલ્ટ ફોર્મેટ 24 કલાકનું હોય છે, જેમ કે HH:MM:SS (કલાક:મિનિટ:સેકન્ડ), એટલે કે 3:30:36 PM (બપોરે 3 વાગીને 30 મિનિટ અને 36 સેકન્ડ) માટે 15:30:36.

TIMESTAMP અથવા DATETIME : તારીખ અને સમય બંનેનું સંયોજન છે. અહીં ડેટાનું ડિફોલ્ટ ફોર્મેટ

YYYY-MM-DD HH:MM:SS છે, એટલે કે 2025-05-30 15:30:36. આ ડેટા ટાઈપનો ઉપયોગ Time of Transaction (ટાઈમ ઑફ ટ્રાન્ઝેક્શન) જેવા ઓડિટ ફીલ્ડમાં થાય છે.

બૂલીયન ડેટા ટાઈપ (Boolean Data Type)

બૂલીયન ડેટા ટાઈપ આપણને ફક્ત બે વેલ્યૂ True અથવા False સ્ટોર કરવાની સવલત આપે છે. આ બે વેલ્યૂ Yes/No અને On/Off જેવા અનેક ફોર્મેટમાં પણ રજૂ કરી શકાય છે.

બાઈનરી ડેટા ટાઈપ (Binary Data Type) :

બાઈનરી ડેટા આપણને ડિજિટાઈઝ્ડ ઈમેજ, વીડિયો, અવાજ અથવા ફાઈલ જેવી માહિતી સ્ટોર કરવાની મંજૂરી આપે છે. આ તમામ માહિતી શૂન્ય (0) અને એક (1)ની લાંબી સ્ટ્રિંગ તરીકે સ્ટોર થાય છે. તેના ઉદાહરણો Profile Picture અને Voice Sample છે. ટેબલ 1.10 વિવિધ બાઈનરી ડેટા ટાઈપની સૂચિ દર્શાવે છે.

ટાઈપ	બેઝમાં ફીલ્ડ ડેટા ટાઈપ	વેલ્યૂની રેન્જ	મેમરીમાં જરૂરી બાઇટ્સ
બાઈનરી ફીલ્ડ (ફિક્ષ)	BINARY	ઈન્ટિજર જેટલી	ફિક્સ
બાઈનરી ફીલ્ડ	VARBINARY	ઈન્ટિજર જેટલી	વેરીએબલ
ઈમેજ	LONGVARBINARY	ઈન્ટિજર જેટલી	વેરીએબલ - ખૂબ મોટી ઈમેજ માટે ઉપયોગી.

ટેબલ 1.10 : બાઈનરી ડેટા ટાઈપ

સારાંશ

આ પ્રકરણમાં આપણે ડેટા અને માહિતી વચ્ચેનો તફાવત સમજ્યા. ડેટામાં કાચી હકીકતો હોય છે, જ્યારે માહિતી એ ડેટાનું વ્યવસ્થિત સ્વરૂપ છે. આગળ, આપણે ડેટાબેઝમાં ડેટા મોડેલ વિશે જોયું, જે દર્શાવે છે કે ડેટા કેવી રીતે સ્ટોર થાય છે અને રીટ્રાઈવ કરવામાં આવે છે. ડેટાબેઝ મેનેજમેન્ટ સિસ્ટમ (DBMS) વિવિધ ડેટા મોડેલનો ઉપયોગ કરે છે, જેમાં ફ્લેટ, હાયરાર્કિકલ અને રિલેશનલ મોડેલનો સમાવેશ થાય છે. રિલેશનલ મોડેલ ડેટાબેઝની અંદરના ટેબલને જોડે છે. આપણે ડેટાબેઝ અને DBMSના ખ્યાલો પણ શીખ્યા. ડેટાબેઝ એ ડેટાનો એક વ્યવસ્થિત સંગ્રહ છે અને DBMS આ ડેટાને એડ કરવા, અપડેટ કરવા અને એક્સેસ કરવામાં મદદ કરે છે. આપણે જાણ્યું કે વાસ્તવિક દુનિયાની વસ્તુઓને એન્ટિટી કહેવામાં આવે છે, અને તેમની ચોક્કસ વિગતો એટ્રીબ્યુટ તરીકે ઓળખાય છે. ટેબલ એ એન્ટિટીનું પ્રતિનિધિત્વ છે અને તેમાં રેકોર્ડ હોય છે જે રો અને કોલમના સ્વરૂપમાં ગોઠવાયેલા હોય છે, જેમાં સૌથી નાનો એકમ ફીલ્ડ છે જે એટ્રીબ્યુટનું પ્રતિનિધિત્વ કરે છે. ફીલ્ડમાં સ્ટોર કરેલ ડેટા સંખ્યાઓ, અક્ષરો અથવા ખાસ અક્ષરો હોઈ શકે છે. આ રીતે, એક રેકોર્ડ એ ચોક્કસ એન્ટિટી માટે ડેટા વેલ્યૂનો સંપૂર્ણ સમૂહ છે. પ્રાઈમરી કી એક રોને યુનિક રીતે ઓળખે છે, અને ફોરેન કી વિવિધ ટેબલને જોડે છે. કેન્ડિડેટ કી એ ટેબલ માટે સંભવિત મુખ્ય કી હોય છે. આ પ્રકરણે આવનારા પ્રકરણોમાં ડેટાબેઝના વધુ ખ્યાલો શીખવા માટે એક મજબૂત પાયો નાખ્યો છે.

સ્વાધ્યાય

1. ડેટા અને માહિતીને યોગ્ય ઉદાહરણો સાથે સમજાવો.
2. ડેટાબેઝ અને ડેટાબેઝ મેનેજમેન્ટ સિસ્ટમ વચ્ચેનો તફાવત સ્પષ્ટ કરો.
3. ટેબલ, રેકોર્ડ અને ફીલ્ડની વ્યાખ્યા આપો.
4. ડેટાબેઝની જરૂરિયાત શા માટે છે તે સમજાવો.
5. ડેટાબેઝના સામાન્ય ઘટકો પર નોંધ લખો.
6. ડેટા ટાઈપ શું છે? બેઝમાં ઉપલબ્ધ ડેટા ટાઈપની યાદી બનાવો.
7. CHAR અને VARCHAR વચ્ચે શું તફાવત છે?
8. કરન્સી ડેટા ટાઈપનો ઉપયોગ શું છે?
9. ક્યારે બૂલીયન ડેટા ટાઈપનો ઉપયોગ કરવો જોઈએ?
10. STUDENT એન્ટિટીના કોઈપણ પાંચ એટ્રીબ્યુટ ઓળખો.
11. **સાચું કે ખોટું જણાવો.**
    1) ડેટાબેઝ એ એક સૉફ્ટવેર છે જે આપણને ડેટામાં ફેરફાર કરવાની મંજૂરી આપે છે.

2) જો આપણે બસ વિશે ડેટા સ્ટોર કરીએ તો તેને એક એન્ટિટી ગણી શકાય.

3) ફોર્મ આપણને ટેબલમાંથી ડેટા ડીલીટ કરવાની મંજૂરી આપે છે.

4) ટેક્સ્ટ ડેટા ટાઇપમાં ન્યુમેરિક મૂલ્ય સ્ટોર કરવું શક્ય છે.

5) ઓડિયો ક્લિપને ટાઇમ ડેટા ટાઇપમાં સ્ટોર કરી શકાય છે.

12. ખાલી જગ્યા પૂરો.

(1) ડેટાબેઝ એ સંબંધિત ડેટા આઇટમનો ____________ છે.

(2) એન્ટિટીના એટ્રીબ્યુટસને ટેબલમાં ____________ તરીકે ઓળખવામાં આવે છે.

(3) ટેબલમાં એન્ટિટીની વિગતો આપતા ____________ નો સમૂહ હોય છે.

(4) BIGINT મેમરી સ્ટોરેજ માટે ____________ બાઇટ્સનો ઉપયોગ કરે છે.

(5) ડિજિટાઇઝ્ડ ઇમેજીસ ____________ ડેટા ટાઇપ માં સ્ટોર કરી શકાય છે.

13. બહુવિકલ્પી પ્રશ્નો. સૌથી યોગ્ય જવાબ પસંદ કરો.

(1) નીચેનામાંથી કયું DBMS નું પૂરું નામ છે?

(a) ડેટા બ્લોક મેનેજમેન્ટ સિસ્ટમ (Data Block Management System)

(b) ડેટા બેઝ મેનેજમેન્ટ સિસ્ટમ (Data Base Management System)

(c) ડેટા બાઇટ મેનેજમેન્ટ સિસ્ટમ (Data Byte Management System)

(d) ડેટા બિટ મેનેજમેન્ટ સિસ્ટમ (Data Bit Management System)

(2) નીચેનામાંથી કયો શબ્દ પ્રોસેસ કરેલા ડેટાનો ઉલ્લેખ કરે છે?

(a) ઇન્ફોર્મેશન (b) રૉ (c) ફેક્ટ્સ (d) ફીલ્ડ

(3) નીચેનામાંથી કયું ડેટા મોડેલ નથી?

(a) ફ્લેટ (Flat) (b) રીલેશનલ (Relational)

(c) રોટેશનલ (Rotational) (d) A અને B બંન્ને

(4) નીચેનામાંથી કયું ડેટાબેઝમાં StudentNameનું શ્રેષ્ઠ વર્ણન કરે છે?

(a) રિલેશનશિપ (b) એટ્રિબ્યુટ (c) એન્ટિટી (d) ડેટા

(5) નીચેનામાંથી કયું ડેટાબેઝ નથી?

(a) MySQL (b) લિબ્રેઓફિસ બેઝ (c) લિબ્રેઓફિસ ઇમ્પ્રેસ (d) SQL સર્વર

(6) નીચેનામાંથી કયું ડેટાબેઝનું સામાન્ય ઘટક નથી?

(a) ચાર્ટ (b) ટેબલ (c) ક્વેરીઝ (d) ફોર્મ

(7) બેઝ ડેટાબેઝમાં ઇમેજ સ્ટોર કરવા માટે નીચેનામાંથી કઈ ડેટા ટાઇપ નો ઉપયોગ થાય છે?

(a) ટેક્સ્ટ (Text) (b) બાઇનરી (Binary) (c) બૂલીયન (Boolean) (d) બાઇટ (Byte)

(8) નીચેનામાંથી કયા સ્વરૂપમાં ડેટા વેલ્યૂ રજૂ કરી શકાતી નથી?

(a) ટેક્સ્ટ (Text) (b) ન્યુમેરિક (Numeric)

(c) આલ્ફાન્યુમેરિક (Alphanumeric) (d) પિક્ચર (Picture)

(9) નીચેનામાંથી કયું આપણને સરળ અને યુઝર ફ્રેન્ડલી રીતે ટેબલમાં ડેટા એન્ટર કરવાની મંજૂરી આપે છે?

(a) રિપોર્ટ (b) ક્વેરી (c) ફોર્મ (d) ફીલ્ડ

(10) નીચેનામાંથી કોનો ઉપયોગ સંખ્યાઓ સ્ટોર કરવા માટે કરી શકાતો નથી?

(a) TINYINT (b) DOUBLE (c) DATE (d) CHAR

પ્રાયોગિક સ્વાધ્યાય

1. નીચેની એન્ટિટીઝ માટે તેના એટ્રિબ્યુટસ ઓળખીને ડેટા સ્ટોર કરવા માટે યોગ્ય ડેટા ટાઈપ નક્કી કરો.

a. બુક (Book)

b. કસ્ટમર (Customer)

c. ડીપાર્ટમેન્ટ (Department)

d. એમ્પ્લોયી (Employee)

e. ફ્લાઈટ (Flight)

f. મેગેઝીન (Magazine)

g. મુવી (Movie)

h. પ્રોડક્ટ (Product)

i. સેલ્સમેન (Salesman)

j. ટ્રેન (Train)

k. વીહિકલ (Vehicle)

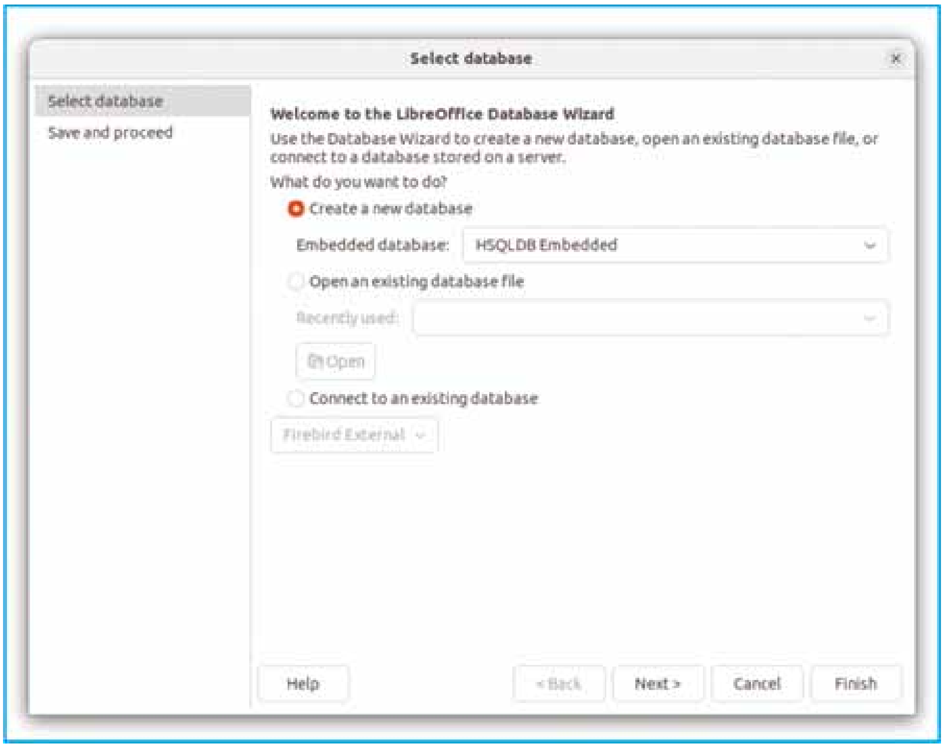
2 લિબ્રેઑફિસ બેઝનો પરિચય

પરિચય

લિબ્રેઑફિસ બેઝ એ ફ્રી અને ઓપન સોર્સ ડેસ્કટોપ ડેટાબેઝ ટૂલ છે. તે લિબ્રેઑફિસ સ્યૂટના ભાગ રૂપે આવે છે. તે MySQL, MS Access, અને PostgreSQL જેવા મુખ્ય ડેટાબેઝ એન્જિનને સપોર્ટ કરે છે. બેઝમાં ડિફોલ્ટ HSQL રિલેશનલ ડેટાબેઝ એન્જિન સામેલ છે. યુઝર્સ વિઝાર્ડ્સ અને પહેલેથી તૈયાર કરેલા ટેમ્પ્લેટની મદદથી ટેબલ, ક્વેરી, ફોર્મ અને રીપોર્ટ બનાવી શકે છે. વધારામાં બેઝ અન્ય લિબ્રેઑફિસ એપ્લિકેશન સાથે પણ સારી રીતે ઈન્ટીગ્રેટ થાય છે, રાઇટરમાં મેઇલ મર્જ માટે ડેટા પૂરો પાડે છે અને વિશ્લેષણ માટે કેલ્કમાં લીન્કડ ડેટા બનાવે છે. આ પ્રકરણમાં, આપણે લિબ્રેઑફિસ બેઝનો ઉપયોગ કેવી રીતે કરવો અને એક નવો ડેટાબેઝ તથા ટેબલ કેવી રીતે બનાવવા તે શીખીશું.

બેઝ ખોલવું (Opening Base)

લિબ્રેઑફિસ બેઝ સામાન્ય રીતે ઉબુન્ટુ લિનક્સમાં મૂળભૂત રીતે ઈન્સ્ટોલ હોતું નથી, કારણ કે તે Java પર આધારિત છે. તેથી, આપણે તેનો ઉપયોગ શરૂ કરીએ તે પહેલાં, તેને ઇન્સ્ટોલ કરવું જરૂરી છે. ઉબુન્ટુ લિનક્સમાં તેને ઇન્સ્ટોલ કરવાની સૌથી સરળ રીતોમાંની એક છે ટર્મિનલ વિન્ડો માં "sudo apt install libreoffice-base" ક્માન્ડ રન કરવો. અથવા, તમે લિબ્રેઑફિસની સત્તાવાર વેબસાઇટ પરથી પણ તેને ડાઉનલોડ કરી શકો છો. એકવાર ઇન્સ્ટોલ થઈ ગયા પછી, આપણે તેનો ઉપયોગ શરૂ કરી શકીએ છીએ. લિબ્રેઑફિસ બેઝને તમે ઘણી રીતે ખોલી શકો છો. સૌથી સરળ રીત એ છે કે લિબ્રેઑફિસ બેઝ એપ્લિકેશન ને લૉન્ચર પર પિન કરી દો અને તેના પર ક્લિક કરો. આમ કરવાથી, આકૃતિ 2.1માં બતાવ્યા મુજબ, “ડેટાબેઝ વિઝાર્ડ” નામનો એક ડાયલોગ બોક્સ ખૂલશે. વિઝાર્ડ એ એક સ્ટેપ-બય-સ્ટેપ ગ્રાફિકલ ઇન્ટરફેસ ટૂલ છે, જે યુઝર્સને માહિતી પૂછીને જટિલ કાર્યોને સરળ બનાવે છે અને તે કાર્ય કરવાની પ્રક્રિયાને આપમેળે પાર પાડે છે.



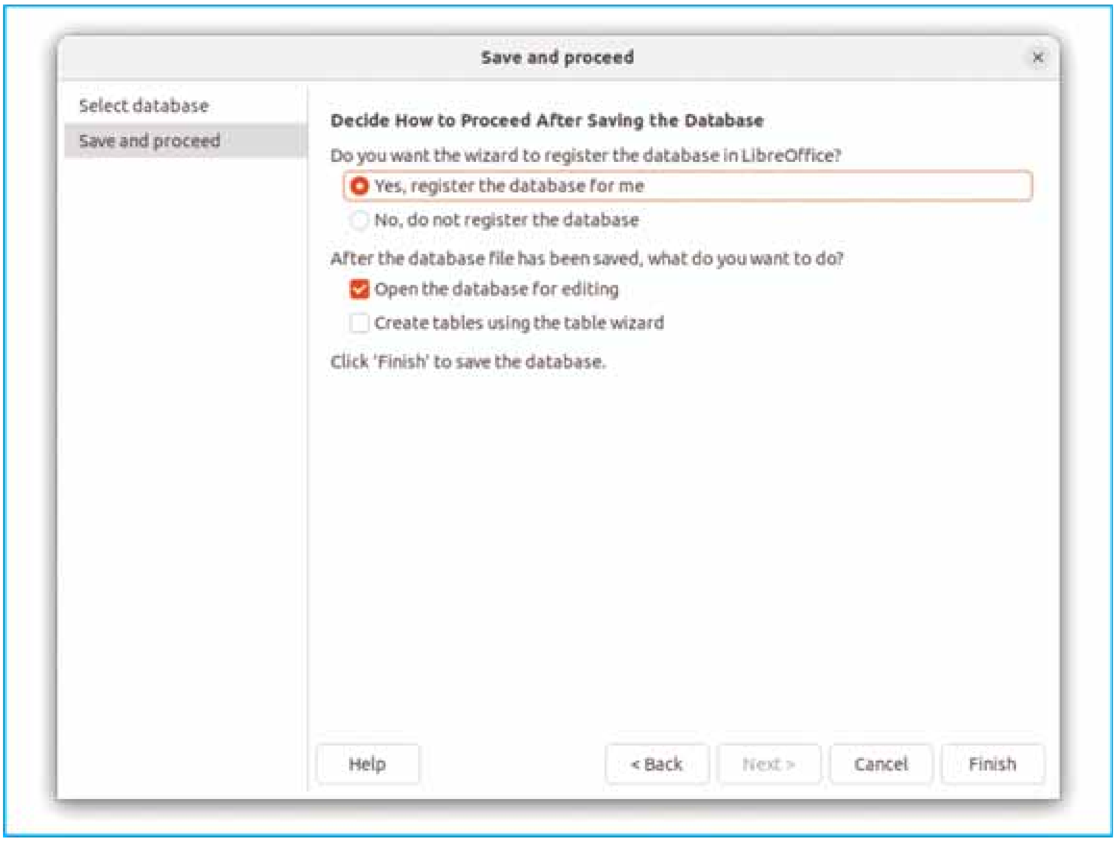
આકૃતિ 2.1 : ડેટાબેઝ વિઝાર્ડની શરૂઆતની સ્ક્રીન

તમે આકૃતિ 2.1માં જોઈ શકો છો કે તે સ્ક્રીન આપણને ત્રણ વિકલ્પો આપે છે, જેમાંથી આપણે કોઈ એક પસંદ કરી શકીએ છીએ: *Create a new database, Open an existing database file, Connect to an existing database.* આપણે આમાંથી કોઈ એક વિકલ્પ પસંદ કરવો પડશે. સરળતાથી સમજી શકાય તે માટે, વિકલ્પ એકનો

ઉપયોગ કરીને આપણે એક નવો ડેટાબેઝ બનાવી શકીએ છીએ. અન્ય વિકલ્પો આપણને એવા ડેટાબેઝ સાથે કામ કરવાની પરવાનગી આપે છે જે પહેલાંથી જ આપણા કમ્પ્યૂટરમાં બનાવવામાં અને સંગ્રહ કરવામાં આવ્યા છે.

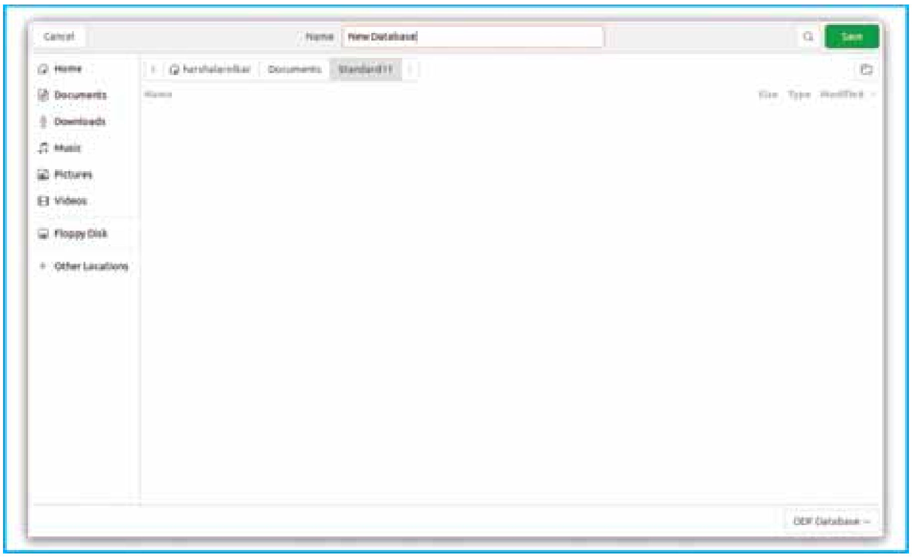
નોંધ : આકૃતિ 2.1માં દેખાતી સ્ક્રીન પરની વિગતો તમારા કમ્પ્યૂટરમાના ઈન્સ્ટોલેશન પ્રમાણે અલગ-અલગ હોઈ શકે છે.

ચાલો, આપણે પ્રકરણ-1 માં સ્કૂલ મેનેજમેન્ટ સીસ્ટમ માટે જે ટેબલ તૈયાર કર્યા હતા તેના માટે એક ડેટાબેઝ બનાવવાનો પ્રયાસ કરીએ. આ નવો ડેટાબેઝ બનાવવા માટે, *"Create a new database"* વિકલ્પને સિલેક્ટ કરો અને પછી *"Next"* બટન પર ક્લિક કરો.આમ કરવાથી, આકૃતિ 2.2માં બતાવેલ સ્ક્રીન તમારી સામે દેખાશે.



આકૃતિ 2.2 : ડેટાબેઝ વિઝાર્ડની સ્ક્રીન

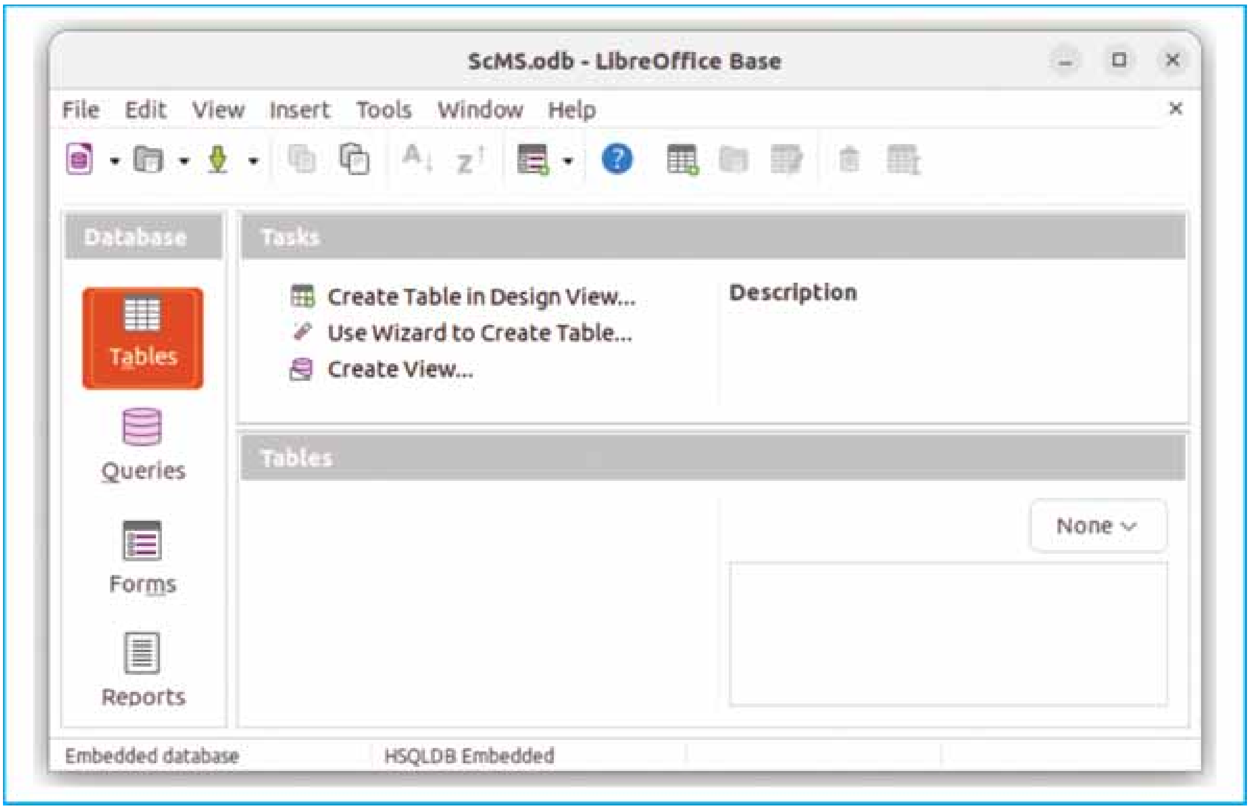
અહીં આપણે બે કામ કરવાના છે તે ધ્યાનમાં લો. પહેલું કામ, આપણે આપણી ડેટાબેઝ લિબ્રેઓફિસમાં રજિસ્ટર કરવી છે કે નહીં? તે નક્કી કરવું. મૂળભૂત રીતે, *Yes, register the database for me* એ વિકલ્પ પસંદ કરેલો છે. જો રજિસ્ટર કરવામાં આવે, તો આ ડેટાબેઝ લિબ્રેઓફિસના બીજા એપ્લિકેશન જેવા કે કેલ્ક અને રાઈટરમાં પણ



આકૃતિ 2.3 ડેટાબેઝ સેવ કરવાની સ્ક્રીન

વાપરી શકાશે. બીજું, એકવાર આપણો ડેટાબેઝ સંગ્રહ કર્યા પછી આપણે તેના પર કામ કઈ રીતે શરૂ કરવું છે તે નક્કી કરવું. તમે જોઈ શકો છો કે, મૂળભૂત રીતે, *Open the database for editing* એ વિકલ્પ પસંદ કરેલો છે. જો આપણે આ બંને ક્રિયાઓ કરવી હોય, તો આપણે *Create tables using the table wizard* એ વિકલ્પ પણ પસંદ કરી શકીએ છીએ. હાલમાં, આપણે પહેલા ફક્ત ડેટાબેઝ બનાવીશુ. તે માટે *Finish* બટન પર ક્લિક કરો. આમ કરવાથી આકૃતિ 2.3માં દર્શાવ્યા પ્રમાણેની સ્ક્રીન ખુલશે.

સૌ પ્રથમ, ડેટાબેઝની ફાઈલ સંગ્રહ કરવા માટે એક યોગ્ય ફોલ્ડર પસંદ કરો. ધ્યાન આપો કે આપણે આ ડેટાબેઝ કમ્પ્યૂટરના Documents નામના ફોલ્ડરની અંદર આવેલા Standard11 નામના ફોલ્ડરમાં સેવ કરવાનો પ્રયત્ન કરી રહ્યા છીએ. *Name* લેબલવાળા ટેક્સ્ટબોક્સમાં *New Database* ને બદલે ScMS લખો. નોંધ લો કે ડીફોલ્ટ રીતે *ODF (Open Document Format) Database* સિલેક્ટ કરેલ છે. તેથી, લિબ્રેઓફિસ બેઝ આપણે જે ડેટાબેઝની ફાઈલ બનાવવા માંગીએ છીએ તેને આપોઆપ *.odb* એક્સટેન્શન આપી દેશે. છેલ્લે, *Save* બટન પર ક્લિક કરો. હવે આ ડેટાબેઝ સંગ્રહ થઈ અને આગળના ઉપયોગ માટે તૈયાર થઈ જશે. આ પછી, તમને આકૃતિ 2.4માં દર્શાવ્યા પ્રમાણેની સ્ક્રીન જોવા મળશે.

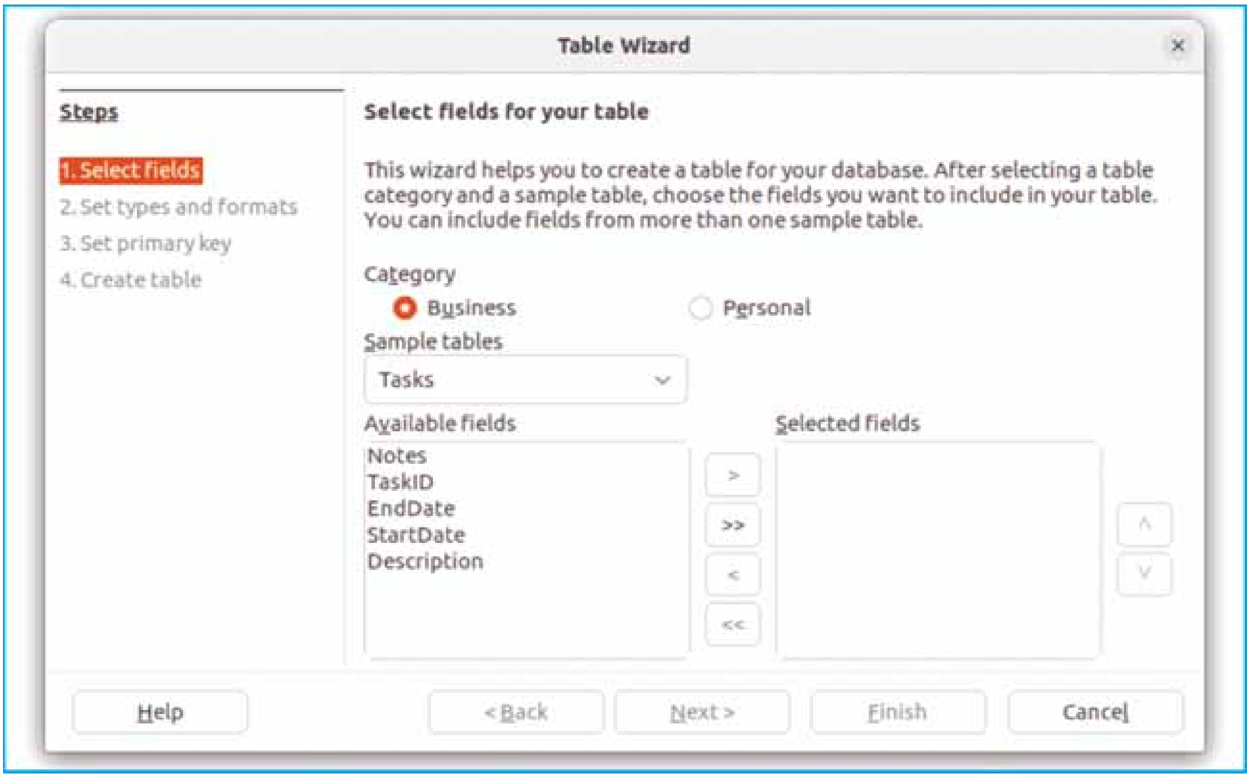


આકૃતિ 2.4 : ScMS ડેટાબેઝની વિન્ડો

ડાબી બાજુના ભાગમાં, આપણે ડેટાબેઝના જુદા જુદા ઓબ્જેક્ટ, જેવા કે ટેબલ, ક્વેરી, ફોર્મ અને રિપોર્ટ જોઈ શકીએ છીએ. ડીફોલ્ટ રીતે, ટેબલ ઓબ્જેક્ટ સિલેક્ટ કરેલો દેખાશે. હવે આપણે ScMS ડેટાબેઝના જુદા જુદા ઓબ્જેક્ટ સાથે કામ કરવા માટે તૈયાર છીએ.

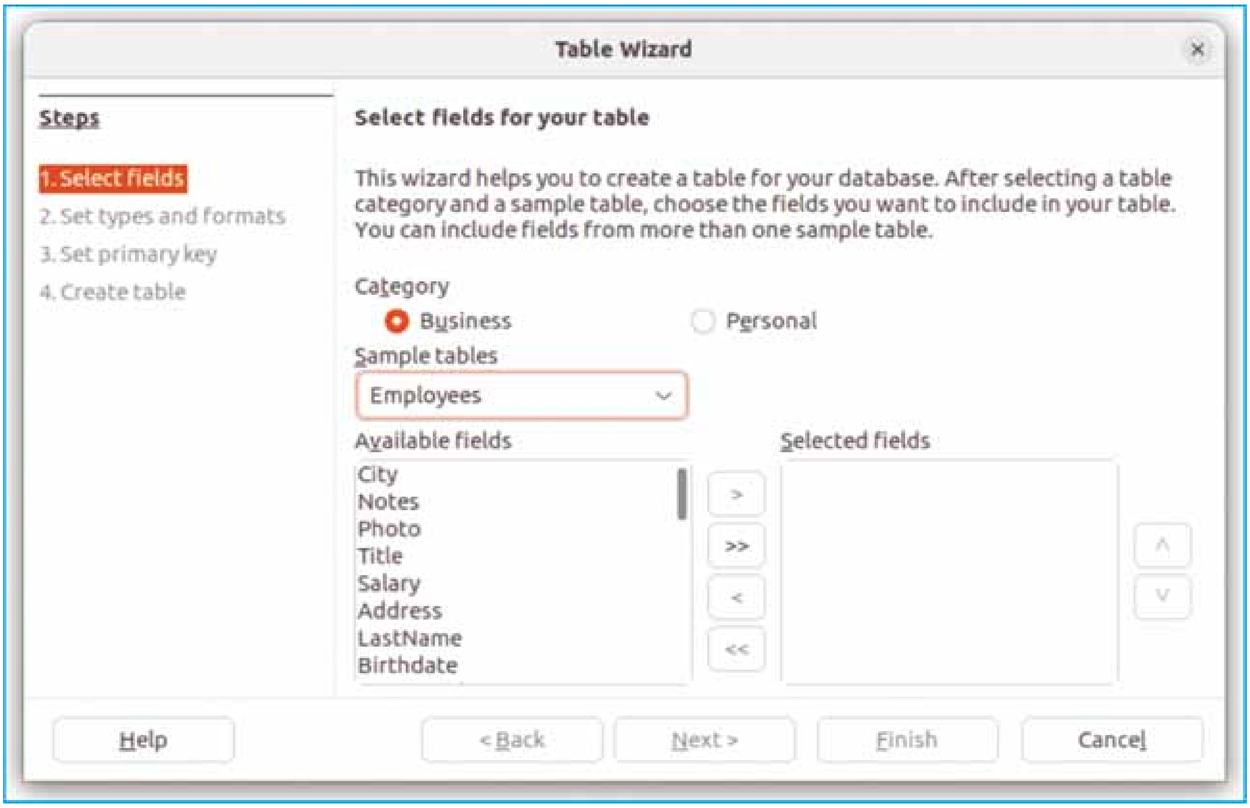
વિઝાર્ડનો ઉપયોગ કરીને ટેબલ બનાવવા (Creating Tables using Wizard)

ડેટાબેઝમાં ટેબલ વગર કોઈ અર્થપૂર્ણ માહિતી સ્ટોર કરી શકાતી નથી, તેથી પહેલા આપણે ટેબલ બનાવવાનું શીખીશું. ટેબલ બનાવવા માટે આપણે વિઝાર્ડનો ઉપયોગ કરીશું. આકૃતિ 2.4માં બતાવ્યા મુજબ ટાસ્ક પેનમાં બતાવેલ બીજા ઓપ્શન એટલે કે *Use Wizard to Create Table...* પર ક્લિક કરો. આનાથી આકૃતિ 2.5માં બતાવ્યા મુજબ એક ટેબલ વિઝાર્ડ ડાયલોગ બોક્સ દેખાશે.



આકૃતિ 2.5 : ટેબલ વિઝાર્ડ ડાયલોગ બોક્સ

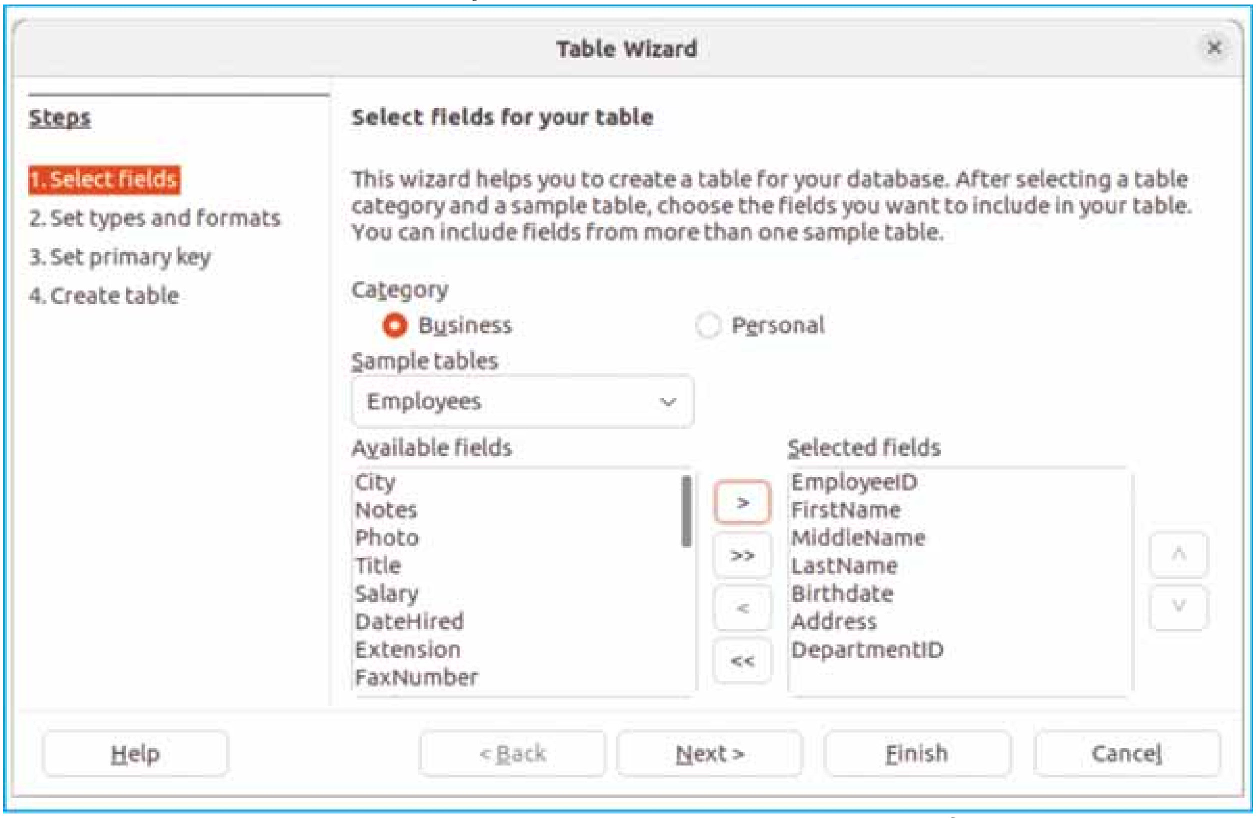
Table Wizard બિઝનેસ અને પર્સનલ એમ બે કેટેગરીના ટેમ્પ્લેટ્સ આપે છે. બિઝનેસ કેટેગરીના ટેબલમાં ટાસ્ક, એસેટ્સ, ઇવેન્ટસ, ઓર્ડર્સ અને અન્ય જેવી એન્ટિટી માટેના ટેમ્પ્લેટ્સ હોય છે. જ્યારે પર્સનલ કેટેગરીના ટેબલમાં પ્લાન્ટ્સ, ઓથર્સ, લાઈબ્રેરીઝ, રેસીપીઝ અને અન્ય જેવી એન્ટિટી માટેના ટેમ્પ્લેટ્સ હોય છે.



આકૃતિ 2.6 : Employees ટેબલ બનાવવા માટેનું ટેબલ વિઝાર્ડ ડાયલોગ બોક્સ

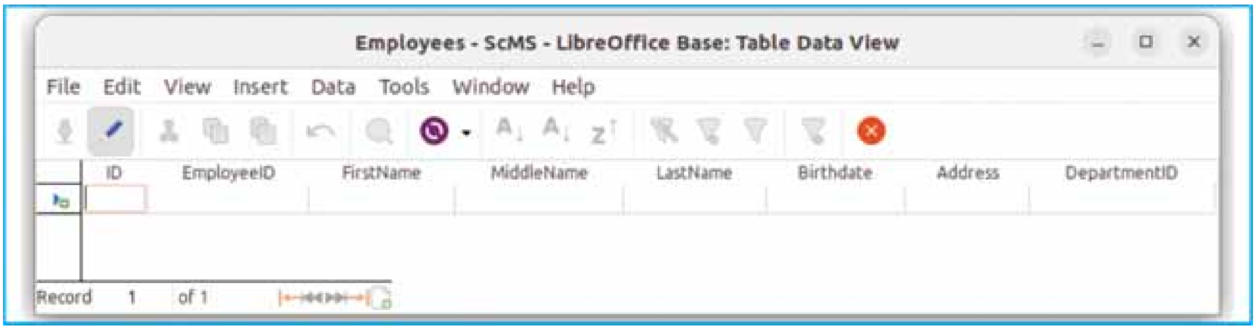
આપણે આકૃતિ 2.5ના *Category* લેબલ હેઠળનો *Business* ઓપ્શન પસંદ કરીશું. ત્યારબાદ *Sample tables* લેબલ હેઠળ દેખાતા ડ્રોપ-ડાઉન લિસ્ટ પર ક્લિક કરો. સમજણ માટે, ચાલો આપણે Employee ટેબલ બનાવીએ. ધ્યાન રાખો કે ડ્રોપ-ડાઉન લિસ્ટમાં *Employees* નામનું એક ટેબલ છે. તેને સિલેક્ટ કરો એટલે આપણને આકૃતિ 2.6માં બતાવ્યા મુજબ ફીલ્ડનું લીસ્ટ દેખાશે.

તમામ ફીલ્ડ્સને આપણા ટેબલનો ભાગ બનાવવા માટે >> બટન પર ક્લિક કરો. વૈકલ્પિક રીતે, આપણે કંટ્રોલ કી દબાવી રાખીને અને ઇચ્છિત ફીલ્ડને એક પછી એક સિલેક્ટ કરીને એક ફીલ્ડ અથવા ફીલ્ડના સમૂહને પસંદ કરી શકીએ છીએ. ચાલો આપણે EmployeeID, FirstName, MiddleName, LastName, Birthdate, Address અને DepartmentID ફીલ્ડને સિલેક્ટ કરીએ. એકવાર આપણે ઇચ્છિત ફીલ્ડને સિલેક્ટ કરી લીધા પછી, તેને આપણા ટેબલનો ભાગ બનાવવા માટે > બટન પર ક્લિક કરો. આકૃતિ 2.7માં બતાવ્યા મુજબ, સિલેક્ટ કરેલા તમામ ફીલ્ડ હવે *Selected fields* લિસ્ટ બોક્સ માં દેખાશે.



આકૃતિ 2.7 : Employees ટેબલના ફીલ્ડ પસંદ કરવા માટેનું ટેબલ વિઝાર્ડ ડાયલોગ બોક્સ

આપણે [∧] અને [∨] બટનનો ઉપયોગ કરીને ફીલ્ડના ક્રમને આપણી જરૂરિયાત મુજબ ફરીથી ગોઠવી શકીએ છીએ. એકવાર બધા ફીલ્ડને આપણી જરૂરિયાત મુજબ ગોઠવવામાં આવે, પછી *Finish* બટન પર ક્લિક કરો. Employees ટેબલ આકૃતિ 2.8માં બતાવ્યા મુજબ *Table Data View* તરીકે ઓળખાતી નવી વિન્ડોમાં ખુલશે. આ *Table Data View* યુઝરને ટેબલમાં રેકોર્ડ એન્ટર કરવાની મંજૂરી આપે છે.



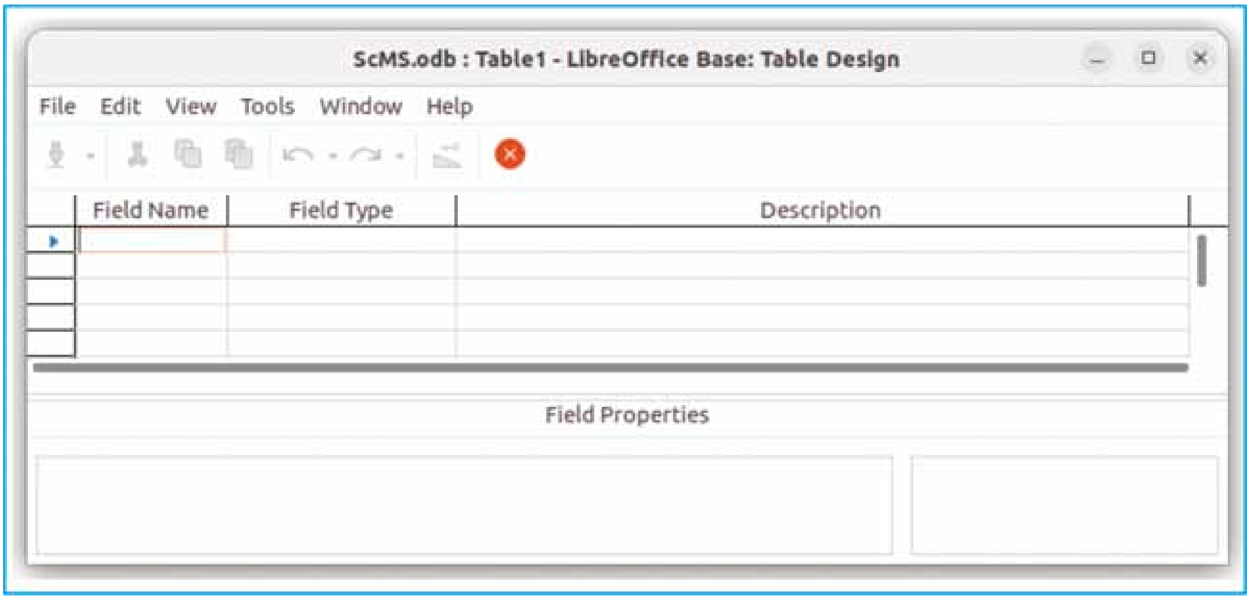
આકૃતિ 2.8 : Employees ટેબલનો ટેબલ ડેટા વ્યૂ

આપણે ટેબલમાં ડેટા એન્ટર કરવાનું શરૂ કરી શકીએ છીએ અને એકવાર ડેટા એન્ટ્રી પૂર્ણ થઈ જાય, પછી આપણે ડેટા વ્યૂ બંધ કરી શકીએ છીએ. જ્યારે આપણે તેને બંધ કરીશું, ત્યારે આપણને પાછા ScMS ડેટાબેઝ વિન્ડો પર લઈ જવામાં આવશે.

ડિઝાઈન વ્યૂનો ઉપયોગ કરીને ટેબલ બનાવવા (Creating Tables using Design View)

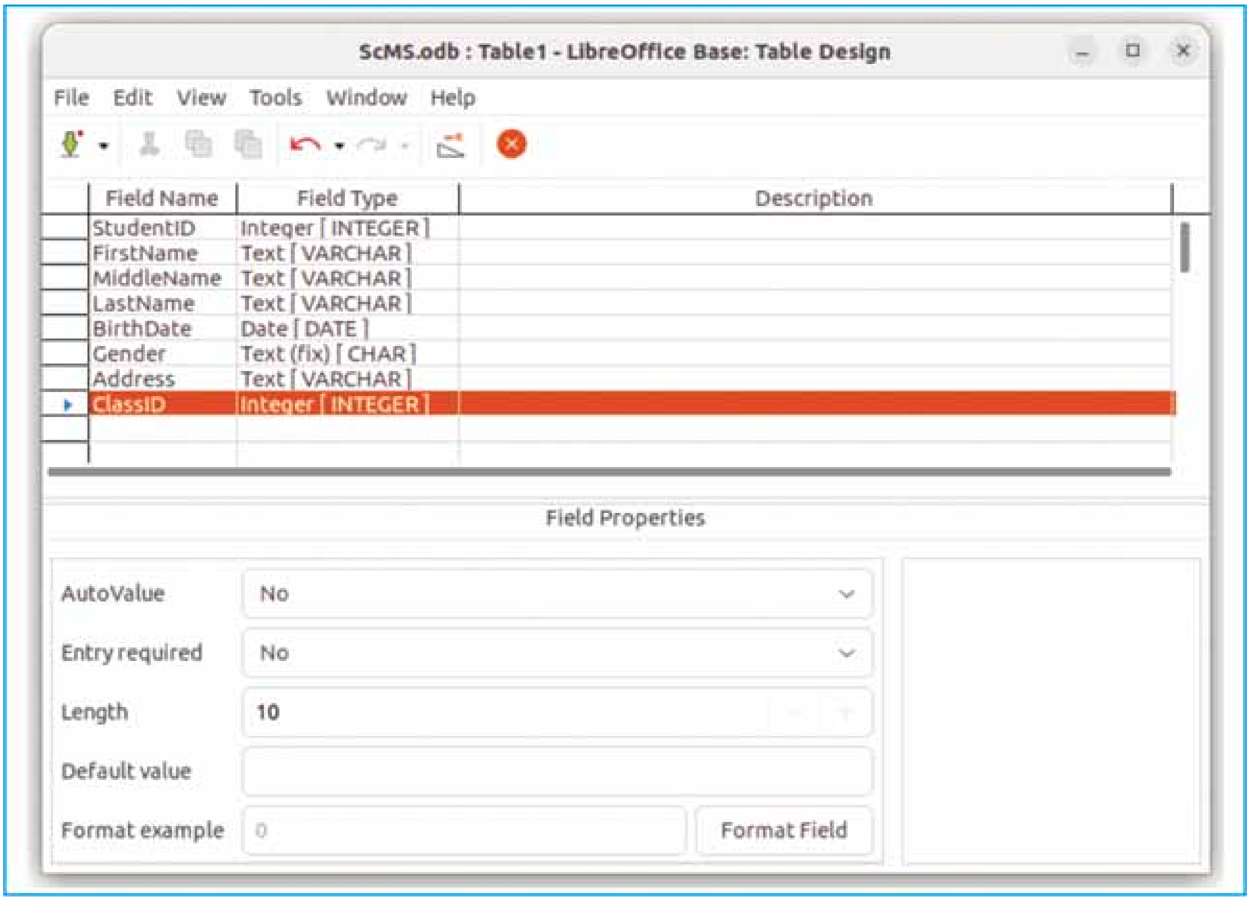
વિઝાર્ડનો ઉપયોગ કરીને બનાવેલા ટેબલમાં ઘણીવાર યુઝરની જરૂરિયાત મુજબ ફેરફાર કરવાની જરૂર પડે છે. જરૂરિયાત મુજબના ટેબલ બનાવવા માટે લિબ્રેઓફિસ બેઝ ડિઝાઈન વ્યૂનો વિકલ્પ આપે છે. ટેબલને ડીઝાઈન વ્યૂમાં બનાવવામાં આવે અથવા ઓપન કરવામાં આવે ત્યારે યુઝર તેમની જરૂરિયાત મુજબ ટેબલના ફીલ્ડને ક્રિએટ, એડિટ, અપડેટ અથવા ડિલીટ કરી શકે છે. ટેબલ ડિઝાઈન વ્યૂ આપણને જરૂરી ફીલ્ડ નેમ બનાવવાની, ફીલ્ડમાં સ્ટોર કરી શકાય તેવા ડેટાનો પ્રકાર સિલેક્ટ કરવાની, જો જરૂર હોય તો યુઝર્સને મદદ કરવા માટે ફીલ્ડનું વર્ણન ઉમેરવાની અને ફીલ્ડમાં સ્ટોર કરી શકાય તેવા ડેટાને કંટ્રોલ અને વેલિડેટ કરવાની પણ મંજૂરી આપે છે.

આપણે પ્રકરણ-1 માં ડિઝાઈન કર્યા મુજબ, ચાલો હવે આપણે Student ટેબલ બનાવીએ, જેમાં StudentID, FirstName, MiddleName, LastName, BirthDate, Gender, Address અને ClassID જેવા એટ્રિબ્યુટ્સ છે. આ ટેબલ બનાવવા માટે આકૃતિ 2.4ના *Create Table in Design View...* ઓપ્શન પર ક્લિક કરો. આનાથી આકૃતિ 2.9માં બતાવ્યા મુજબ એક ખાલી ટેબલ ડિઝાઈન વ્યૂ ખુલશે.



આકૃતિ 2.9 : ખાલી ટેબલ ડિઝાઈન વ્યૂ

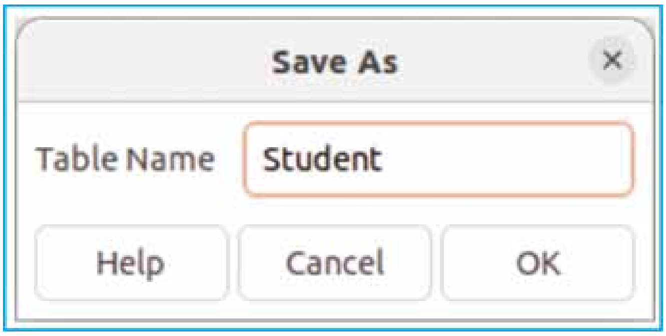
તમે જોઈ શકો છો કે, અહીં આપણને ત્રણ કોલમ સાથેની ડેટા ગ્રીડ પૂરી પાડવામાં આવી છેઃ *Field Name, Field Type* and *Description*. ટેબલ ગ્રીડની નીચે *Field Properties* પેન પણ ડિસપ્લે થશે. આપણે હવે *Field Name* કોલમ હેઠળ ફીલ્ડના નામ ટાઈપ કરી અને *Field Type* કોલમ હેઠળ જરૂરી ડેટા ટાઈપ સિલેક્ટ કરીને ટેબલ સ્ટ્રક્ચર બનાવવાનું શરૂ કરી શકીએ છીએ. આ તબક્કે આપણે ફીલ્ડનું વર્ણન ઉમેરીશું નહીં. Student ટેબલ માટે અગાઉ જણાવેલા તમામ ફીલ્ડ નામ એન્ટર કરો અને સંબંધિત ડેટા ટાઈપ સિલેક્ટ કરો. પ્રક્રિયા પૂર્ણ થયા પછી સ્ક્રીન આકૃતિ 2.10માં બતાવેલ છે તેવી દેખાશે.



આકૃતિ 2.10 : ફીલ્ડના નામ સાથે ટેબલ ડિઝાઈન વ્યૂ

ગ્રીન ડાઉન એરો સાથેના લાલ ટપકાં (*Save* બટન) પર ક્લિક કરીને ટેબલને સેવ કરો. આમ કરવાથી આકૃતિ 2.11માં બતાવ્યા મુજબ એક *Save As* ડાયલોગ બોક્સ ખુલશે.

ટેબલનું નામ (Student) ટાઈપ કરો અને *OK* બટન પર ક્લિક કરો. આકૃતિ 2.12માં બતાવ્યા મુજબ એક પોપઅપ દેખાશે.



આકૃતિ 2.11 : ટેબલ ને સેવ કરવું

આકૃતિ 2.12 : પ્રાઈમરી કી માટેનું એલર્ટ

આ સ્ક્રીન એક એલર્ટ મેસેજ બતાવે છે. જે દર્શાવે છે કે યુઝરને યુનિક પ્રાઈમરી કી સિલેક્ટ કરવાની જરૂર છે. જેમ આપણે પ્રકરણ-1 માં શીખ્યા તે પ્રમાણે, પ્રાઈમરી કી એ એક ફીલ્ડ છે જે ટેબલની એક રોને યુનિક રીતે ઓળખે છે. હાલમાં *No* બટન પર ક્લિક કરો. હવે ટેબલ સેવ થઈ ગયું છે અને આપણે તેના પર જરૂરી ઓપરેશન કરી શકીએ છીએ. ટેબલ બંધ કરો, ધ્યાન રાખો કે ScMS ડેટાબેઝ વિન્ડો હવે Employees અને Student એમ બે ટેબલનું લિસ્ટ બતાવશે.

ટેબલના સ્ટ્રક્ચરમાં ફેરફાર કરવા (Modifying the Structure of Table)

અત્યાર સુધી, આપણે લિબ્રેઓફીસ બેઝમાં ટેબલ બનાવવાના બે રસ્તા શીખ્યા. ઘણીવાર આપણે બનાવેલા ટેબલના સ્ટ્રક્ચરમાં ફેરફાર કરવાની જરૂર પડે છે. આપણે એક નવું ફીલ્ડ ઉમેરવાની, હાલના ફીલ્ડનું નામ બદલવાની, ડેટા ટાઈપ બદલવાની અથવા ફીલ્ડ પ્રોપર્ટીઝ ઉમેરવાની કે દૂર કરવાની જરૂર પડી શકે છે. અહીં એક એવું સૂચન કરવામાં આવે છે કે આ તમામ ઓપરેશન ટેબલમાં ડેટા એન્ટર કરતા પહેલા કરવા જોઈએ. ચાલો આપણે Employees ટેબલમાં ફેરફાર કરવાનો પ્રયત્ન કરીએ જેથી તેનો ઉપયોગ પ્રકરણ-1 માં આપણે ડીઝાઈન કરેલા Teacher ટેબલ તરીકે કરી શકાય.

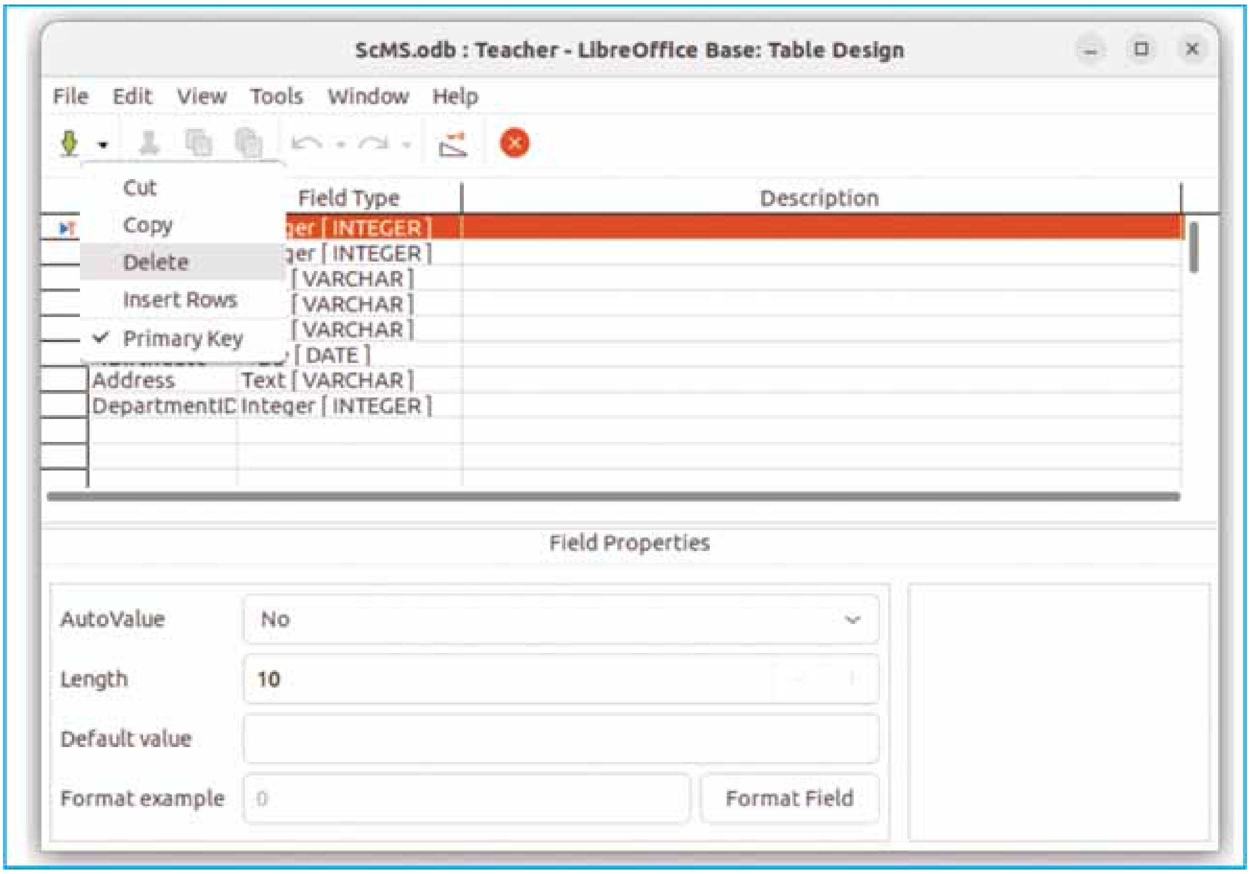
આકૃતિ 2.13 : રીનેમ ટૂ ડાયલોગ બોક્ષ

Teacher ટેબલમાં એટ્રીબ્યુટ – TeacherID, FirstName, MiddleName, LastName, BirthDate, Gender, Address અને SubjectID છે. Employees ટેબલ પણ કંઈક અંશે સમાન સ્ટ્રક્ચર ધરાવે છે. Employees ટેબલના સ્ટ્રક્ચરને બદલવા માટે, ScMS વિન્ડો પર જાઓ, Employees ટેબલને સિલેક્ટ કરો અને તેના પર રાઈટ ક્લિક કરો. ડ્રોપ ડાઉન મેનુ માંથી *Rename...* ઓપ્શન પસંદ કરો, આનાથી આકૃતિ 2.13માં બતાવ્યા પ્રમાણે એક *Rename* ડાયલોગ બોક્ષ ખુલશે.

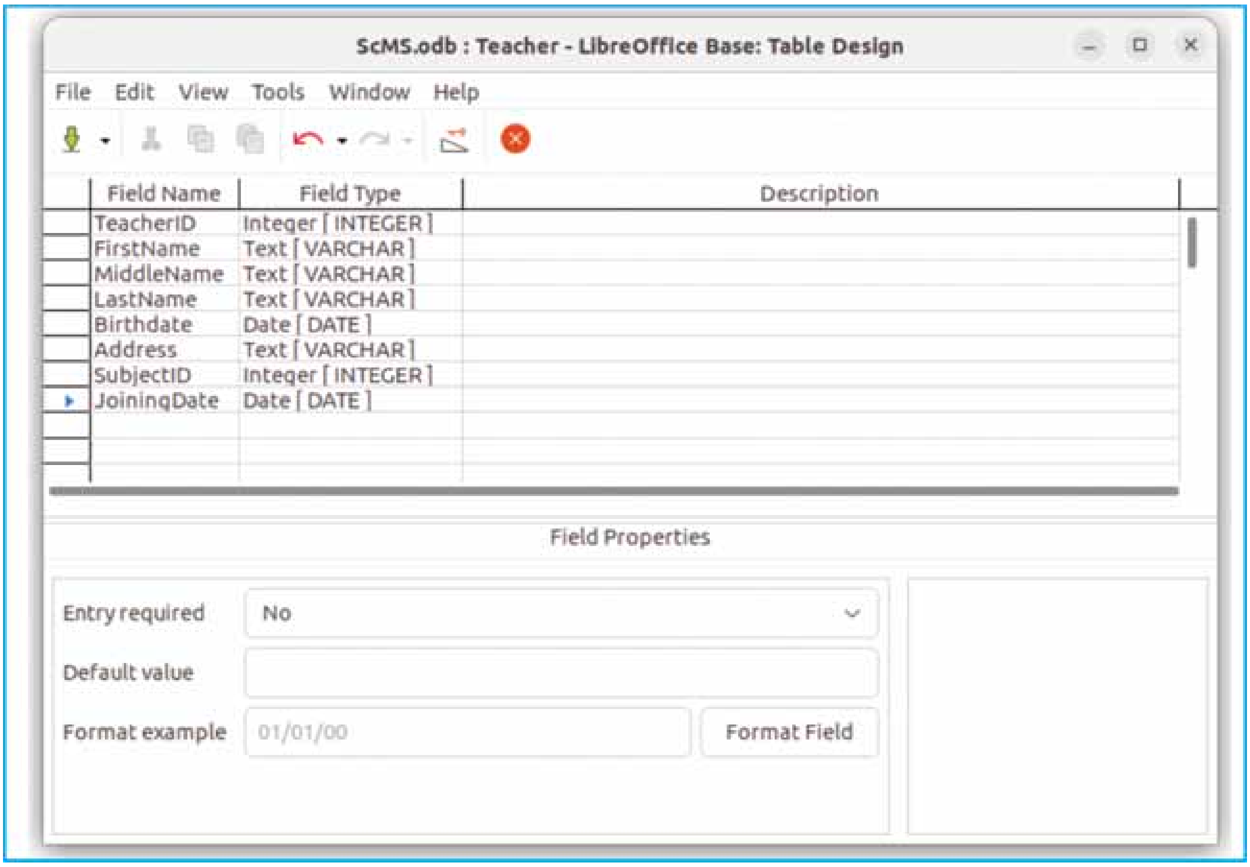
Employees માંથી નામ બદલીને Teacher કરો અને *OK* બટન પર ક્લિક કરો. હવે ટેબલનું નામ બદલાઈ જશે અને આપણે પાછા ScMS વિન્ડોમાં આવી જઈશું.

ફરીથી Teacher ટેબલ સિલેક્ટ કરો અને તેના પર રાઈટ ક્લિક કરો, ડ્રોપ ડાઉન મેનુમાંથી *Edit...* ઓપ્શન પસંદ કરો, આનાથી ટેબલ ડીઝાઈન વ્યૂમાં ખુલશે. જુઓ આપણને ID નામનું એક ફીલ્ડ દેખાય છે, જેની સાથે વાદળી ત્રિકોણનું આઈકોન અને તેની આગળ એક કી(ચાવી) છે. ત્રિકોણનું આઈકોન કરન્ટ રો પોઈન્ટર તરીકે ઓળખાય છે, જે દર્શાવે છે કે કઈ રો કરન્ટ રો છે. કી પર લેફ્ટ ક્લિક કરીને કરન્ટ રોને સિલેક્ટ કરો. રો હાઈલાઈટ થઈ જશે. ID ફીલ્ડ આપણે બનાવેલા ટેબલ ડિઝાઈનનો ભાગ ન હતો, તેથી આ ફીલ્ડને ડિલીટ કરવા માટે કરન્ટ રો આઈકોન પર રાઈટ ક્લિક કરો એટલે આકૃતિ 2.14માં બતાવ્યા પ્રમાણેની એક સ્ક્રીન દેખાશે.

આકૃતિ 2.14માં બતાવેલ મેનુ વિવિધ ઓપરેશન કરવા માટે વાપરી શકાય છે. ફીલ્ડને રીમુવ કરવા માટે આપણે મેનુમાંથી *Cut* અથવા *Delete* ઓપ્શનનો ઉપયોગ કરી શકીએ છીએ. *Insert Rows* ઓપ્શનનો ઉપયોગ કરીને નવી રો ઇન્સર્ટ કરી શકાય છે. *Copy* ઓપ્શનનો ઉપયોગ કરીને આપણે ફીલ્ડ નેમ અને તેના ડેટા ટાઈપની નકલ કરી શકીએ છીએ. છેલ્લે, *Primary Key* ઓપ્શન પસંદ કરીને આપણે પસંદ કરેલા ફીલ્ડને પ્રાઈમરી કી બનાવી શકીએ છીએ. જુઓ, સ્ક્રીન પરના પ્રાઈમરી કી ઓપ્શનની આગળ એક ✓ નિશાની છે. આ સૂચવે છે કે આપણે જે ફીલ્ડ પર કામ કરવાનો પ્રયાસ કરી રહ્યા છીએ તે હાલમાં ટેબલની પ્રાઈમરી કી છે.



આકૃતિ 2.14 : ટેબલ ના સ્ટ્રક્ચરમાં ફેરફાર કરવો



આકૃતિ 2.15 : ટેબલનું સ્ટ્રક્ચર અપડેટ કરવું

સૌપ્રથમ, ડ્રોપ ડાઉન મેનુમાંથી *Delete* ઓપ્શન પસંદ કરો અને ID ફીલ્ડ ડિલીટ થઈ જશે અને EmployeeID ફીલ્ડને કરન્ટ રેકોર્ડ તરીકે દર્શાવવામાં આવશે. ફીલ્ડ નેમને અપડેટ કરવા માટે ફક્ત નામ પર ડબલ ક્લીક કરો અને નવું નામ ટાઈપ કરો. આ રીતે, EmployeeID ફીલ્ડનું નામ બદલીને TeacherID અને DepartmentIDનું નામ બદલીને SubjectID કરો. ટેબલ સ્ટ્રક્ચરમાં થયેલો ફેરફાર બતાવવા માટે આપણે અહીં એક નવું ફીલ્ડ JoiningDate પણ ઉમેરીશું. આ ફેરફારને સેવ કરો, જો પ્રાઈમરી કી ડિલીટ કરવા બદલ કોઈ ચેતવણી મળે તો અત્યારે તેને અવગણો. હવે સ્ક્રીન આકૃતિ 2.15માં બતાવ્યા પ્રમાણે દેખાશે.

ડીઝાઈન વ્યૂનો ઉપયોગ કરીને પ્રકરણ-1 મુજબના બાકીના ટેબલ બનાવો એટલે હવે ડેટાબેઝમાં પાંચ ટેબલ થશે. હવે આપણે બનાવેલા ટેબલમાં ડેટા એન્ટર કરવા માટે તૈયાર છીએ પરંતુ તે કરીએ તે પહેલાં, ચાલો ટેબલ ડિઝાઈન વ્યૂમાં આપેલ Description અને Field Properties જોઈએ.

ડિસ્ક્રીપ્શન અને ફીલ્ડ પ્રોપર્ટીઝ સેટ કરવી (Seting Description and Field Properties)

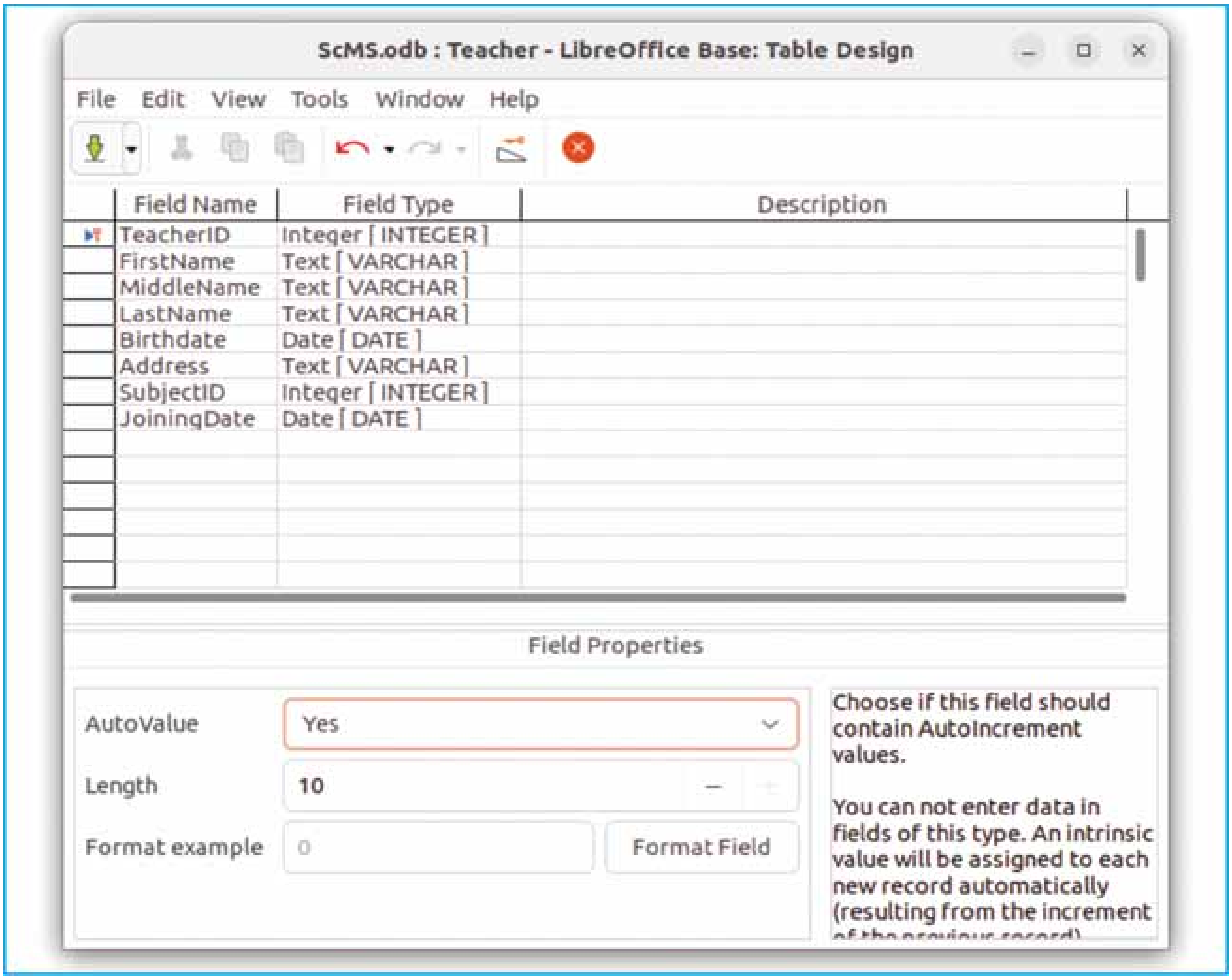
દરેક ફીલ્ડ માટે ડિસ્ક્રીપશન એન્ટર કરવું ફરજિયાત નથી, પરંતુ ડિસ્ક્રીપશન (વિવરણ) ઉમેરવું હંમેશા સારી પ્રથા છે કારણ કે તે ડોક્યુમેન્ટેશનમાં મદદ કરે છે. તે યુઝરને અથવા જે કોઈ ડેટાબેઝનો ઉપયોગ કરવાનો પ્રયાસ કરી રહ્યો હોય તેને દરેક ફીલ્ડનો હેતુ સમજવામાં મદદ કરે છે.

Field Properties ઓપ્શનનો ઉપયોગ કરવામાં આવે ત્યારે તે આપણને એન્ટર કરવાના ડેટાને નિયંત્રિત અને વેલીડેટ કરવાની મંજૂરી આપે છે. તે ફીલ્ડમાં વેલ્યૂ કેવી રીતે સ્ટોર અને ડિસ્પ્લે થાય છે એ પણ ઓળખે છે. ઉદાહરણ તરીકે, યુઝર BirthDate ફીલ્ડમાં કયા ફોર્મેટ જેવા કે DD-MM-YY અથવા MM-DD-YYમાં ડેટા એન્ટર કરશે તેમજ ડેટાને પ્રિન્ટ કરતી વખતે અથવા વ્યૂ કરતી વખતે અને અમાન્ય તારીખ એન્ટર કરવાના કિસ્સામાં યુઝરને સંદેશ આપતી વખતે તે કેવી રીતે ડિસ્પ્લે થશે તે આપણે નક્કી કરી શકીએ છીએ.

યુઝર જે ડેટાટાઈપ સિલેક્ટ કરે છે તેના સંબંધિત જુદા-જુદા ફીલ્ડ પ્રોપર્ટીઝનો એક સેટ ડિસપ્લે થાય છે. આપણી જરૂરિયાત મુજબ તમામ ફીલ્ડ પ્રોપર્ટીઝને આપણે બદલી શકીએ છીએ. આ વિભાગમાં કેટલીક સામાન્ય ફીલ્ડ પ્રોપર્ટીઝની ચર્ચા કરવામાં આવી છે.

ડિફોલ્ટ વેલ્યૂ (Default value) : આપણે એક વેલ્યૂ સ્પષ્ટ કરી શકીએ છીએ જે ફીલ્ડમાં ડિફોલ્ટ રૂપે સ્ટોર થશે. એકવાર આપણે કોઈપણ ફીલ્ડ માટે આ પ્રોપર્ટી સેટ કરી દઈએ, પછી જ્યારે આપણે ટેબલમાં નવો રેકોર્ડ એડ કરીશું, ત્યારે સેટ કરેલ ડીફોલ્ડ વેલ્યૂ આપોઆપ આવી જશે. ડેટા એન્ટ્રીના સમયે જો જરૂર હોય તો યુઝર વેલ્યૂ બદલી શકે છે.

ઓટો વેલ્યૂ (AutoValue) : આ પ્રોપર્ટીનો ઉપયોગ ન્યુમરીક પ્રકારના ફીલ્ડસ સાથે થાય છે, સામાન્ય રીતે તે ફીલ્ડ જે પ્રાઈમરી કી તરીકે હોય. ઉદાહરણ તરીકે, Teacher ટેબલમાં TeacherID પ્રાઈમરી કી તરીકે સેટ કરેલ છે અને તેની ડેટા ટાઈપ ઈન્ટીજર છે. આપણે અપેક્ષા રાખીએ છીએ કે TeacherID ફીલ્ડમાં 1,2,3 અને તે પ્રમાણેની વેલ્યૂ સ્ટોર થાય. આમ, આપણે આ ફીલ્ડ માટે ઓટૉ વેલ્યૂ (AutoValue) પ્રોપર્ટીને સક્ષમ (enable) કરી શકીએ છીએ. ચાલો Teacher ટેબલમાં TeacherID માટે ઓટૉ વેલ્યૂ (AutoValue) પ્રોપર્ટી સેટ કરીએ. Teacher ટેબલને ડીઝાઈન વ્યૂમાં ખોલો, TeacherID સિલેક્ટ કરો. આપણને આકૃતિ 2.16 માં બતાવ્યા પ્રમાણે *Field Properties* દેખાશે. *AutoValue* લેબલની બાજુમાં દેખાતા ડ્રોપ ડાઉન મેનુમાંથી "Yes" સિલેક્ટ કરો.

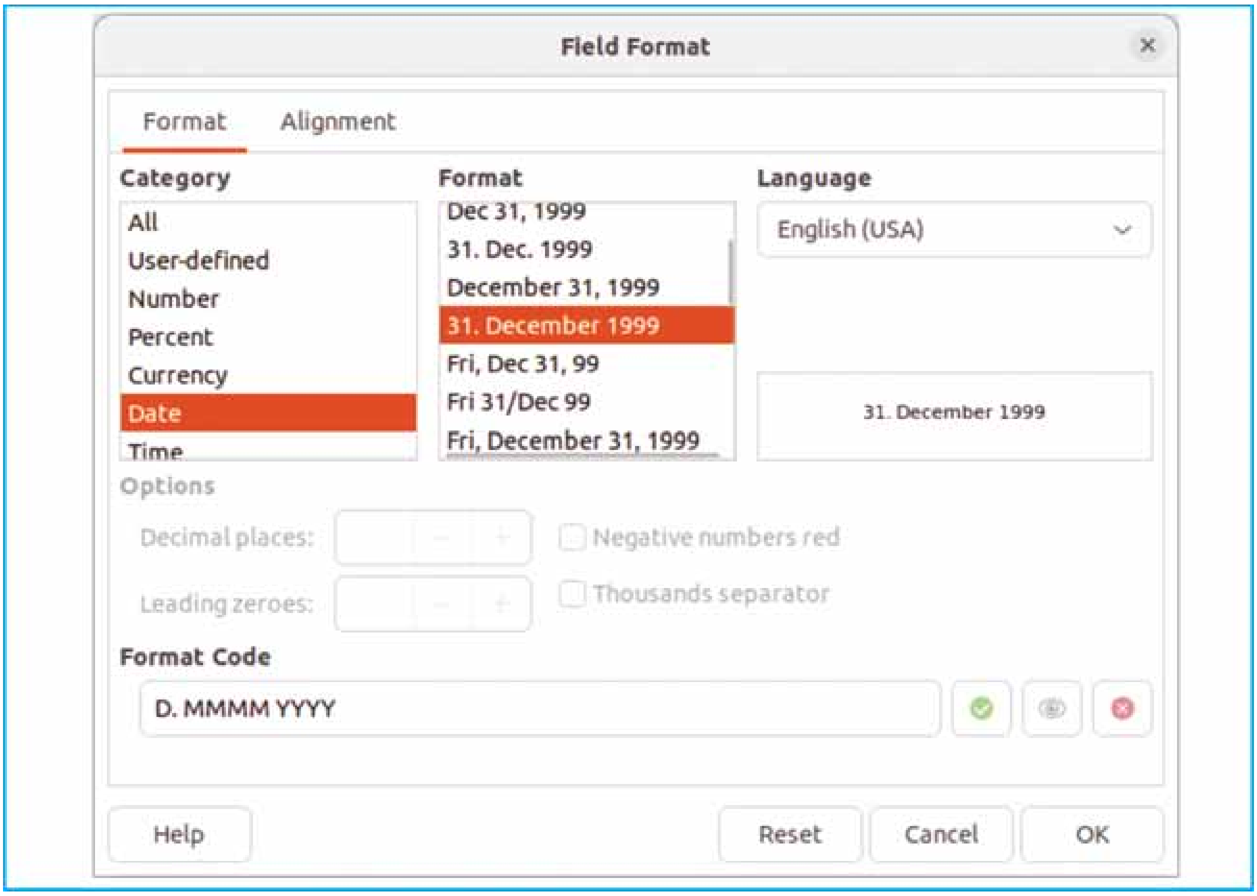


આકૃતિ 2.16 : ફીલ્ડની પ્રોપર્ટીઝ સેટ કરવી

લેન્થ (Length) : લેંથ પ્રોપર્ટી ટેક્સ્ટ ડેટા ટાઈપ સાથે સંકળાયેલ છે. Text[VARCHAR] ડેટાટાઈપના કિસ્સામાં તે આપોઆપ એક વેલ્યૂ, ઉદાહરણ તરીકે 50, અસાઈન કરે છે. આપણે વિવિધ પ્રકારના ડેટા સ્ટોર કરવાની આપણી જરૂરિયાત મુજબ લેંથ પ્રોપર્ટીની સાઇઝ અસાઈન કરી શકીએ છીએ. લિબ્રેઓફીસ બેઝ આપોઆપ Numeric, Date/Time, Yes/No અને Memo જેવા વિવિધ ડેટાટાઈપને પ્રી-ડિફાઈન્ડ લેંથ સાઈઝ આપે છે, આવા કિસ્સાઓમાં આ પ્રોપર્ટી સ્ક્રીન પર ડીસેબલ થઈ જશે. આકૃતિ 2.16માં લેંથ માટે વેલ્યૂ 10 આપેલ છે. અહીં રહેલું ફીલ્ડ ન્યુમરીક હોવાથી, આપણે તેને અપડેટ કરી શકીશું નહીં.

એન્ટ્રી રીકવાયર્ડ (Entry Required) : કેટલીકવાર આપણે ટેબલમાં ઉપયોગ લઈએ છીએ તેવા કેટલાક ફીલ્ડમાં ડેટાની આવશ્યકતા વૈકલ્પિક હોય છે અથવા ફરજિયાત હોય છે. એન્ટ્રી રીક્વાઈર્ડ પ્રોપર્ટી આપણને આવા ફીલ્ડનું સંચાલન કરવાની મંજૂરી આપે છે. ડેટા ખાલી ન રહે તેની ખાતરી કરવા માટે ફીલ્ડ પ્રોપર્ટીને "Yes" સેટ કરવાની રહે છે. સામાન્ય રીતે, ડીફોલ્ટ સીલેક્શન "No" હોય છે. Teacher ટેબલના કિસ્સામાં, FirstName, MiddleName અને LastName ફીલ્ડ્સ માટે આ પ્રોપર્ટીને "Yes" સેટ કરવાની જરૂર પડી શકે છે.

ફોર્મેટ (Format) : આ પ્રોપર્ટી ટેક્સ્ટ, નંબર અથવા ડેટના ફોર્મેટને સ્પષ્ટ કરે છે, જે એન્ટ્રી, ડિસપ્લે અને પ્રિન્ટ કરતી વખતે વપરાય છે. ફોર્મેટ પ્રોપર્ટી વિવિધ ડેટા ટાઈપ માટે જુદા જુદા સેટિંગ્સનો ઉપયોગ કરે છે. લિબ્રેઓફીસ બેઝ Number, Date/Time અને Yes/No ડેટા ટાઈપ માટે કેટલાક પ્રિ-ડિફાઈન્ડ ફોર્મેટ આપે છે. ચાલો ટીચર ટેબલના JoiningDate ફીલ્ડ માટે ફોર્મેટ સેટ કરીએ. ટેબલને ડીઝાઈન વ્યૂમાં ખોલો, *JoiningDate* ફીલ્ડ સિલેક્ટ કરો, આકૃતિ 2.16માં દેખાતા *Format Field* બટન પર ક્લીક કરો એટલે આકૃતિ 2.17માં બતાવ્યા પ્રમાણે *Field Format* ડાઈલોગ બોક્ષ ખુલશે.



આકૃતિ 2.17 : ફીલ્ડ ફોર્મેટ ડાયલોગ બોક્ષ

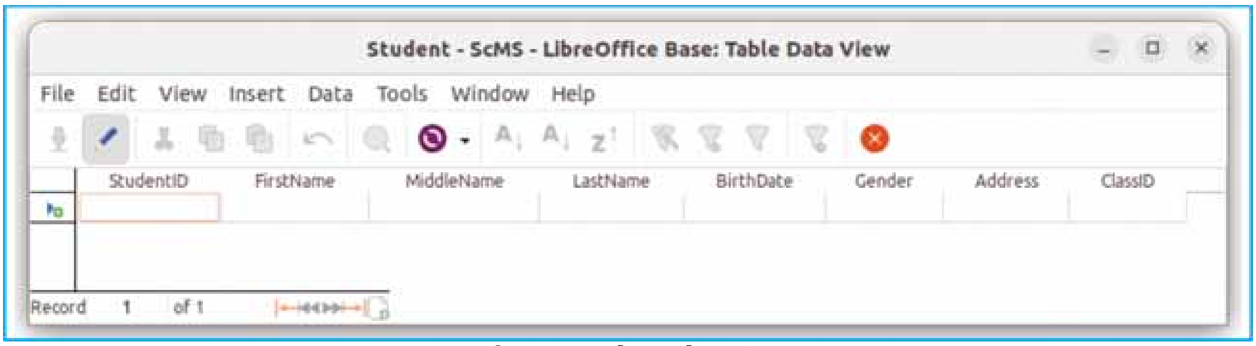
જુઓ, *Category* લેબલ હેઠળ Date સિલેક્ટ કરેલ છે, *Format* લેબલ હેઠળ *31.December 1999* ઓપ્શન સિલેક્ટ કરો. આ ફોર્મેટ લાગુ કરવા માટે *OK* બટન પર ક્લિક કરો, હવે લિબ્રેઓફીસ બેઝ એન્ટર કરેલી કોઈપણ તારીખને આપોઆપ D.MMMM YYYY ફોર્મેટ માં રૂપાંતરિત કરશે.

ટેબલ ડેટા પર ઓપરેશન (Operations on Table Data)

એકવાર ટેબલ સ્ટ્રક્ચર બનાવી તેને આખરી સ્વરૂપ આપવામાં આવે ત્યારપછીનું પગલું એ ટેબલમાં જરૂરી ડેટા ઈન્સર્ટ કરવાનું છે. ચાલો ટેબલ પર કરી શકાય તેવા વિવિધ ઓપરેશન જોઈએ.

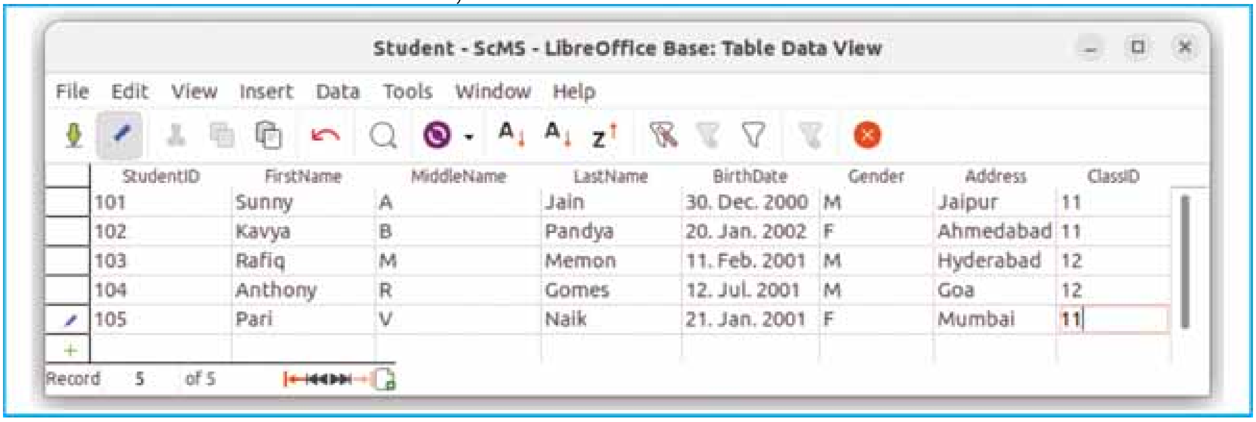
ડેટાને ઈન્સર્ટ કરવા (Insert Data)

ટેબલમાં રેકોર્ડ ઈન્સર્ટ કરવા માટે, પહેલા આપણે જરૂરી ટેબલ ખોલવાની જરૂર છે. ટેબલ ખોલવા માટે ScMS ડેટાબેઝ વિન્ડોમાં ટેબલ પેન પર જાઓ અને ટેબલના નામ પર ડબલ ક્લિક કરો અથવા ઇચ્છિત ટેબલ પર રાઈટ ક્લિક કરો અને મેનુ ઓપ્શનમાંથી *Open...* પર ક્લિક કરો. આમ કરવાથી સિલેક્ટ કરેલ ટેબલ આકૃતિ 2.18માં બતાવ્યા પ્રમાણે ડેટાશીટ વ્યૂમાં ખુલશે. નોંધ લો કે જો પ્રાઈમરી કી સેટ કરેલી ન હોય તો આપણે ડેટા એન્ટર કરી શકીશું નહીં.



આકૃતિ 2.18 : ટેબલ ડેટાશીટ વ્યૂ

આકૃતિ 2.18 માં જુઓ, ફીલ્ડ નેમને ટાઇટલ લાઇન તરીકે ઓળખાતી આડી લીટીમાં બતાવવામાં આવ્યા છે. ટાઇટલ લાઇનની નીચે ખાલી બોક્સની એક રો છે. StudentID ફીલ્ડ હેઠળનું ટેક્સ્ટ બોક્સ સિલેક્ટ કરેલું છે અને તેની આગળ લીલા રંગના + સાઈન સાથે વાદળી એરો છે. વાદળી એરોને Record Selector આઇકન તરીકે ઓળખવામાં આવે છે. તે આપણે જેના પર કામ કરી રહ્યા છીએ તે કરન્ટ રેકોર્ડ બતાવે છે. આકૃતિ 2.19 માં બતાવ્યા પ્રમાણે ડેટા એન્ટર કરો. હવે, આપણી પાસે Student ટેબલમાં પાંચ રેકોર્ડ છે.



આકૃતિ 2.19 : સ્ટુડન્ટ ટેબલનો ડેટા

ડેટા એન્ટર કરતી વખતે જુઓ કે જ્યારે આપણે કોઈપણ રોમાં ડેટા એન્ટર કરીએ છીએ ત્યારે વાદળી ત્રિકોણ પેન્સિલ આઇકનમાં બદલાઈ જાય છે. ઉપરાંત, છેલ્લી ખાલી રોની શરૂઆતમાં લીલા રંગનું + સાઇન દેખાય છે. ટેબલમાં નવો રેકોર્ડ એડ કરવા માટે આપણે છેલ્લી રો પર સ્ક્રોલ કરવાની અને પછી તેના કોઈપણ ફીલ્ડમાં ક્લિક કરવાની જરૂર છે. હવે આપણું કર્સર પસંદ કરેલા ફીલ્ડમાં ગોઠવાઈ જશે અને આઇકન વાદળી એરો અને લીલા + સાઇનમાં બદલાઈ જશે.

આકૃતિ 2.18ના નીચેના ભાગને નેવિગેશનબાર તરીકે ઓળખવામાં આવે છે. તે ટેબલમાં રેકોર્ડની સંખ્યા બતાવે છે તેમજ કોઈપણ સિલેક્ટ કરેલા રેકોર્ડની સ્થિતિ બતાવવા માટે પણ સક્ષમ છે. તેમાં નેવિગેશન બટન પણ હોય છે, જે આપણને રેકોર્ડને વર્ટિકલી (ઊભા) સ્ક્રોલ કરવાની મંજૂરી આપે છે. આકૃતિ 2.19 જુઓ, સ્ક્રીનની નીચે ડાબી બાજુએ આપણને 'Record 5 of 5' દેખાય છે, જે સૂચવે છે કે ટેબલમાં કુલ પાંચ રો છે અને હાલમાં આપણે રો નંબર 5 પર છીએ.

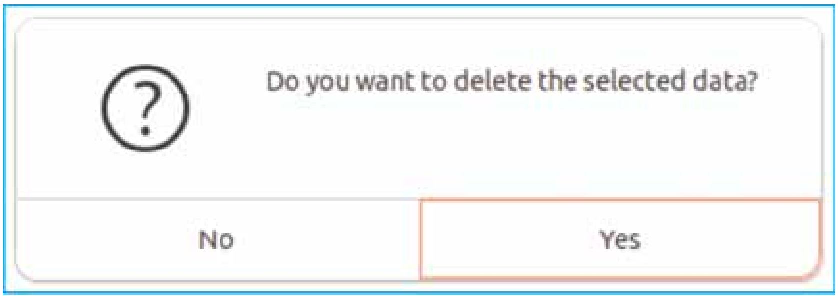
ડેટાને એડિટ કરવા (Edit Data)

ડેટાનું અપડેશન અથવા એડિટિંગ એ એક સતત પ્રક્રિયા છે. ખોટો ડેટા એન્ટર થવાને કારણે અથવા સાચા ડેટામાં ફેરફાર થવાને કારણે ડેટા અપડેટ કરવો પડે છે. ઉદાહરણ તરીકે, જો સની જયપુરથી અમદાવાદ શિફ્ટ થાય તો? આ કિસ્સામાં, જોકે સનીનું એડ્રેસ યોગ્ય રીતે એન્ટર કરવામાં આવ્યું હતું તેમ છતાં આપણે તેને એડિટ કરવાની જરૂર પડે છે.

ટેબલમાં અગાઉથી એન્ટર કરેલા ડેટાને સુધારવાની પ્રક્રિયા સામાન્ય રીતે એડિટિંગ તરીકે ઓળખાય છે. કોઈપણ ડેટા એડિટ કરવા માટે આપણે ટેબલને ડેટાશીટ વ્યૂમાં ખોલવાની, ઇચ્છિત ફીલ્ડ પર જવાની અને આપણે જે ફીલ્ડ વેલ્યૂ એડિટ કરવા માંગીએ છીએ તેના પર કર્સર મૂકવાની જરૂર રહે. આ રીતે આપણે જરૂરી કોઈપણ ફેરફાર કરી શકીએ છીએ.

ડેટાને ડિલિટ કરવા (Delete Data)

આપણા ડેટાબેઝને સ્વચ્છ રાખવા માટે ટેબલમાં કોઈપણ બિનજરૂરી



આકૃતિ 2.20 : ડિલીટ એલર્ટ બોક્ષ

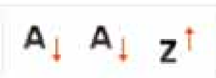
ડેટા અથવા રેકોર્ડને ડિલીટ કરવાની જરૂર પડે છે. ટેબલમાંથી રેકોર્ડ ડિલીટ કરવા માટે, ઇચ્છિત ટેબલને ડેટાશીટ વ્યૂમાં ખોલો. રેકોર્ડ અથવા એક કરતા વધુ રેકોર્ડ પસંદ કરો. (શરૂઆતનો રેકોર્ડ સિલેક્ટ કરીને, શીફ્ટ કી દબાવીને અને છેલ્લો રેકોર્ડ સિલેક્ટ કરીને એકસાથે વધુ રેકોર્ડ્સ સિલેક્ટ કરી શકાય છે.)

રેકોર્ડ ડિલીટ કરવાની સૌથી સરળ રીત કિબોર્ડ પર ડિલીટ કી દબાવવાની છે. વૈકલ્પિક રીતે, આપણે Edit મેનુમાંથી *Delete Record* ઓપ્શન સિલેક્ટ કરી શકીએ છીએ અથવા સિલેક્ટ કરેલા રેકોર્ડ પર રાઈટ ક્લિક કરીને સબમેનુમાંથી *Delete Rows* ઓપ્શન પસંદ કરી શકીએ છીએ. તમામ કિસ્સાઓમાં આપણને આકૃતિ 2.20માં બતાવ્યા પ્રમાણે એક એલર્ટ ડાયલોગ બોક્સ ડિસપ્લે થશે.

Yes બટન પર ક્લિક કરો એટલે રેકોર્ડ ટેબલમાંથી ડિલિટ થઈ જશે. જો યુઝર *No* બટન સિલેક્ટ કરેશે તો રેકોર્ડ પર કોઈ કાર્યવાહી કરવામાં આવશે નહીં.

ડેટાનું સોર્ટીંગ કરવું (Sort Data)

ટેબલમાં ડેટા સોર્ટ કરવાનો ઉદ્દેશ્ય જ્યારે જરૂર પડે ત્યારે તેને સરળતાથી એક્સેસ કરી શકવાનો છે. ટેબલની અંદરનો ડેટા એક ચોક્કસ રીતે ગોઠવવો જોઈએ જેથી ઇચ્છિત માહિતી મેળવવી સરળ બને. જેમ જેમ ટેબલમાં રેકોર્ડની સંખ્યા વધે છે, તેમ તેને શોધવાનું થોડું મુશ્કેલ બની શકે છે. ચોક્કસ ફીલ્ડ પર ડેટા સોર્ટ કરવો એ ડેટાને યોગ્ય રીતે ગોઠવવાની એક રીત છે. ધારો કે Student ટેબલમાં રહેલ ડેટામાંથી આપણે એ જાણવા માંગતા હોઈએ કે, અમદાવાદમાં કેટલા વિદ્યાર્થીઓ રહે છે?. જો ડેટા Address ફીલ્ડ પર સોર્ટ કરેલો હશે તો આપણને આ જવાબ સરળતાથી મળી શકશે. ચાલો Student ટેબલના ડેટાને સોર્ટ કરીએ.

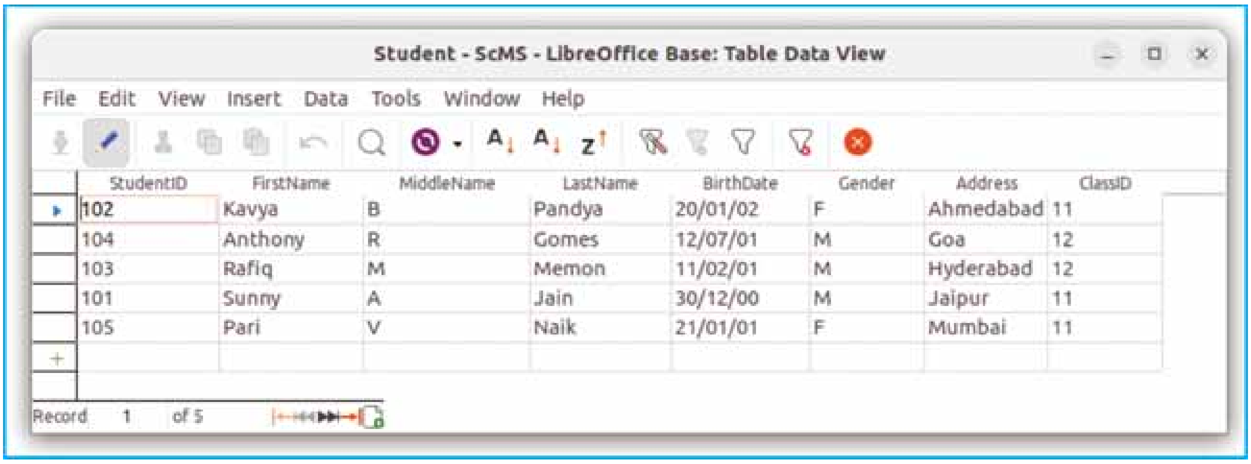
આપણે ટેબલને વિવિધ રીતે સોર્ટ કરી શકીએ છીએ, સૌથી સરળ રીત એ છે કે ડેટાશીટ વ્યૂ ટુલબારમાં બતાવેલા સોર્ટ આઈકન  નો ઉપયોગ કરવો. આ સોર્ટ આઈકન *Data* મેનુના *Sort...*, *Sort Ascending* અથવા *Sort Descending* ઓપ્શન જેવા જ છે.

Sort Ascending (ચડતા ક્રમમાં) અથવા *Sort Descending* (ઉતરતા ક્રમમાં) ઓપ્શન કર્સરના સ્થાનથી સોર્ટીંગ કરવા માટે વપરાય છે. ઉદાહરણ તરીકે, જો તમે MiddleName નામના ફીલ્ડના ચોથા રેકોર્ડ પર હોય અને આપણે આ ઓપ્શનનો ઉપયોગ કરીએ, તો ટેબલનો સમગ્ર ડેટા MiddleName ફીલ્ડના આધારે ચડતા અથવા ઉતરતા ક્રમમાં સોર્ટ કરવામાં આવશે.

આકૃતિ 2.21 : સોર્ટ ઓર્ડર ડાયલોગ બોક્ષ

Sort... ઓપ્શન એ એક એડવાન્સ સોર્ટ ઓપ્શન છે, તે આપણને એક કરતાં વધુ ફીલ્ડ સિલેક્ટ કરવાની મંજૂરી આપે છે. જ્યારે આપણે આ ઓપ્શનનો ઉપયોગ કરીએ છીએ ત્યારે આકૃતિ 2.21માં બતાવ્યા પ્રમાણે એક સોર્ટ ઓર્ડર ડાયલોગ બોક્સ બતાવવામાં આવે છે.

હવે યુઝર *Field name* લેબલ હેઠળ ડ્રોપડાઉન મેનુ માંથી યોગ્ય ફીલ્ડ નેમ પસંદ કરી શકશે. આ કિસ્સામાં આપણે ફીલ્ડ નેમ Address પસંદ કર્યું છે. આપણે વધુમાં વધુ ત્રણ ફીલ્ડના સંયોજનથી ડેટા સોર્ટ કરી શકીએ છીએ. સોર્ટેડ ટેબલ આકૃતિ 2.22માં બતાવ્યા પ્રમાણે છે



StudentID	FirstName	MiddleName	LastName	BirthDate	Gender	Address	ClassID
102	Kavya	B	Pandya	20/01/02	F	Ahmedabad	11
104	Anthony	R	Gomes	12/07/01	M	Goa	12
103	Rafiq	M	Memon	11/02/01	M	Hyderabad	12
101	Sunny	A	Jain	30/12/00	M	Jaipur	11
105	Pari	V	Naik	21/01/01	F	Mumbai	11

આકૃતિ 2.22 : Address ફીલ્ડ પર સોર્ટ કરેલ ટેબલ ડેટા

રીડન્ડન્સી અને કી (Redundancy and Keys)

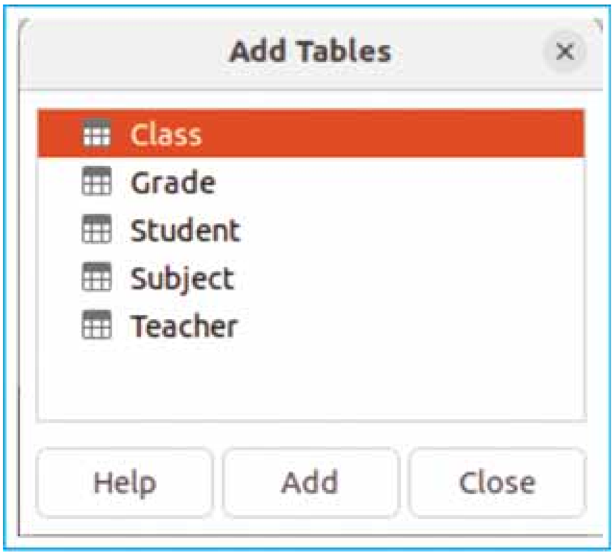
ઘણી વખત આપણે અવલોકન કરી શકીએ છીએ કે એક જ ડેટા એક કરતાં વધુ જગ્યાએ સ્ટોર થયેલ હોય છે, જેને રીડન્ડન્સી તરીકે ઓળખવામાં આવે છે. તે ડેટાબેઝની બિનકાર્યક્ષમતા, અસંગતતા અને વધુ સ્ટોરેજ ખર્ચ તરફ દોરી જાય છે. ઉદાહરણ તરીકે, આપણે બનાવેલા Grade ટેબલમાં વિદ્યાર્થી અને વિષય બંને વિશેની માહિતીની જરૂરિયાત છે. જો આપણે તેમાં વિદ્યાર્થીનું નામ તેમજ વિષયનું નામ સ્ટોર કર્યું હોત તો આપણે વિદ્યાર્થી અને વિષયનું નામ ઘણી વખત પુનરાવર્તિત કરવું પડ્યું હોત, જેનાથી રીડન્ડન્સી થાય.

રીડન્ડન્સી અનિચ્છનીય છે અને સામાન્ય રીતે ડેટાબેઝ નોર્મલાઈઝેશન પ્રક્રિયા દ્વારા દૂર કરવામાં આવે છે. અહીં આપણે ડેટાને સંબંધિત ટેબલમાં વિભાજિત કરીએ છીએ જેથી માહિતીનો દરેક ભાગ માત્ર એક જ વાર સ્ટોર થાય. પ્રકરણ-1 માં ચર્ચા કરેલી પ્રાઈમરી કી અને ફોરેન કી નોર્મલાઈઝેશન પ્રક્રિયામાં ખૂબ જ મહત્વપૂર્ણ ભૂમિકા ભજવે છે, કારણ કે તે ટેબલને એક કરતાં વધુ ટેબલમાં વિભાજિત કરવાની અને એક બીજા ટેબલને જોડવાની મંજૂરી આપે છે.

આપણે રીડન્ડન્સીને નિયંત્રિત કરવા માટે માસ્ટર ડેટા અથવા ઓટોમેટેડ ડેટા વેલિડેશન જેવી અન્ય પદ્ધતિઓનો ઉપયોગ કરી શકીએ છીએ. ScMS ડેટાબેઝના કિસ્સામાં Student, Teacher, Subject અને Class માસ્ટર ડેટા તરીકે ગણી શકાય, અને આ એન્ટિટીઝ વિશેની સામાન્ય માહિતી માટે એકમાત્ર વિશ્વસનીય સ્રોત તરીકે ઉપયોગમાં લઈ શકાય છે. અહીં Grade ટેબલ એક ટ્રાન્ઝેક્શન ટેબલ છે, કારણ કે તેમાં એવો ડેટા છે જેને કામચલાઉ ગણી શકાય.

રિલેશનશિપની રચના કરવી (Creating Relationship)

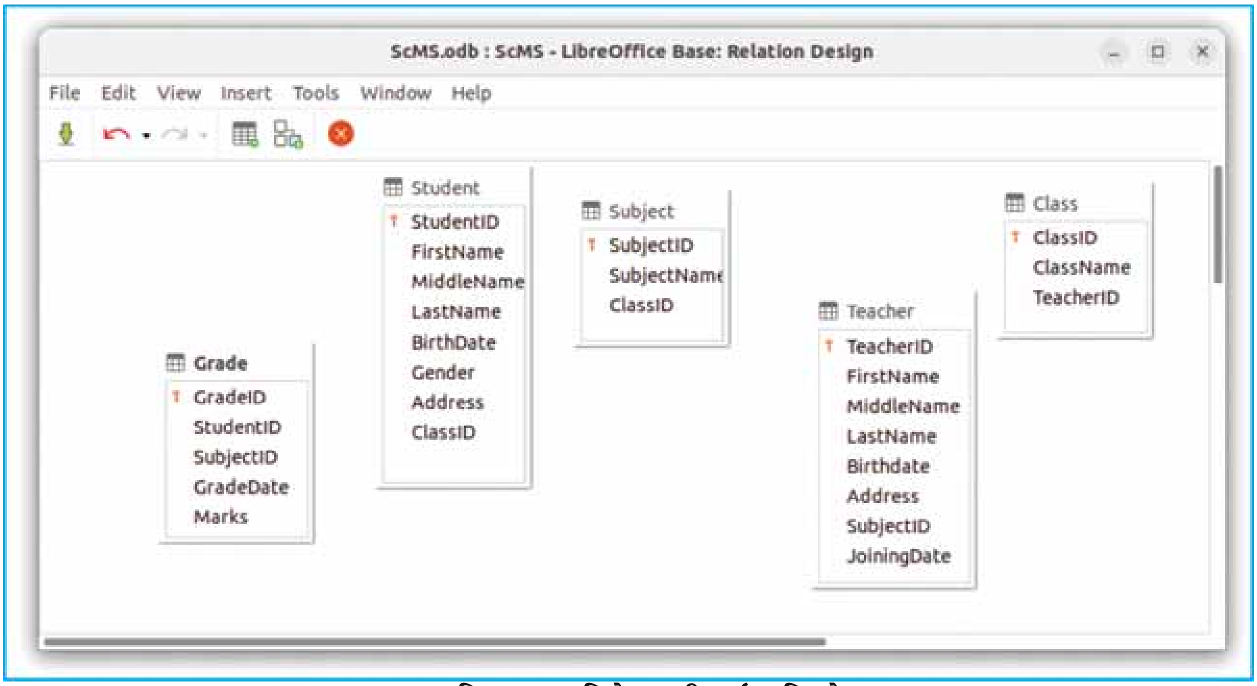
આપણે ScMS ડેટાબેઝમાં અત્યાર સુધી બનાવેલા ટેબલને રીલેશનશીપથી જોડેલ નથી. આપણે ફક્ત અમુક ફીલ્ડનો ઉપયોગ કર્યો છે જે સામાન્ય છે. ઉદાહરણ તરીકે, Student ટેબલમાં ClassID ફીલ્ડ છે, જે Class ટેબલનો પણ ભાગ છે. તેવી જ રીતે Teacher ટેબલમાં SubjectID ફીલ્ડ છે, જે પણ Subject ટેબલનો ભાગ છે. કોમન ફીલ્ડ નેમ એ ખાતરી આપતું નથી કે તેમાં એન્ટર કરેલો ડેટા હંમેશા સાચો હશે. જો કોઈ યુઝર Student ટેબલમાં ClassID 21 એન્ટર કરે પરંતુ તે Class ટેબલમાં ન હોય તો શું? આવા કિસ્સામાં ડેટાબેઝને અસંગત ગણવામાં આવશે. વિવિધ ટેબલ વચ્ચે યોગ્ય રિલેશનશિપની રચના કરવાથી આવી અસંગતતાઓ ન થાય તેની ખાતરી મળે છે. ચાલો હવે આપણે બનાવેલા વિવિધ ટેબલ વચ્ચે રિલેશનશિપ વિકસાવવાનું શીખીએ. ScMS વિન્ડૉ ખોલો, *Tools* મેનુ પર જાઓ, તેમાંથી



આકૃતિ 2.23 : Add ટેબલ ડાયલોગ બોક્સ

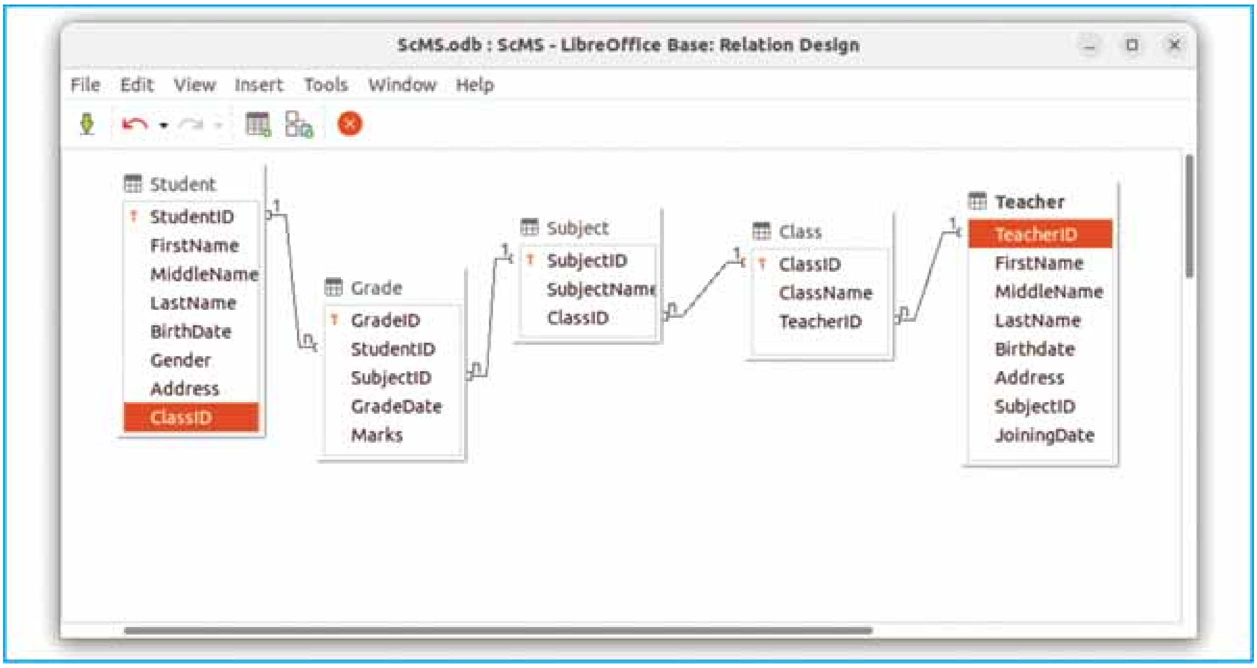
Relationships... ઓપ્શન સિલેક્ટ કરો. આમ કરવાથી આકૃતિ 2.23માં બતાવ્યા પ્રમાણે બનાવેલા ટેબલની યાદી સાથે *Add Tables* ડાયલોગ બોક્સ ખુલશે.

એક પછી એક ટેબલ પર ડબલ ક્લિક કરીને તેમને સિલેક્ટ કરો અથવા એક સમયે એક ટેબલ સિલેક્ટ કરીને Add બટન પર ક્લિક કરો, આપણે જોઈ શકીશું કે બેકગ્રાઉન્ડમાં સિલેક્ટ કરેલ ટેબલ તેના ફીલ્ડ સાથે *Relation Design* વિન્ડોમાં ડિસ્પ્લે થશે. બધા ટેબલ પસંદ થઈ જાય પછી *Add Tables* ડાયલોગ બોક્સ બંધ કરો. હવે *Relation Design* વિન્ડો આકૃતિ 2.24માં દર્શાવ્યા મુજબ દેખાશે.



આકૃતિ 2.24 : રિલેશન ડીઝાઈન વિન્ડો

આકૃતિ 2.24માં દેખાતા ટેબલના સ્થાનને ડ્રેગ એન્ડ ડ્રોપ ઓપરેશનનો ઉપયોગ કરીને ફરીથી ગોઠવી શકાય છે. રિલેશનશિપ બનાવતા પહેલા એ ખાતરી કરો કે, તમામ ટેબલમાં પ્રાઈમરી કી સેટ કરેલી છે અન્યથા રિલેશનશિપ બનાવતી વખતે ભૂલ થવાની સંભાવના છે.



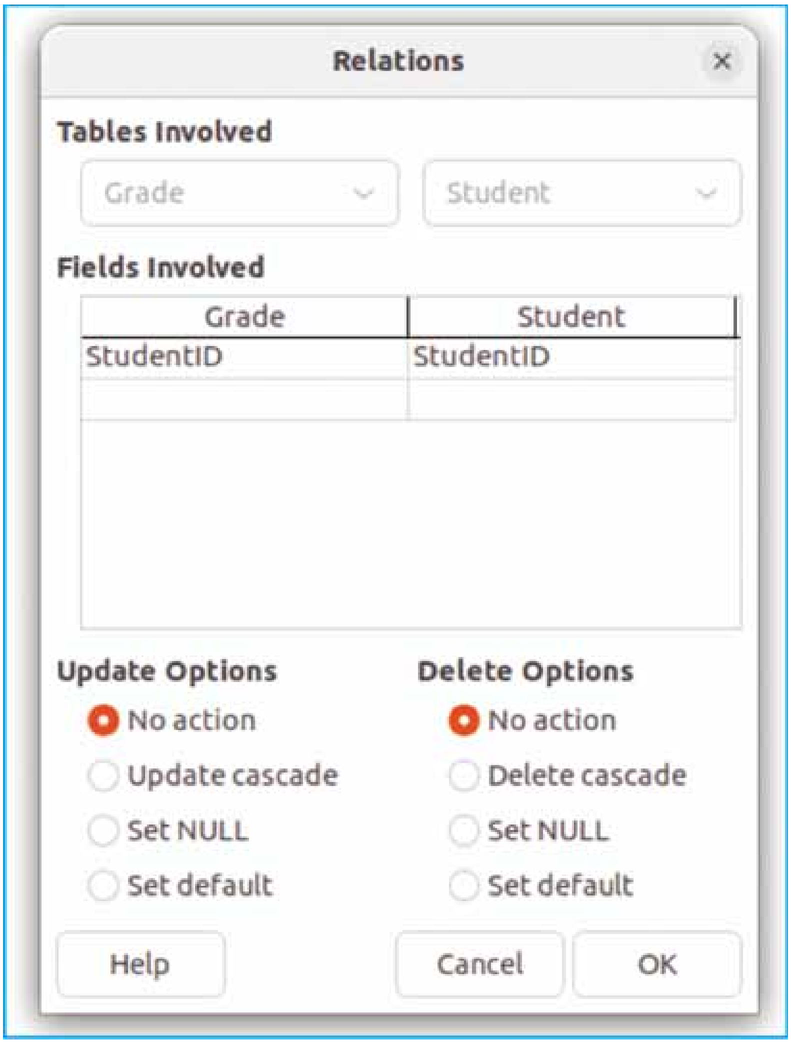
આકૃતિ 2.25 : સેમ્પલ રિલેશનશિપ સેટઅપ કર્યા પછી રિલેશન ડિઝાઈન વિન્ડો

અંતમાં, રિલેશનશિપ બનાવવા માટે, આપણે ડ્રેગ એન્ડ ડ્રોપ ઓપરેશનનો ઉપયોગ કરીશું. સૌપ્રથમ Student ટેબલના StudentID ફીલ્ડ પર ક્લિક કરો, લેફ્ટ માઉસ કી દબાવી રાખો અને માઉસને Grade ટેબલના StudentID ફીલ્ડ પર ડ્રેગ કરો અને લેફ્ટ માઉસ કી છોડી દો. આપણે બે ફીલ્ડ વચ્ચે જોડેલી એક રેખા જોઈશું, જેના પર લેબલ 1 અને n લખેલ છે. જુઓ કે Student ટેબલના StudentID (પ્રાઈમરી કી) પર લેબલ 1 ડિસ્પ્લે થાય છે અને Grade ટેબલના StudentID (ફોરેન કી) પર લેબલ *n* ડિસ્પ્લે થાય છે. તે સૂચવે છે કે Student ટેબલમાં StudentID ફીલ્ડ યુનિક વેલ્યૂ ધરાવશે જ્યારે Grade ટેબલમાં StudentID ફીલ્ડમાં વેલ્યૂ વારંવાર (n) હોય શકે છે.

નોંધ લો કે જ્યારે આપણે રિલેશનશિપ બનાવીએ છીએ, ત્યારે સંબંધિત બંને ફીલ્ડના ડેટા ટાઈપ સમાન હોવા જોઈએ. શક્ય તમામ રિલેશન બનાવો અને તેમને સેવ કરો, હવે રિલેશન ડિઝાઈન વિન્ડો આકૃતિ 2.25માં દર્શાવ્યા મુજબ દેખાશે.

એકવાર ટેબલ વચ્ચે રિલેશનશિપ બનાવવામાં આવે પછી, આપણે ડેટાબેઝના ડેટા ઈન્ટેગ્રિટી કન્સ્ટ્રેઈન્ટ્સ (Integrity Constraints) પર કામ કરી શકીએ છીએ. આપણે જાણીએ છીએ કે ScMS ડેટાબેઝમાં Student ટેબલ અને Grade ટેબલ એકબીજા સાથે જોડાયેલ છે. આ રિલેશન જણાવે છે કે Grade ટેબલમાં એન્ટર કરેલ StudentID પહેલા Student ટેબલમાં ઉપલબ્ધ હોવો જોઈએ. Student ટેબલમાં વિદ્યાર્થીઓના સંબંધિત રેકોર્ડને માસ્ટર અથવા પેરેંટ રેકોર્ડ ગણવામાં આવે છે, જ્યારે Grade ટેબલમાં વિદ્યાર્થીઓના સંબંધિત રેકોર્ડને ટ્રાન્ઝેક્શન અથવા ચાઈલ્ડ રેકોર્ડ ગણવામાં આવે છે.

રેફરન્શિયલ ઈન્ટેગ્રિટી (Referential Integrity) : ધારો કે, એક પરિસ્થિતિ એવી છે કે જ્યારે યુઝર Student ટેબલ માંથી પેરેંટ રેકોર્ડ ડિલિટ કરે, ત્યારે ટ્રાન્ઝેક્શન ટેબલમાં તેના સંબંધિત ચાઈલ્ડ રેકોર્ડનું શું થવું જોઈએ? ઉદાહરણ તરીકે, જો Student ટેબલમાં StudentID તરીકે 101 વેલ્યૂ ધરાવતો રેકોર્ડ છે અને Grade ટેબલમાં આ StudentID ને અનુરૂપ અનેક રેકોર્ડ છે. જો યુઝર માસ્ટર ટેબલમાંથી StudentID 101 વાળો રેકોર્ડ ડિલિટ અથવા અપડેટ કરે તો Grade ટેબલમાં 101 વેલ્યૂ સાથેના StudentIDના રેકોર્ડનું શું થશે? રેફરેન્શલ ઈન્ટેગ્રિટી નિયમ કહે છે કે માસ્ટર ટેબલમાં સંબંધિત ડેટા વિના ટ્રાન્ઝેક્શન ટેબલમાં ડેટાની કોઈ એન્ટ્રી હોવી જોઈએ નહીં. આમ, ડેટાબેઝમાં એકસમાન ના હોય તેવી ફોરેન કી વેલ્યૂ ન હોવી જોઈએ.



આકૃતિ 2.26 : રેફરન્શિયલ ઈન્ટેગ્રિટી સેટ કરવા માટે રિલેશન્સ ડાયલોગ બોક્સ

રેકોર્ડ પર અપડેટ અથવા ડિલિટ ઓપરેશન કરતી વખતે ડેટાબેઝમાં રેફરન્શિયલ ઈન્ટેગ્રિટી લાગુ કરવાની જરૂર છે. ચાલો હવે Student અને Grade ટેબલ પર રેફરન્શિયલ ઈન્ટેગ્રિટી નિયમો લાગુ કરવાનો પ્રયાસ કરીએ. આકૃતિ 2.25માં Student અને Grade ટેબલ વચ્ચે દેખાતી રિલેશનશિપ લાઈન પર ડબલ ક્લિક કરો. આનાથી આકૃતિ 2.26માં બતાવ્યા પ્રમાણે રિલેશન્સ ડાયલોગ બોક્સ ખુલશે.

હવે ડેટાબેઝ ડિઝાઈનર ઓર્ગેનાઈઝેશનના વ્યવહારની જરૂરિયાતના આધારે આકૃતિ 2.26માં દેખાતા ચાર ઓપ્શનમાંથી એકની પસંદગી કરી શકે છે. આપણે અપડેટ અને ડિલિટ ઓપરેશન માટે જુદા જુદા ઓપ્શન પસંદ કરી શકીએ છીએ. ઉદાહરણ તરીકે, આપણે *Update Options* લેબલ હેઠળ *No action* ઓપ્શન તેમજ *Delete Option* લેબલ હેઠળ *Delete cascade* ઓપ્શન પસંદ કરી શકીએ છીએ. ચાલો આમાંના દરેક વિકલ્પનો અર્થ શું છે તે જોઈએ.

નો એક્શન (No action) : જો બીજા ટેબલમાં રીલેટેડ રેકોર્ડ હોય અને આ ઓપ્શન સિલેક્ટ કરેલો હોય તો, યુઝર રેકોર્ડ ડિલીટ કે અપડેટ કરી શકશે નહીં. ઉદાહરણ તરીકે, જો આપણે Student ટેબલમાં StudentID 101 ધરાવતો રેકોર્ડ ડિલીટ કરવાનો પ્રયાસ કરીએ, તો આપણને આમ કરવાની મંજૂરી આપવામાં આવશે નહીં. કારણ કે Grade ટેબલમાં અનુરૂપ રેકોર્ડ અસ્તિત્વમાં છે.

અપડેટ કાસ્કેડ (Update cascade) : જો માસ્ટર રેકોર્ડને અપડેટ કરવામાં આવે તો આ ઓપ્શન તમામ સંબંધિત ફીલ્ડને પણ સાથે સાથે અપડેટેડ કરશે. ઉદાહરણ તરીકે, જો આપણે Student ટેબલમાં StudentID 101 ધરાવતા રેકોર્ડ ને 125 થી અપડેટ કરીએ, તો Grade ટેબલમાંના તમામ અનુરૂપ StudentID રેકોર્ડની વેલ્યૂ 125 કરવામાં આવશે.

ડિલીટ કાસ્કેડ (Delete cascade) : જો માસ્ટર રેકોર્ડ ડિલીટ થઈ જાય તો આ ઓપ્શન તમામ સંબંધિત ફીલ્ડને ડિલિટ કરી નાખશે. ઉદાહરણ તરીકે, જો આપણે Student ટેબલમાંથી StudentID 101 ધરાવતો રેકોર્ડ ડિલિટ કરીએ, તો Grade ટેબલમાંના તમામ અનુરૂપ StudentID રેકોર્ડ પણ ડિલિટ થઈ જશે.

સેટ નલ (Set NULL) : જો માસ્ટર રેકોર્ડ ડિલિટ અથવા અપડેટ થઈ જાય તો આ ઓપ્શન તમામ સંબંધિત ફીલ્ડને NULL વેલ્યૂ અસાઈન કરશે. ઉદાહરણ તરીકે, જો આપણે Student ટેબલમાં StudentID 101 ધરાવતો રેકોર્ડ ડિલિટ કરીએ, તો Grade ટેબલમાંના તમામ અનુરૂપ StudentID રેકોર્ડની વેલ્યૂ પણ NULL થઈ જશે.

સેટ ડિફોલ્ટ (Set default) : જો માસ્ટર રેકોર્ડ ડિલિટ અથવા અપડેટ થાય તો આ ઓપ્શન તમામ સંબંધિત ફીલ્ડની કોઈ નિશ્ચિત ડિફોલ્ટ વેલ્યૂ અસાઈન કરશે. ઉદાહરણ તરીકે, જો આપણે Student ટેબલમાં StudentID 101 ધરાવતો રેકોર્ડ ડિલિટ કરીએ, તો Grade ટેબલમાંના તમામ અનુરૂપ StudentIDના રેકોર્ડને ડિફોલ્ટ વેલ્યૂ અસાઈન થશે.

જો જરૂર હોય તો ટેબલ વચ્ચે સ્થાપિત થયેલ રિલેશનશિપને આપણે ડિલિટ અથવા એડિટ કરી શકીએ છીએ. આ કરવા માટે, *Relationship Design* વિન્ડો ખોલો, ટેબલ વચ્ચે દેખાતી રિલેશનશિપ લાઈન સિલેક્ટ કરો, તેના પર રાઈટ ક્લિક કરો, *Delete* અને *Edit...* ઓપ્શન સાથેનું એક પોપઅપ મેનુ દેખાશે. જરૂરી કાર્યવાહી કરી અને રિલેશનશિપને ફરીથી સેવ કરો.

હવે એવો ડેટાબેઝ તૈયાર છે જેનો ઉપયોગ ડેટા સ્ટોર કરવા અને તેનું વિશ્લેષણ કરવા માટે થઈ શકે છે. ડિઝાઈન કરેલા તમામ ટેબલમાં યોગ્ય રેકોર્ડ એન્ટર કરો, જેથી આપણે ક્વેરીનો ઉપયોગ કરીને તેમાંથી માહિતી મેળવી શકીએ.

સારાંશ

આ પ્રકરણમાં આપણે ડેટાબેઝ બનાવવા માટેનું ટૂલ , લિબ્રેઓફિસ બેઝનો ઉપયોગ કેવી રીતે કરવો તે શીખ્યા. જે ચાર ઓબ્જેક્ટ એટલે કે ટેબલ, કવેરી, ફોર્મ અને રિપોર્ટ પુરા પાડે છે. વિઝાર્ડ અને કસ્ટમ ડિઝાઈનનો ઉપયોગ કરીને આપણે બે રીતે ટેબલ બનાવતા શીખ્યા. વિઝાર્ડ એ એક માર્ગદર્શિત સ્ટેપ-બાય-સ્ટેપ ગ્રાફિકલ ઇન્ટરફેસ ટૂલ છે, જે યુઝર્સને માહિતી પૂછીને અને કાર્ય કરવાની પ્રક્રિયાને સ્વયંસંચાલિત કરીને જટિલ કાર્યોને સરળ બનાવે છે. આપણે ફીલ્ડ પ્રોપર્ટીઝ ઓપ્શન વિશે પણ શીખ્યા, જેનો ઉપયોગ ટેબલના રેકોર્ડમાં એન્ટર કરવાના ડેટાને નિયંત્રિત અને વેલીડેટ કરવા માટે થઈ શકે છે. અંતે આપણે ટેબલ વચ્ચે રિલેશનશિપ કેવી રીતે સેટ કરવી તે શીખ્યા, જે આપણને ડેટા રીડન્ડન્સીને નિયંત્રિત કરવાની મંજૂરી આપે છે અને રેફરન્શિયલ ઈન્ટેગ્રિટી પણ સ્થાપિત કરે છે. હવે આપણે ટેબલ બનાવવા અને તેના પર વિવિધ કામગીરી કરવા માટે તૈયાર છીએ. આગલા પ્રકરણ માં આપણે કવેરીનો ઉપયોગ કરીને ટેબલમાંથી જરૂરી માહિતી કેવી રીતે મેળવવી તે શીખીશું.

સ્વાધ્યાય

1. ડેટાબેઝ એટલે શું? ડેટાબેઝનો ઉપયોગ શું છે?

2. ટેબલ શું છે? ટેબલ બનાવવાની વિવિધ રીતો સમજાવો.

3. વિઝાર્ડનો ઉપયોગ શું છે?

4. ટેબલ વચ્ચેના રીલેશનનું મહત્વ સમજાવો.

5. ટેબલના રીલેશનશીપમાં 1 અને n લેબલ ના મહત્વ ને સમજાવો.

6. યોગ્ય ઉદાહરણ આપીને ડેટા રીડન્ડન્સીનો ખ્યાલ સમજાવો.

7. નોર્મલાઇઝેશનનો ઉપયોગ શું છે?

8. ડેટાબેઝમાં રેફરન્શિયલ ઇન્ટિગ્રિટી શા માટે જરૂરી છે તે જણાવો.

9. ફીલ્ડ પ્રોપર્ટી શું સૂચવે છે?

10. ફોર્મેટ ફીલ્ડ પ્રોપર્ટીનું મહત્વ સમજાવો.

11. સાચું કે ખોટું જણાવો.

(1) લિબ્રેઓફીસ બેઝ આપણને ટેક્સ્ટ ડોક્યુમેન્ટ્સ બનાવવાની સવલત આપે છે.

(2) Authors ટેબલ લિબ્રેઓફીસ બેઝના બીઝનેસ કેટેગરી હેઠળ આપવામાં આવ્યું છે.

(3) ટેબલ વિઝાર્ડનો ઉપયોગ કરીને જરૂરિયાત મુજબનું ટેબલ બનાવી શકાય છે.

(4) ફીલ્ડ પ્રોપર્ટીઝ વિકલ્પનો ઉપયોગ કરીને આપણે ડેટાને નિયંત્રિત અને વેલીડેટ કરી શકીએ છીએ.

(5) રેકોર્ડ સિલેક્ટર આઇકોન આપણે જે ટેબલને એડિટ કરી રહ્યા છીએ તેનો વર્તમાન રેકોર્ડ દર્શાવે છે.

12. ખાલી જગ્યા પૂરો.

(1) લીબ્રેઓફીસ બેઝ તેમાં બનાવેલા ડેટાબેઝને ડિફોલ્ટ રૂપે ____________ એક્સટેન્શન આપે છે.

(2) ટેબલ વિઝાર્ડ બે કેટેગરીના ટેબલના ટેમ્પ્લેટ્સ પૂરા પાડે છે, ____________ અને પર્સનલ.

(3) ટેબલ ડિઝાઇન વ્યૂમાં આપણે ફીલ્ડનું નામ, ટાઈપ, ____________ અને પ્રોપર્ટીઝ જોઈ શકીએ છીએ.

(4) ઓટોવેલ્યૂ પ્રોપર્ટીનો ઉપયોગ ____________ ફીલ્ડ સાથે થાય છે

(5) રીડન્ડન્સી ____________ નામના કોન્સેપ્ટ દ્વારા દૂર કરવામાં આવે છે.

13. બહુવિકલ્પી પ્રશ્નો. સૌથી યોગ્ય જવાબ પસંદ કરો.

(1) લિબ્રેઓફીસ બેઝ આપણને નીચેનામાંથી કેટલી રીતે ડેટાબેઝ બનાવવાની મંજૂરી આપે છે?

(a) 1 (b) 2 (c) 3 (d) 4

(2) જ્યારે આપણે લિબ્રેઓફીસ બેઝ ડેટાબેઝ વિન્ડો ખોલીએ છીએ ત્યારે નીચેનામાંથી કયો ઑબ્જેક્ટ દેખાતો નથી?

(a) ટેબલ (Table) (b) ક્વેરીઝ (Queries) (c) ફોર્મ્સ (Forms) (d) વ્યૂઝ (Views)

(3) ડેટાબેઝમાં રેકોર્ડ્સને યોગ્ય રીતે ગોઠવવા માટે નીચેનામાંથી કઈ કામગીરી કરી શકાય છે?

(a) ઈન્સર્ટ (Insert) (b) ડીલીટ (Delete) (c) અપડેટ (Update) (d) સોર્ટ (Sort)

(4) ડેટાબેઝમાં ડેટા રીડન્ડન્સીનું શ્રેષ્ઠ વર્ણન નીચેનામાંથી કયું કરે છે?

(a) ટેબલનું વિઘટન (b) ટેબલને જોડવા

(c) ડેટાનું પુનરાવર્તન (d) ડેટાને ડીલીટ કરવા

(5) નીચેનામાંથી કયું વિધાન સાચું નથી?

(a) પ્રાઈમરી કી અને ફોરેન કીના નામ સમાન હોવા જોઈએ.

(b) દરેક ટેબલમાં પ્રાઈમરી કી હોવી આવશ્યક છે.

(c) એક પ્રાઈમરી કીમાં એક કરતાં વધુ ફીલ્ડ હોઈ શકે છે.

(d) પ્રાઈમરી કી અને ફોરેન કીનો ડેટા ટાઈપ સમાન હોવો જોઈએ.

(6) જો આપણે Student અને Grade ટેબલના StudentID ફીલ્ડને જોડીએ તો નીચેનામાંથી કયા લેબલ રીલેશન લાઈન પર દેખાશે?

(a) n…n (b) n…1 (c) 1…n (d) No labels

(7) ટેબલ વચ્ચે રેફરન્શિયલ ઈન્ટિગ્રિટી સેટ કરતી વખતે, જો તેનો સંબંધિત રેકોર્ડ અન્ય ટેબલમાં હોય તો નીચેનામાંથી કયો વિકલ્પ યુઝરને રેકોર્ડ ડીલીટ કરવા અથવા અપડેટ કરવાની મંજૂરી નહિ આપે?

(a) સેટ ડીફોલ્ટ (Set default) (b) સેટ નલ (Set NULL)

(c) નો એક્શન (No action) (d) ડીલીટ કાસકેડ (Delete cascade)

(8) નીચેનામાંથી કયા ફીલ્ડ પ્રોપર્ટીઝને Not NULL ની સમકક્ષ ગણી શકાય?

(a) ડીફોલ્ટ (Default) (b) ફોર્મેટ (Format)

(c) લેન્થ (Length) (d) રીક્વાયર્ડ (Required)

(9) એન્ટિટી વિશે સામાન્ય માહિતી માટે એકમાત્ર સ્રોત તરીકે ઉપયોગમાં લેવાતો ડેટા નીચેનામાંથી કયા સ્ટોર થાય છે?

(a) ટ્રાન્જેક્શન ટેબલ (Transaction Table) (b) માસ્ટર ટેબલ (Master Table)

(c) બીગ ટેબલ (Big Table) (d) એકાઉન્ટ ટેબલ (Accounts Table)

(10) ફોરેન કીનો ઉપયોગ કરવાથી નીચેનામાંથી કયો ફાયદો થાય છે?

(a) ડેટા કન્સિસ્ટન્સી (Data consistency)

(b) ઝડપી ડેટા રીટ્રાઈવલ (Faster data retrieval)

(c) ડેટા રીડન્ડન્સીમાં સુધારો (Improved data redundancy)

(d) સ્ટોરેજ સ્પેસમાં ઘટાડો કરે છે (Reduced storage space)

પ્રાયોગિક સ્વાધ્યાય

1. નીચેના ટેબલમાં આપેલ ડેટાબેઝ માટે નીચેની કામગીરી કરો:

(a) ટેબલ બનાવો.

(b) દરેક ટેબલ માટે યોગ્ય પ્રાઈમરી કી અને ફોરેન કી ઓળખો.

(c) ટેબલમાં દરેક એટ્રીબ્યુટ માટે કઈ ડેટા ટાઈપ યોગ્ય રહેશે તે નક્કી કરો.

(d) ટેબલ વચ્ચે રીલેશનશીપ સેટ કરો.

(e) દરેક ટેબલમાં ઓછામાં ઓછા દસ રેકોર્ડ એન્ટર કરો.

DB1	Student (StudentId, StudentName, Address, City, BirthDate, ContactNo, Email) Book (BookId, BookTitle, Description, BookAuthor, Status, PublishYear) Book_Issue (BookIssueId, BookId, StudentId, IssueDate, ReturnDate, FineAmount)
DB2	Product (ProductId, ProductName, Quantity, ProductPrice, ManufactureYear) Salesman (SalesmanID, SalesmanName, Address, BirthDate, ContactNo, Gender) SalesOrder (SalesID, SalesmanID, ProductId, QtySold)
DB3	Customer (CustomerId, CustomerName, Gender, Address, City, Area, Email, ContactNo) Magazine (MagazineId, MagazineName, UnitRate, PublisherName, PublishedMonth) Subscription (CustomerId, MagazineId, StartDate, EndDate)
DB4	Employee (EmployeeID, EmployeeName, Address, City, Salary, DesignationID) Designation (DesignationID, DesignationName, BasicSalary) Project (ProjectID, ProjectName, StartDate, ProjectPrice) ProjectWorkHrs (PId, ProjectID, EmployeeID, HoursWorked)
DB5	Vehicle (VehicleId, VehicleType, Price, Description, ManufactureDate) Customer (CustomerId, CustomerName, Address, BirthDate, ContactNo) VehicleOwner (VehicleId, CustomerId, PurchaseDate, DeliveryDate)

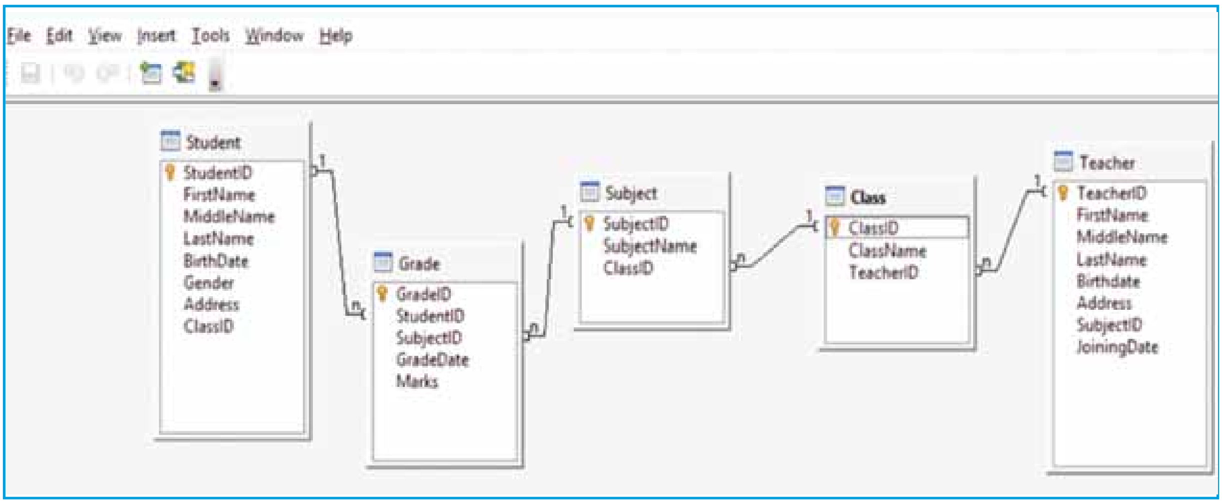
ક્વેરીનો ઉપયોગ કરીને ડેટાની પુનઃપ્રાપ્તિ

પરિચય

અગાઉના પ્રકરણોમાં, આપણે ડેટાબેઝ વિશે અને લિબ્રેઓફિસ બેઝનો ઉપયોગ કરીને ડેટાબેઝ કેવી રીતે બનાવી શકાય તે વિશે અભ્યાસ કર્યો. બેઝ ફ્રી અને ઓપન-સોર્સ રિલેશનલ ડેટાબેઝ મેનેજમેન્ટ સિસ્ટમ છે જે ડેટાબેઝ બનાવવા માટે ગ્રાફિકલ ઇન્ટરફેસ પૂરું પાડે છે. આપણે જાણીએ છીએ કે, બેઝમાં ડેટા ટેબલ સ્વરૂપમાં સ્ટોર થાય છે. બેઝ ટેબલમાં ડેટા સ્ટોર કરવા અને ટેબલમાંથી ડેટાને પુનઃપ્રાપ્ત (રીટ્રાઈવ) કરવા માટે યુઝર ફ્રેન્ડલી પદ્ધતિ પ્રદાન કરે છે. ક્વેરી (Query) ભાષાનો ઉપયોગ કરીને ડેટાબેઝ સાથે સંવાદ (ઇન્ટરેક્શન) થાય છે. આ પ્રકરણમાં, આપણે ડેટાબેઝમાંથી માહિતી કેવી રીતે પુનઃપ્રાપ્ત કરવી તે વિશે શીખીશું. આપણે સૌ પ્રથમ ક્વેરી વિશે શીખીશું અને પછી આપણે ક્વેરીનો ઉપયોગ કરીને ડેટા કેવી રીતે પુનઃપ્રાપ્ત કરી શકીએ તે વિશે ચર્ચા કરીશું.

ક્વેરી પ્રક્રિયા માટેનું ઉદાહરણ ડેટાબેઝ (Example database for query processing)

અગાઉના પ્રકરણમાં ચર્ચા કરેલા પગલાંને અનુસરીને, ચાલો પહેલા આકૃતિ 3.1માં બતાવ્યા મુજબ એક ડેટાબેઝ બનાવીએ. આ એક સ્ટુડન્ટ ડેટાબેઝ છે, જેમાં વિદ્યાર્થીઓ, તેમના ગ્રેડ, વિષયો, ક્લાસ અને શિક્ષકો સંબંધિત માહિતીનો સંગ્રહ કરે છે. હવે, જો આપણે જાણવા માંગીએ કે, ‘કેટલા વિદ્યાર્થીઓ 70% થી વધુ માર્ક્સ સાથે ઉત્તીર્ણ થયા છે?’ અથવા ‘કેટલા વિદ્યાર્થીઓ વર્ગ-IXમાં અભ્યાસ કરી રહ્યા છે?’, તો આપણે ડેટાબેઝમાં ક્વેરી કરી અને આપણા પ્રશ્નોના જવાબો મેળવી શકીએ છીએ. આગળના વિભાગમાં, આપણે ડેટાબેઝમાંથી માહિતી પુનઃપ્રાપ્ત કરવા માટે ક્વેરી કેવી રીતે લખવી તે વિશે શીખીશું.



આકૃતિ 3.1 : સ્ટુડન્ટ ડેટાબેઝ

ક્વેરી (Query) શું છે?

ડેટાબેઝ ક્વેરીએ ડેટાબેઝના ડેટા પુનઃપ્રાપ્ત કરવા, સુધારવા અથવા મેનેજ કરવા માટે કરવામાં આવેલી વિનંતી છે. તે ડેટાબેઝ સાથે ક્રિયા પ્રતિક્રિયા કરવા માટેના પ્રાથમિક માધ્યમ તરીકે કાર્ય કરે છે, જે યુઝર અને એપ્લીકેશનને સંગ્રહિત માહિતીને પ્રાપ્ત કરવા અને તેમાં ફેરફાર કરવા કાર્યક્ષમ રીતે સક્ષમ બનાવે છે. મૂળભૂત રીતે, એક ડેટાબેઝ ક્વેરી એ ક્વેરી લેંગ્વેજ (સામાન્ય રીતે SQL - Structured Query Language)માં લખાયેલો એક કમાન્ડ છે, જે નીચે મુજબની ક્રિયાઓ કરવા માટે વપરાય છે, જેમ કે:

- **ડેટા પુનઃપ્રાપ્ત કરવા :** એક અથવા વધુ ટેબલમાંથી ચોક્કસ માહિતી પુનઃપ્રાપ્ત કરવા
- **ડેટા ઇન્સર્ટ કરવા :** ટેબલમાં નવા રેકોર્ડ ઉમેરવા.

- **ડેટા અપડેટ કરવા :** હાલના રેકોર્ડને સુધારવા.
- **ડેટાને ડીલીટ કરવા :** ટેબલમાંથી રેકોર્ડ ડીલીટ કરવા.

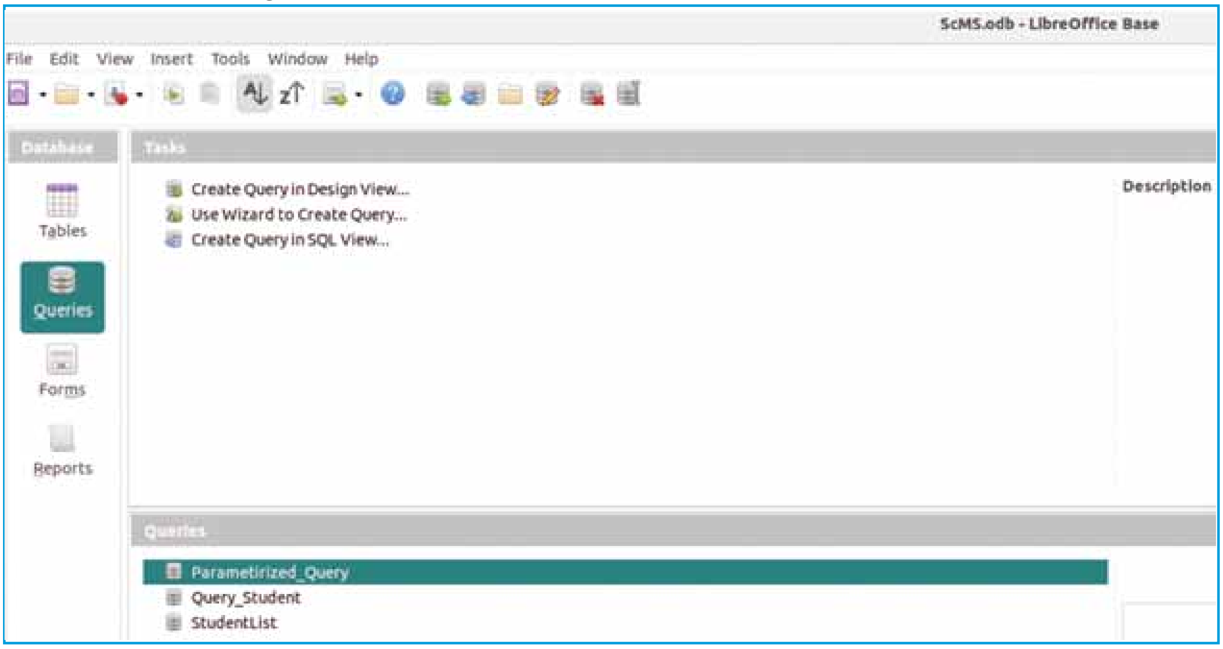
ટેબલ 3.1 મૂળભૂત SQL ક્વેરીનો સારાંશ આપે છે.

રીટ્રાઈવલ	*SELECT * FROM STUDENT;* STUDENT ટેબલમાંથી તમામ ડેટા રીટ્રાઈવ કરે છે અને ડિસ્પ્લે કરે છે.
ઈન્સર્ટ	*INSERT INTO SUBJECT (SubjectID, SubjectName, ClassID) VALUES (1, Science, 9);* Subject ટેબલમાં SubjectID = 1, SubjectName = Science અને ClassID = 9 સાથે એક નવો રેકર્ડ ઈન્સર્ટ કરે છે.
અપડેટ	*UPDATE SUBJECT SET ClassID = 8 WHERE SubjectID = 1;* Subject ટેબલમાં SubjectID = 1 ધરાવતી રોમાં ClassIDની વેલ્યુ બદલીને 8 કરશે.
ડીલીટ	*DELETE FROM SUBJECT* *Subject* ટેબલની તમામ રો ડીલીટ કરશે. *DELETE FROM SUBJECT WHERE SubjectName = 'Science';* SubjectName માં 'Science' છે તે તમામ રો Subject ટેબલમાંથી ડીલીટ કરશે.

ટેબલ 3.1 : SQL ક્વેરીના ઉદાહરણો

બેઝ ક્વેરી લખવા માટે યુઝર ફ્રેન્ડલી ઈન્ટરફેસ પૂરો પાડે છે. આ ક્વેરી ઈન્ટરફેસનો ઉપયોગ કરીને, આપણે ડેટાબેઝ ટેબલમાંથી માહિતી મેળવવા માટે નિયમોનો એક સમૂહ નિર્ધારિત કરી શકીએ છીએ. બીજા શબ્દોમાં કહીએ તો, કોઈ વ્યક્તિ બેઝને જણાવી શકે છે કે, ડેટાબેઝમાંથી તે કયા ફીલ્ડ અને રેકોર્ડ જોવા માંગે છે. ક્વેરીનું પરિણામ એક ટેબલના સ્વરૂપમાં મળે છે. તે રો અને કોલમમાં ગોઠવાયેલા રેકોર્ડના સમૂહનું બનેલું હોય છે.

જ્યારે આપણે બેઝમાં કોઈ ડેટાબેઝ ખોલીએ છીએ, ત્યારે ડાબી બાજુની પેનલમાં *Queries* આઈકન દેખાય છે. ડાબી બાજુના પેનલમાં રહેલ *Queries* આઈકન પર ક્લિક કરવાથી, *Tasks* પેનલમાં ક્વેરી બનાવવા માટેની વિવિધ રીતો દર્શાવવામાં આવે છે. આકૃતિ 3.2 *Queries* આઈકન સિલેક્ટ કરવામાં આવે ત્યારે દેખાતી વિન્ડો દર્શાવે છે.



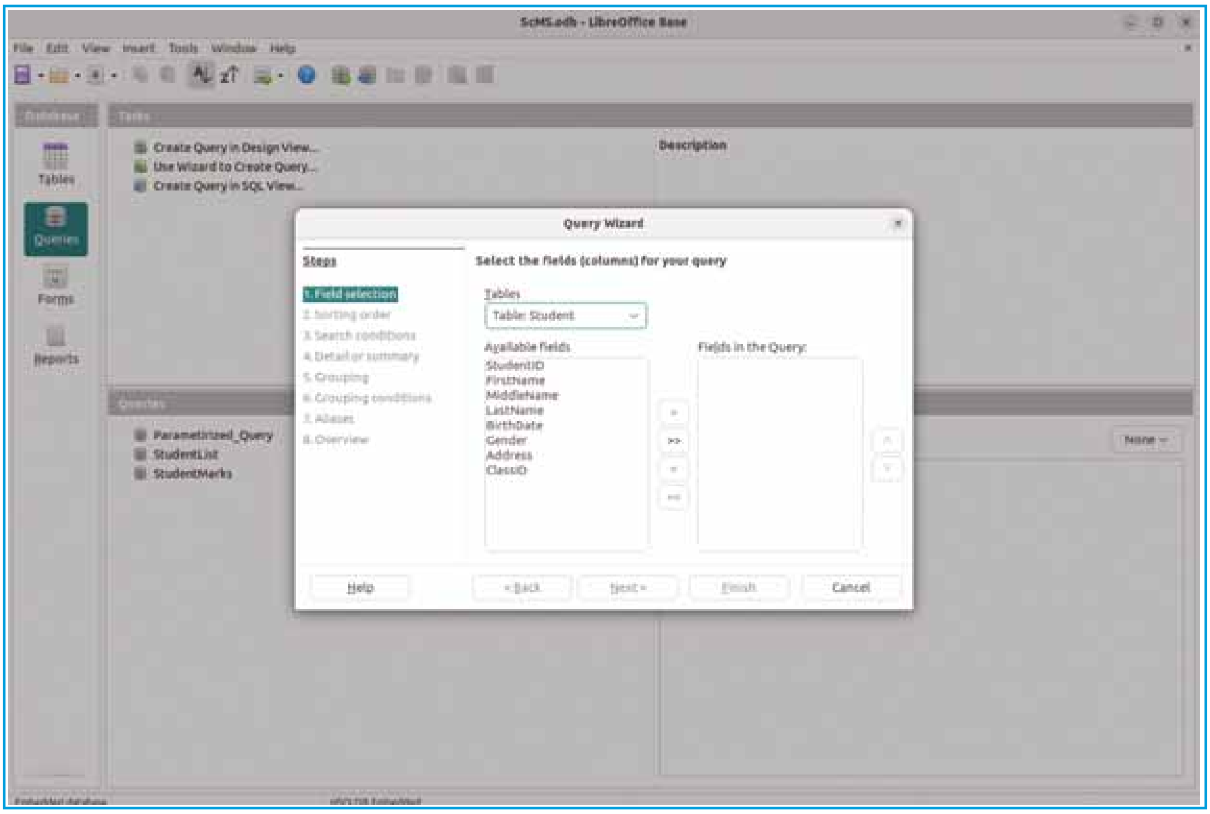
આકૃતિ 3.2 : ક્વેરી વિન્ડો

વિઝાર્ડનો ઉપયોગ કરીને ક્વેરી બનાવવી (Creating Query Using Wizard)

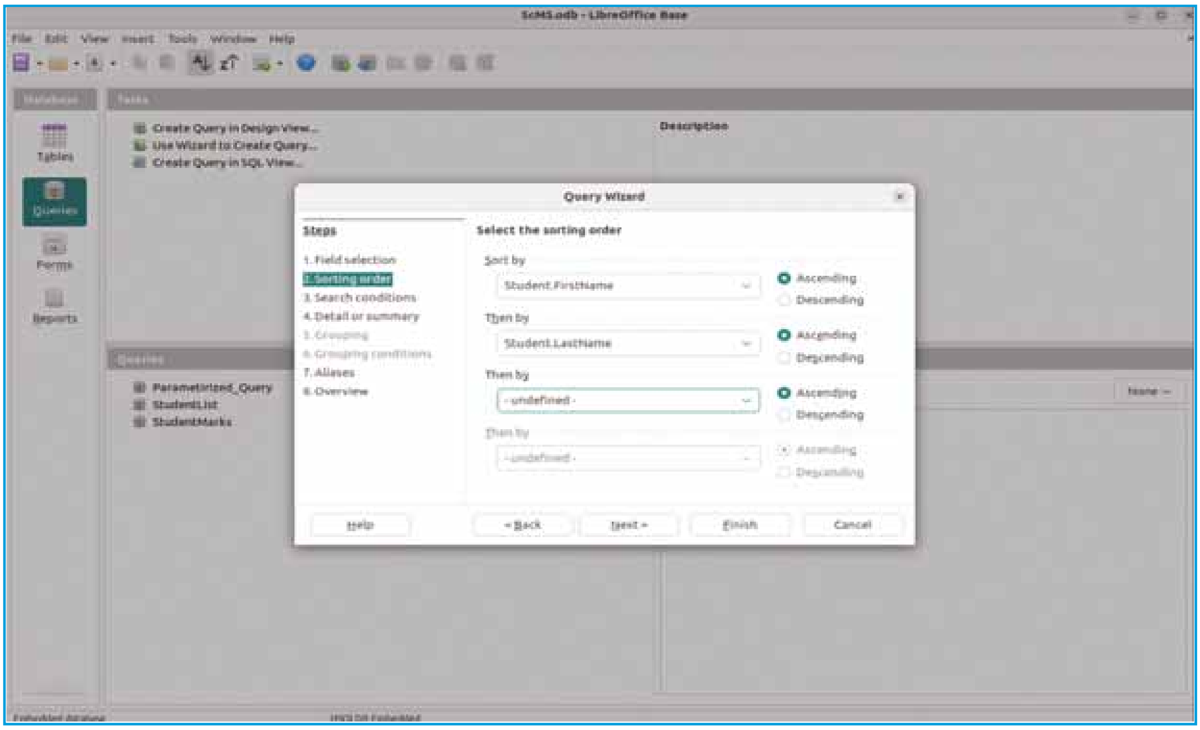
ડેટાબેઝમાં ક્વેરી બનાવવા માટે સૌથી સરળ રીત ક્વેરી વિઝાર્ડ છે. *Use Wizard to Create Query'* વિકલ્પ પર ડબલ ક્લિક કરવાથી આકૃતિ 3.3 માં બતાવ્યા મુજબ ક્વેરી વિઝાર્ડ ડાયલોગ બોક્ષ ખુલે છે. ડાયલોગ બોક્ષના ડાબી બાજુની પેનલમાં જોઈ શકાય છે કે, ક્વેરી બનાવવા માટે કુલ આઠ સ્ટેપ્સની જરૂર પડે છે. જો કે, તે બધા ફરજિયાત નથી. ખરેખર તો, ફક્ત પહેલું સ્ટેપ, *Field selection* જ ફરજિયાત છે. તે આપણને આઉટપુટમાં જે ફીલ્ડ ડિસ્પ્લે થવાના છે તે ફીલ્ડને ઓળખવામાં મદદ કરે છે. બાકીના બીજા સ્ટેપ આઉટપુટને ફોર્મેટ કરવાની સવલત પૂરી પાડે છે છે અને જો તેની જરૂર ન હોય તો તેને છોડી શકાય છે.

પગલું-1 : ક્વેરી બનાવવાનું પ્રથમ સ્ટેપ એ ટેબલ અને તે ટેબલમાંથી પુનઃપ્રાપ્ત કરવાની માહિતીના ફીલ્ડના સમૂહને પસંદ કરવાનું છે. એકવાર આપણે *Tables* લેબલ હેઠળના ડ્રોપડાઉનમાંથી ટેબલ પસંદ કરીએ, પછી ડાબી બાજુનું *Available fields* લીસ્ટ બોક્ષ તે ટેબલના ફીલ્ડ દર્શાવે છે, જેનો ઉપયોગ ક્વેરીમાં થઈ શકે છે. આપણે ડાબા અને જમણા એરોનો ઉપયોગ કરીને *Available fields* માંથી *Fields in the Query* તરફ અને તેનાથી વિપરીત ફીલ્ડને મુવ કરી શકીએ છીએ. જે ફીલ્ડને આપણે આપણી ક્વેરીનો ભાગ બનાવવા માટે પસંદ કરીએ છીએ તે ફીલ્ડ *Field in the Query* લિસ્ટ બોક્ષ હેઠળ સૂચિબદ્ધ થશે. ત્યારબાદ આ ફીલ્ડને અપ અને ડાઉન બટનનો ઉપયોગ કરીને જરૂરી ક્રમમાં ગોઠવી શકાય છે. એકવાર ફીલ્ડ નક્કી થઇ જાય, પછી *Next* બટન પર ક્લિક કરો. આકૃતિ 3.3માં અવલોકન કરો કે, આપણે *Table: Student* પસંદ કર્યું છે અને તેની સાથે સંબંધિત તમામ ફીલ્ડને *Available fields* લિસ્ટ બોક્ષ હેઠળ જોઈ શકાય છે.

પગલું-2 : બીજું સ્ટેપ ક્વેરીના આઉટપુટને જે ક્રમમાં ડિસ્પ્લે કરવાનું છે તે માટેનું સોર્ટ ઓર્ડરનું પગલું છે. તે આપણને આઉટપુટનો સોર્ટીંગ ઓર્ડર નક્કી કરવા માટે વધુમાં વધુ ચાર ફીલ્ડ પસંદ કરવાની મંજૂરી આપે છે. ઉદાહરણ તરીકે, આપણે ક્વેરીનું પરિણામ શરૂઆતમાં *FirstName*ના ક્રમમાં અને ત્યારબાદ વિદ્યાર્થીના *LastName*ના ક્રમમાં ગોઠવેલું જોવા માંગી શકીએ છીએ. એકવાર સોર્ટીંગ ઓર્ડર નક્કી થઈ જાય, પછી *Next* બટન પર ક્લિક કરો. આકૃતિ 3.4 ફીલ્ડનો સોર્ટીંગ ઓર્ડર કેવી રીતે પસંદ કરવો તે દર્શાવે છે.

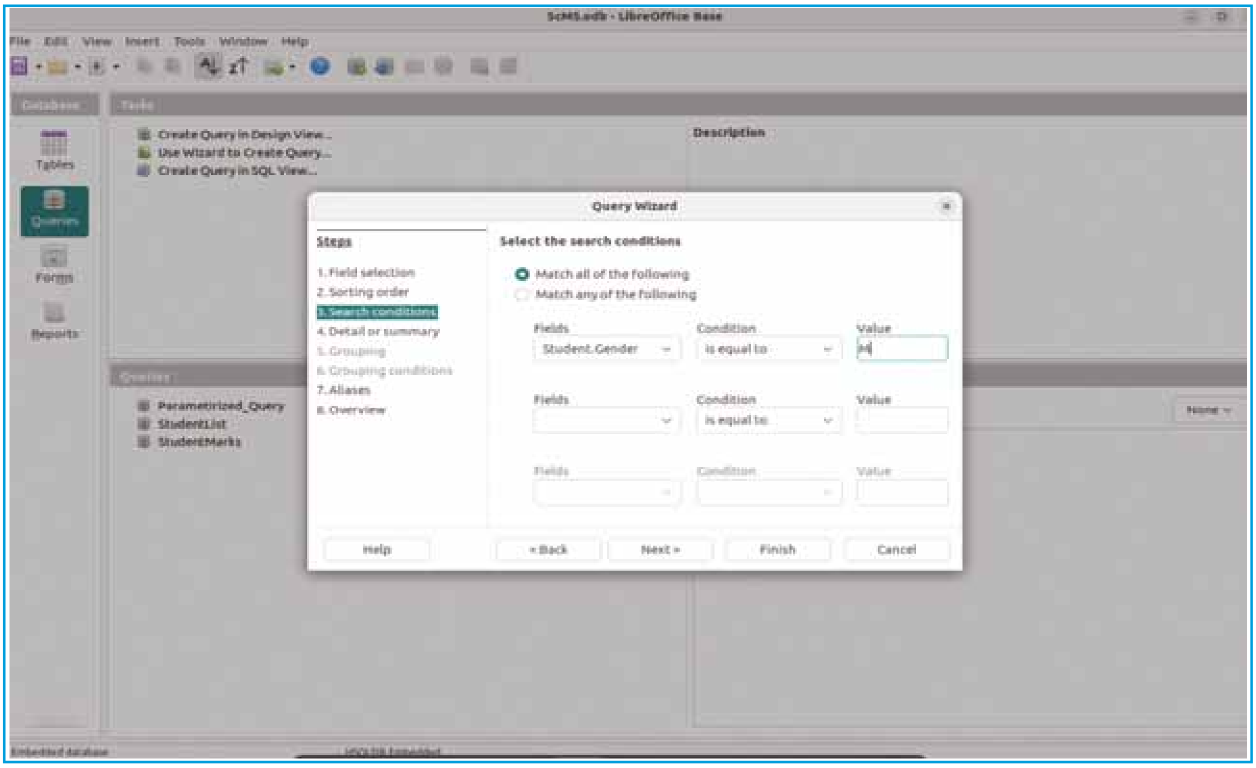


આકૃતિ 3.3 : ક્વેરી માટે ટેબલ અને ફીલ્ડની પસંદગી



આકૃતિ 3.4 : ફીલ્ડ પર સોર્ટિંગ એપ્લાઈ કરવું

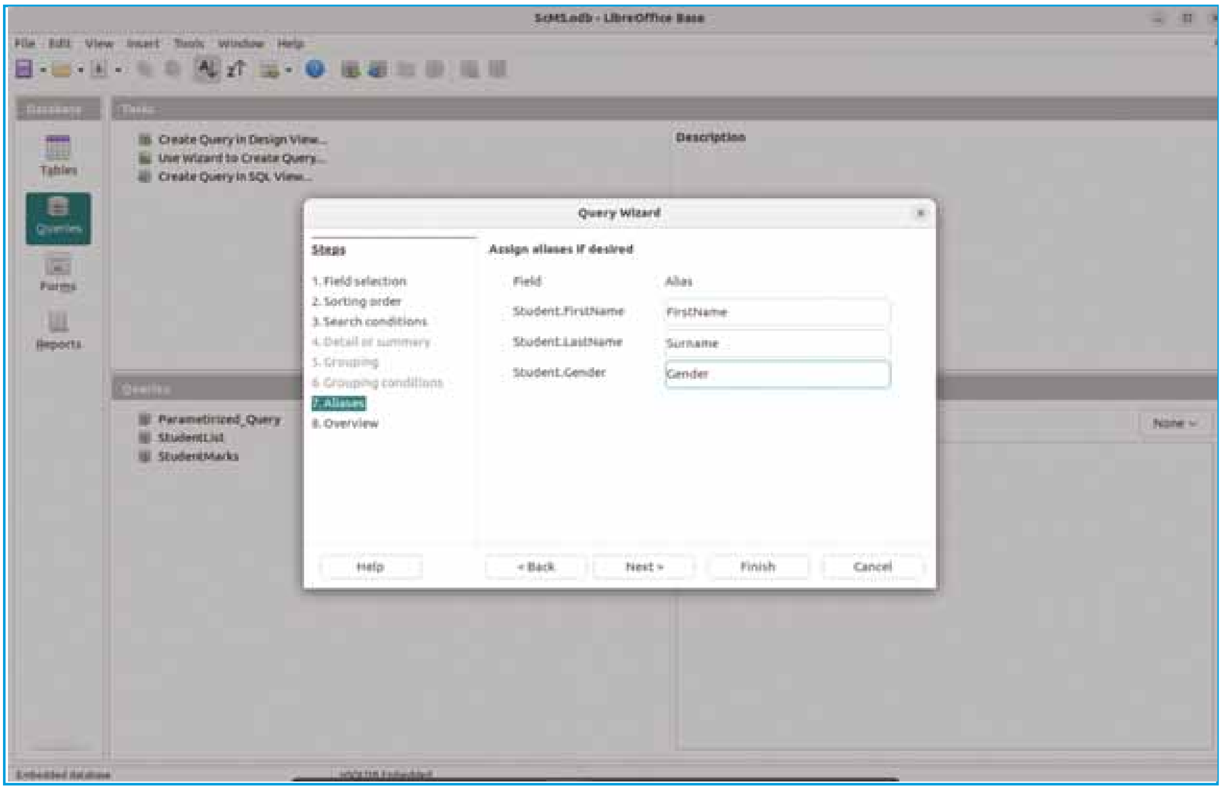
પગલું-3 : વિઝાર્ડના ત્રીજા સ્ટેપમાં, આપણે ક્વેરી સેટ કરીએ છીએ. અહીં આપણે *Fields, Condition* અને *Value* પેરામીટર્સ માટે યોગ્ય વેલ્યુ પસંદ કરવાની હોય છે. આપણે એક ક્વેરીમાં વધુમાં વધુ ત્રણ સર્ચ કન્ડિશન નિર્ધારિત કરી શકીએ છીએ. વિદ્યાર્થીઓના નામ ડિસ્પ્લે કરવાના કિસ્સામાં, જો આપણે માત્ર પુરુષ વિદ્યાર્થીઓનું લિસ્ટ જોઈતું હોય, તો ક્રાઈટેરિયા એકદમ સરળ રહેશે, આપણે *Fields* લેબલ હેઠળના ડ્રોપડાઉન માંથી *Gender* ફીલ્ડ પસંદ કરીશું. ત્યારબાદ *Condition* લેબલ હેઠળના ડ્રોપડાઉનમાંથી *is equal to* પસંદ કરીશું અને અંતે,



આકૃતિ 3.5 : ફીલ્ડ પર સર્ચ કન્ડિશન લાગુ કરવી

Value લેબલ હેઠળના ટેક્સ્ટ બોક્ષમાં *M* ટાઈપ કરીશું. ધ્યાન રાખો કે આ સ્ટેપમાં આપણી પાસે બે વિકલ્પો છે : *Match all of the following* અને *Match any of the following.* આપણી પાસે માત્ર એક જ કન્ડિશન હોવાથી, આપણે ડિફોલ્ટ સેટિંગ બદલવાની જરૂર નથી. જો એક કરતા વધુ સર્ચ કન્ડિશન સેટ કરવાની હોય, જેમ કે આપણે *M* અથવા *F* જેવી લિંગ ધરાવતા વિદ્યાર્થીઓને શોધી રહ્યા હોઈએ, તો આપણે *Match any of the following* પસંદ કરવાની જરૂર રહે છે. એકવાર સર્ચ કન્ડિશન નક્કી થઇ જાય પછી *Next* બટન પર ક્લિક કરો. આકૃતિ 3.5 તમામ પુરુષ વિદ્યાર્થીઓની યાદી માટેની સર્ચ કંડીશન દર્શાવે છે.

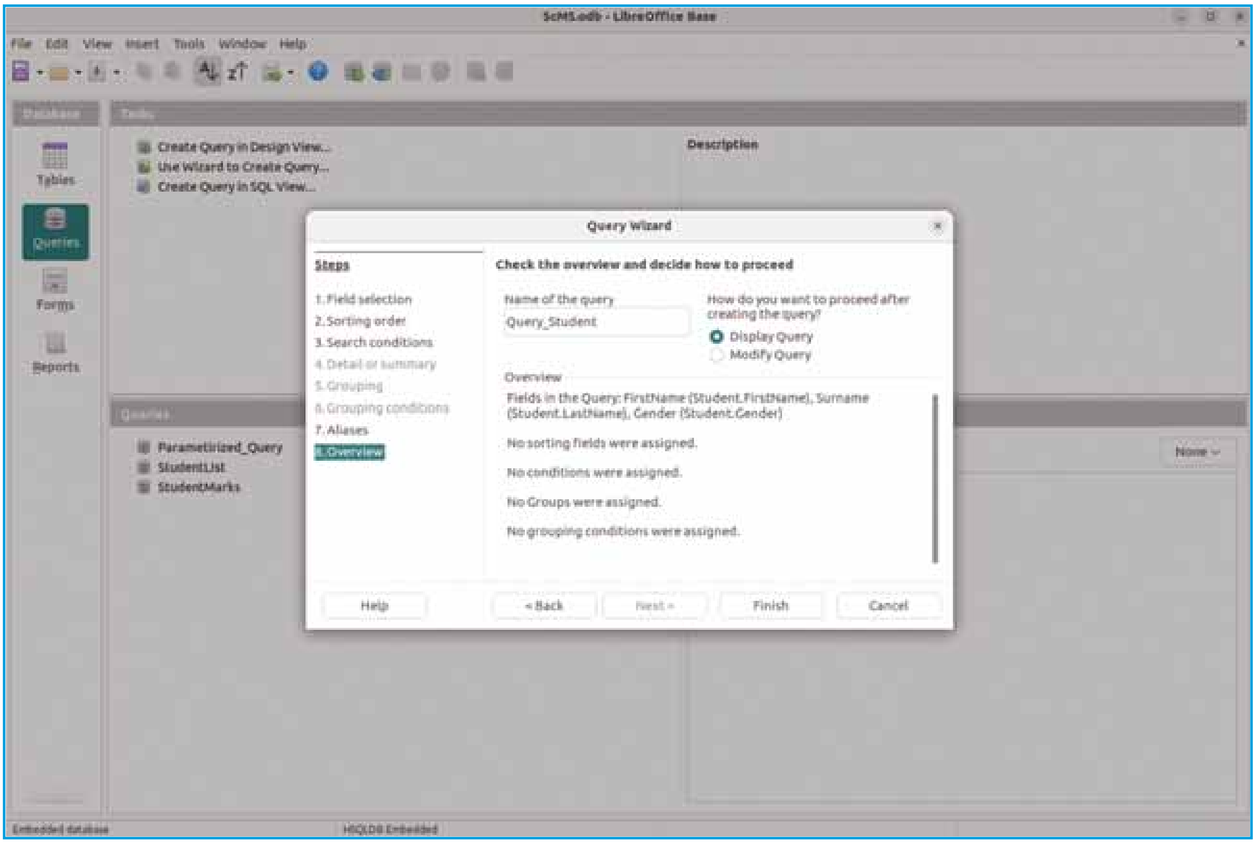
પગલું-4, 5, 6 : આ સ્ટેપ માહિતીને સંક્ષિપ્ત કરવા અને તેના ગ્રુપ કરવા માટેના વિકલ્પોને લગતા છે. આપણી ઉદાહરણ ક્વેરીમાં, પસંદ કરેલ Student ટેબલમાં કોઈ ન્યુમેરિક ફીલ્ડ નથી અને તેથી માહિતીને સંક્ષિપ્ત કરવા અથવા આંકડાકીય ગણતરીઓ કરવાના વિકલ્પો ધરાવતા સ્ટેપને છોડી દેવામાં આવ્યા છે.



આકૃતિ 3.6 : ઉપનામ ઉમેરવા

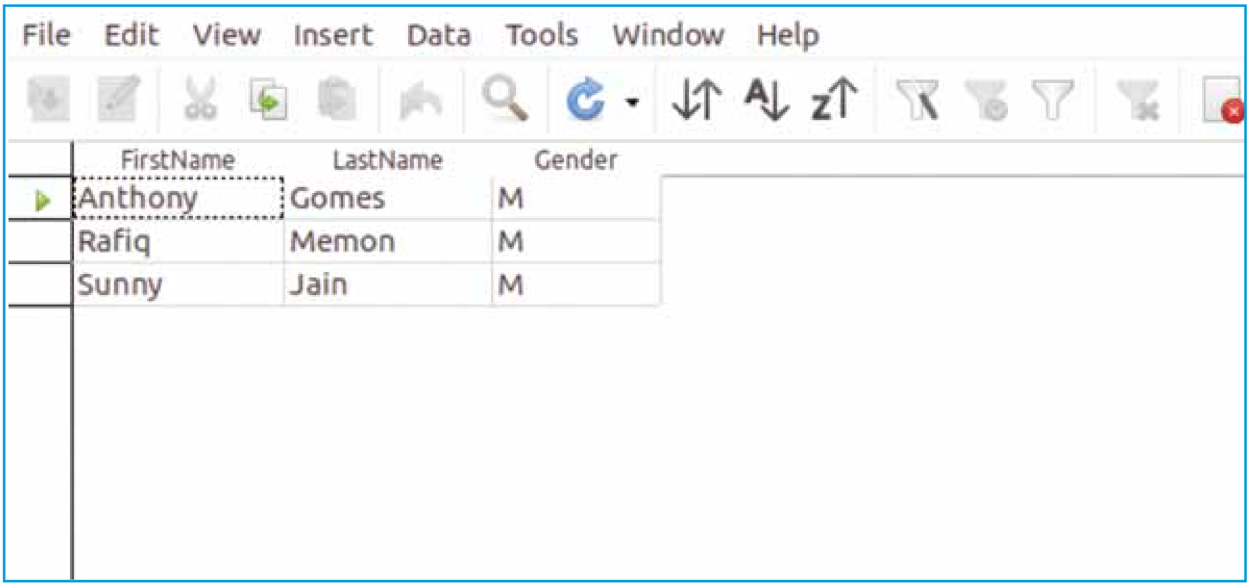
પગલું - 7 : સાતમા સ્ટેપમાં, બેઝ પસંદ કરેલા ફીલ્ડના નામો માટે ઉપનામો આપવાની સગવડ આપે છે. સામાન્ય રીતે ડેટાબેઝ પ્રોગ્રામર દ્વારા આપવામાં આવેલ વાસ્તવિક ફીલ્ડ નામ યુઝર માટે સરળતાથી સમજી શકાય તેવા ફોર્મેટમાં ન પણ હોય. ઉદાહરણ તરીકે, ડેટાબેઝ ફીલ્ડ નેમમાં *first_name* જેમાં કેટલાક સ્પેશીયલ કેરેક્ટર હોઈ શકે છે, અથવા તે *fName;* જેવું સંક્ષિપ્ત હોઈ શકે છે. આવા કિસ્સાઓ માટે, આપણે *FirstName* જેવું સહજ રીતે સમજી શકાય તેવું ઉપનામ આપી શકીએ છીએ. આમ, ઉપનામ બનાવવાનો હેતુ એ છે કે ક્વેરી વિઝાર્ડ ફીલ્ડના નામને વાંચવામાં સરળ રહે તેવા સ્વરૂપમાં ડિસ્પ્લે કરે છે. આ સ્ટેપ પણ વૈકલ્પિક છે; જો જરૂરી હોય તો જ આપણે ઉપનામ ઉમેરી શકીએ છીએ. જો આપણે ઉપનામ ઉમેરીશું નહીં તો, ટેબલના ફીલ્ડના નામ તે જ રીતે ડિસ્પ્લે થશે. એકવાર બધા ઉપનામ નક્કી થઈ જાય, પછી *Next* બટન પર ક્લિક કરો. આકૃતિ 3.6 ઉપનામ કેવી રીતે ઉમેરવા તે દર્શાવે છે.

પગલું - 8 : અંતે, આઠમા સ્ટેપમાં, આપણે અત્યાર સુધી કરેલા તમામ સ્ટેપની એક ઝાંખી દર્શાવવામાં આવે છે. આકૃતિ 3.7 હાલમાં બનાવેલ ક્વેરીની ઝાંખી દર્શાવે છે. *Name of the query* લેબલવાળા ટેક્સ્ટ બોક્ષમાં ક્વેરીનું નામ ટાઈપ કરો. આપણા કિસ્સામાં, આપણે તેને *Query_Student* નામ આપ્યું છે. આપણે શું કર્યું છે તેના પર એક ક્ષણ માટે ધ્યાન આપો. જો કોઈ ફેરફાર કરવાના હોય, તો *Back* બટનનો ઉપયોગ કરીને જરૂરી ફેરફારો કરો.



આકૃતિ 3.7 : બનાવેલ ક્વેરીની ઝાંખી

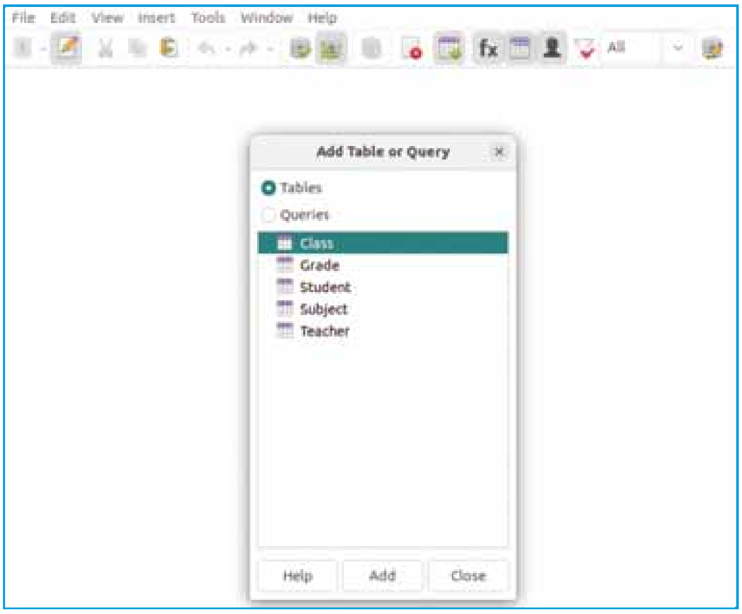
આપણે *Finish* બટન પર ક્લિક કરીશું ત્યારે ક્વેરીનું પરિણામ આકૃતિ 3.8માં બતાવ્યા મુજબ ડેટાશીટ વ્યૂમાં ડિસ્પ્લે થશે. (નોંધ લો કે 'એન્થોની ગોમ્સ, રફીક મેનન અને સની જૈન' નામો ડેટાબેઝ ટેબલમાંથી મેળવેલ વેલ્યુ છે. જો આપણે ''રમેશ, સુરેશ'' જેવા અન્ય નામો ઇન્સર્ટ કર્યા હશે, તો તે નામો બતાવશે.)



આકૃતિ 3.8 : ક્વેરીનું પરિણામ

ડિઝાઈન વ્યૂનો ઉપયોગ કરીને ક્વેરી બનાવવી

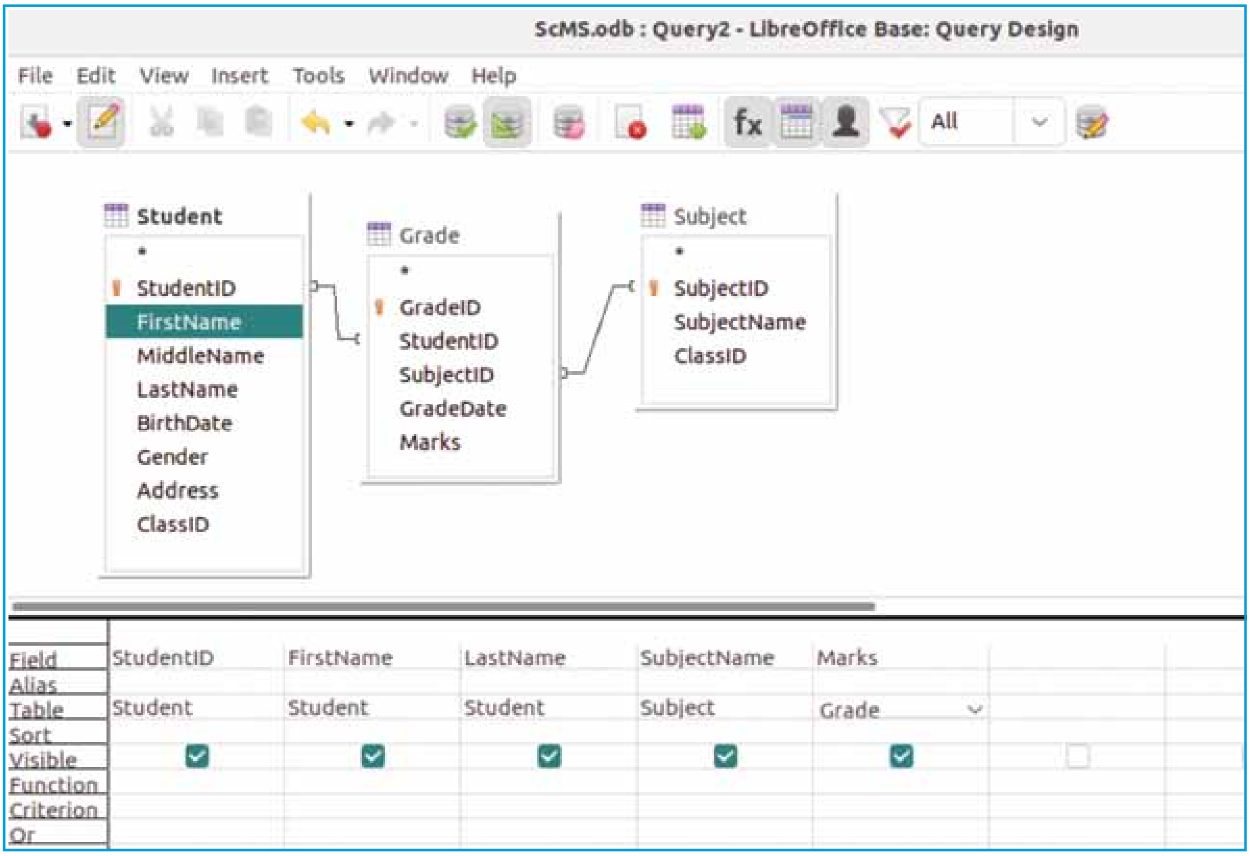
Query Wizard કે જેનો ઉપયોગ આપણે અગાઉના વિભાગમાં કર્યો છે, તે સરળ છે અને નવા શીખનારાઓ માટે ઉપયોગી છે. બેઝમાં ક્વેરી બનાવવા માટે ડિઝાઈન વ્યૂ બીજો એક વિકલ્પ છે. જ્યારે *Query Wizard* એ મૂળભૂત કવેરી બનાવવા માટે એક માર્ગદર્શક, સ્ટેપ-બાય-સ્ટેપ ટુલ છે અને *Query Design View* ક્વેરી બનાવવા અને તેમાં ફેરફાર કરવા માટે વધુ અદ્યતન નિયંત્રણ અને અનુકૂલન કે ફેરફારના વિકલ્પો પ્રદાન કરે છે. વિઝાર્ડ સરળ કવેરી માટે અને એવા યુઝર્સ માટે મદદરૂપ છે કે જેઓ વધુ માર્ગદર્શિત અભિગમ પસંદ કરે છે, જ્યારે ડીઝાઈન વ્યૂ જટિલ કવેરી માટે અથવા જ્યારે વધુ બારીક નિયંત્રણની જરૂર હોય ત્યારે ઉપયોગી છે.



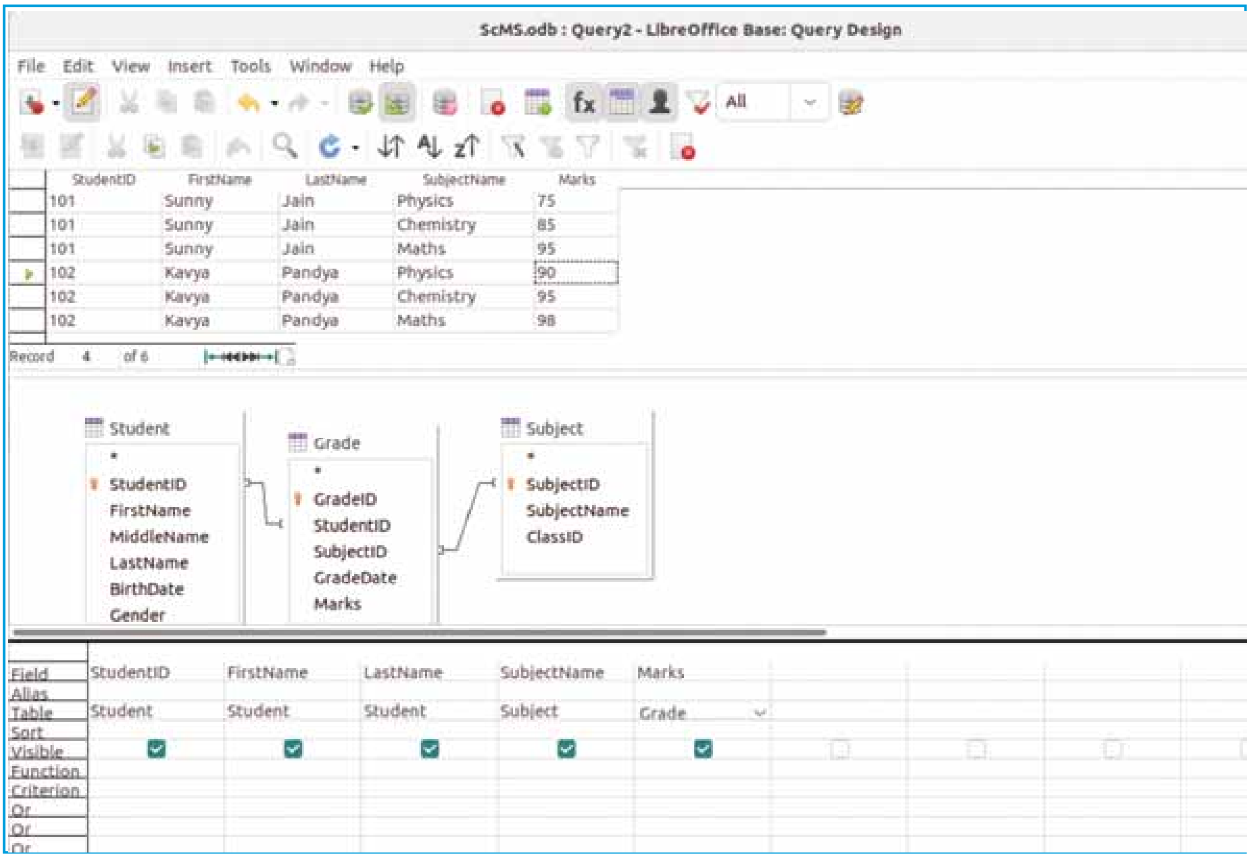
આકૃતિ 3.9 : Add ટેબલ ઓર ક્વેરી ડાયલોગ બોક્સ

હવે આપણે ડિઝાઈન વ્યૂનો ઉપયોગ કરીને એક ક્વેરી બનાવીશું. આકૃતિ 3.2માં બતાવેલ ક્વેરી ટેબમાં, *Create Query in Design View* પર ક્લિક કરો. આમ કરવાથી આકૃતિ 3.9માં બતાવ્યા મુજબ *Add Table or Query* ડાયલોગ બોક્ષ ખુલશે.

- *Student* ટેબલને સિલેક્ટ કરો અને *Add* બટન પર ક્લિક કરો.
- તે જ રીતે, *Grade* અને *Subject* ટેબલને સિલેક્ટ કરો. આકૃતિ 3.10માં બતાવ્યા પ્રમાણે આપણે ટેબલ પેનમાં ત્રણ ટેબલ જોઈ શકીએ છીએ. બેઝ રીલેશનશીપ પણ ડિસ્પ્લે કરે છે.
- *Add Table or Query* ને બંધ કરવા માટે *Close* બટન પર ક્લિક કરો.
- *Student* ટેબલમાંથી *StudentID, FirstName,* અને *LastName* પર ડબલ ક્લિક કરો. તે જ રીતે, *Subject* ટેબલમાંથી *SubjectName* ફીલ્ડ અને *Grade* ટેબલમાંથી *Marks* ફીલ્ડ પસંદ કરો. આકૃતિ 3.10માં બતાવ્યા પ્રમાણે ફીલ્ડના નામ તેમના સંબંધિત ટેબલના નામ સાથે ગ્રીડમાં ડિસ્પ્લે થશે.
- નોંધ લો કે આપણે *Alias, Sort, Visible, Function, Criterion* અને *Or* જેવા કેટલાક રેકોર્ડ (રો) શીર્ષકો પણ જોઈ શકીએ છીએ.મૂળભૂત રીતે દરેક ફીલ્ડ માટે *Visible* ઓપ્શન ટૂ સેટ થયેલું હોય છે. તે સૂચવે છે કે તમામ સિલેક્ટ કરેલા ફીલ્ડ આઉટપુટમાં ડિસ્પ્લે થશે.
- અગાઉના વિભાગમાં ચર્ચા કર્યા મુજબ, ફીલ્ડ માટે અર્થપૂર્ણ નામ ડિસ્પ્લે કરવા માટે ઉપનામ *(Alias)*નો ઉપયોગ કરી શકાય છે. ઉદાહરણ તરીકે, *Firstname*ને બદલે, આપણે ક્વેરીના પરિણામમાં કોલમ ટાઈટલ તરીકે *Name of Student* નો ઉપયોગ કરવાનું પસંદ કરીશું. *FirstName* કોલમ હેઠળ *Alias* રો શીર્ષક પછી દેખાતા ટેક્સ્ટ બોક્સમાં *Name of Student* ટાઈપ કરો.
- સોર્ટીંગ આપણને વેલ્યુના ચડતા અથવા ઉતરતા ક્રમમાં રેકોર્ડ ડિસ્પ્લે કરવાની સવલત આપે છે. વિદ્યાર્થીઓના રેકોર્ડને નામોના મૂળાક્ષર ક્રમમાં ડિસ્પ્લે કરવા માટે, *FirstName* કોલમ હેઠળના *Sort* ઓપ્શનમાં *ascending* પસંદ કરો. તે જ રીતે, *StudentID* નંબરના ચડતા ક્રમમાં રેકર્ડને ડિસ્પ્લે કરવા માટે, *StudentID* કોલમ હેઠળ *Sort* રો હેડીંગ પછી દેખાતા ડ્રોપ ડાઉન બોક્ષમાંથી *ascending* પસંદ કરો.
- *Query Design* ટૂલબાર પરના *Run Query* બટન પર ક્લિક કરો. આકૃતિ 3.11 માં બતાવ્યા મુજબનું ક્વેરીનું પરિણામ ડિસ્પ્લે થશે.
- ક્વેરીને સેવ કરવા માટે, *File* મેનુમાંથી *Save* ઓપ્શન પસંદ કરો. વૈકલ્પિક રીતે, *Close* બટન પર ક્લિક કરો એટલે બેઝ એક *Save* ડાયલોગ બોક્ષ ડિસ્પ્લે કરશે.
- ઈચ્છિત નામ ટાઈપ કરો, ઉદાહરણ તરીકે *StudentMarks,* અને *OK* બટન પર ક્લિક કરો.



આકૃતિ 3.10 : ફીલ્ડની પસંદગી



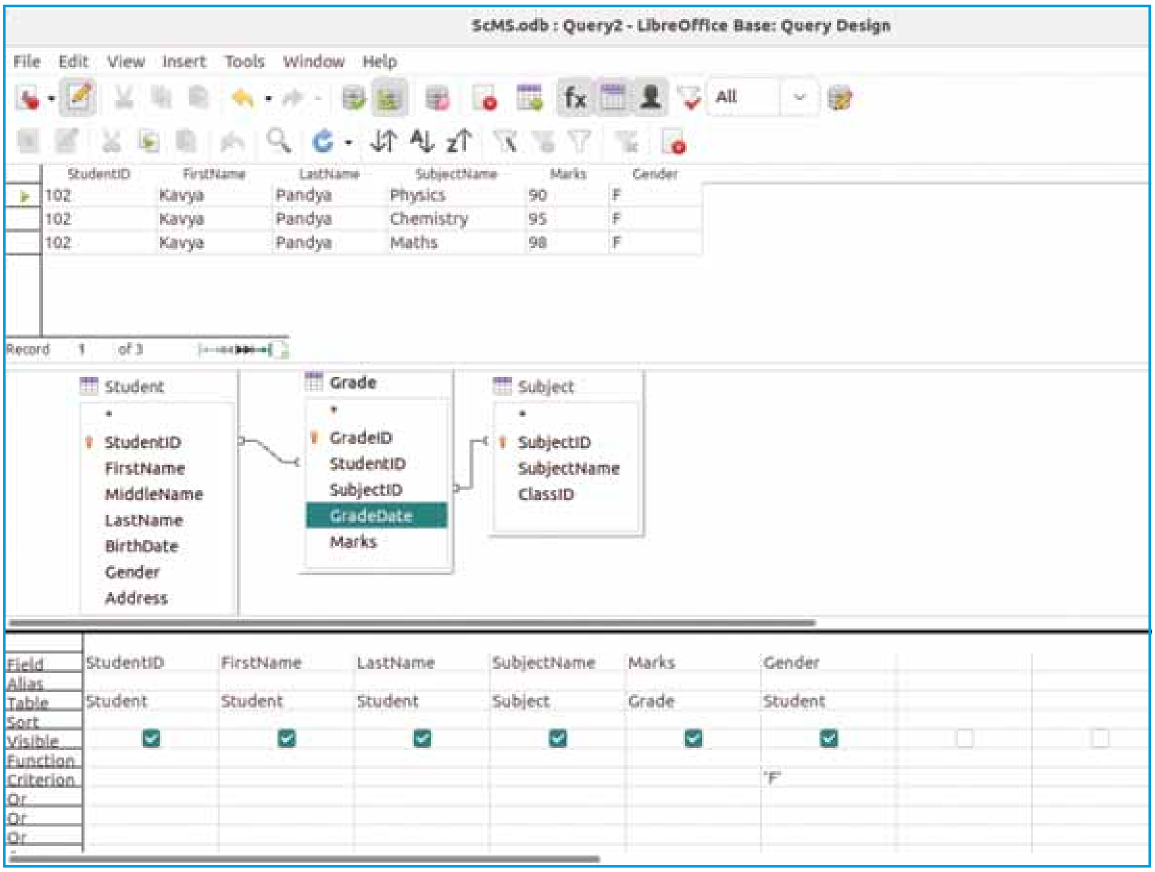
આકૃતિ 3.11 : ક્વેરીનો આઉટપુટ

માપદંડ સાથેની ક્વેરી (Queries with Criteria)

આપણે જોયું છે કે આપણે એવી ક્વેરી લખી શકીએ છીએ, જે ટેબલના પસંદ કરેલા ફીલ્ડને ડિસ્પ્લે કરે છે. હવે, ધારો કે, તમામ રેકોર્ડ જોવાની જગ્યાએ, આપણે એવા રેકોર્ડ જોવા માંગીએ છીએ જે અમુક ચોક્કસ માપદંડને પૂર્ણ કરે છે. ઉદાહરણ તરીકે, માત્ર સ્ત્રી વિદ્યાર્થી (ફીમેલ સ્ટુડન્ટ)ની વિગતો. આનો અર્થ એ છે કે આપણે સિલેક્ટ કરેલા રેકોર્ડનું એક પેટા-જૂથ ડિસ્પ્લે કરવા માંગીએ છીએ. આ માટે, આપણે એક માપદંડ નક્કી કરી શકીએ છીએ જે રેકોર્ડને માત્ર *Gender* ફીલ્ડમાં *'F'* મૂલ્ય ધરાવતા રેકોર્ડ પૂરતા મર્યાદિત કરશે.

એક ફીલ્ડનો ઉપયોગ (Using Single Field)

Gender ફીલ્ડના *Criterion* સેલમાં આકૃતિ 3.12માં બતાવ્યા મુજબ *'F'* ટાઈપ કરો. નોંધ લો કે, ક્રાઈટેરિયા ટેક્સ્ટને ક્વોટેશન ('') ડેલીમીટરની અંદર લખવું આવશ્યક છે, જ્યારે નંબરને કોઈપણ ડેલીમીટરની જરૂર નથી. જો આપણે ડેલીમીટર ન મૂકીએ, તો બેઝ કોઈ ભૂલ દેખાડશે નહીં; તેના બદલે તે જાતે જ ડેલીમીટર લાગુ કરશે. અગાઉ ચર્ચા કર્યા મુજબ ક્વેરીને *Save* અને રન કરીએ એટલે આપણને આઉટપુટ તરીકે સ્ત્રી વિદ્યાર્થીની યાદી જોવા મળશે.

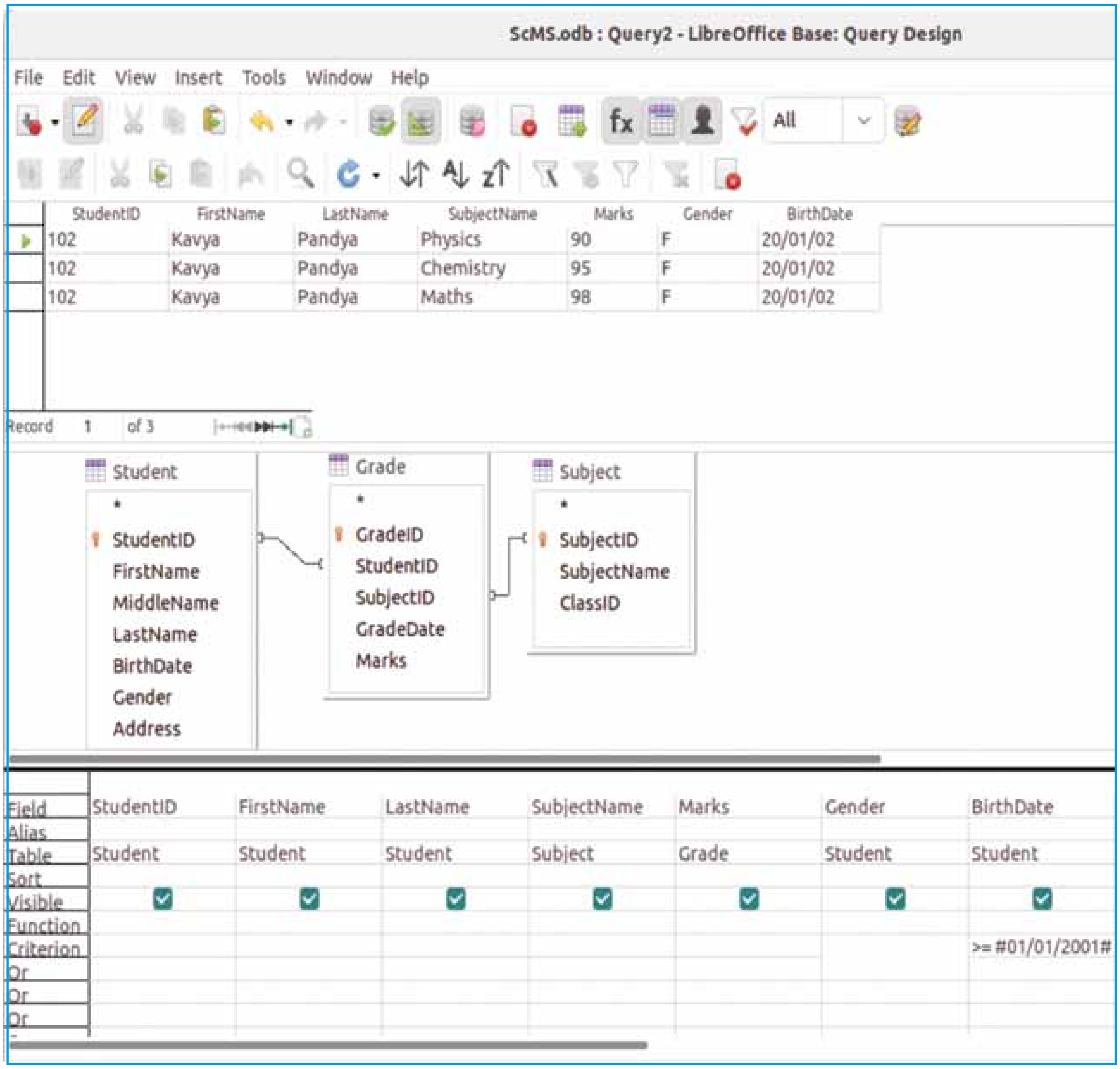


આકૃતિ 3.12 : માપદંડ સેટ કરવા

લોજીકલ ઓપરેટર (Logical Operators) નો ઉપયોગ કરવો

આકૃતિ 3.12માં દર્શાવેલ કોન્સટન્ટ વેલ્યુના ઉપયોગની ઉપરાંત, બેઝ આપણને વિવિધ લોજીકલ ઓપરેટરના આધારે ક્રાઈટેરિયા નક્કી કરવા માટેની પણ મંજૂરી આપે છે, જેમ કે, જે વિદ્યાર્થીઓના ગ્રેડ 60% થી વધુ હોય તેમના નામો પ્રિન્ટ કરવા.

ધારો કે આપણે એવા વિદ્યાર્થીઓની યાદી ડિસ્પ્લે કરવા માંગીએ છીએ કે જેમનો જન્મ 1st જાન્યુઆરી 2001 પછી થયો હોય. *Design View* માં એક નવી ક્વેરી બનાવો. *Student* ટેબલ ઉમેરો. *BirthDate* ફીલ્ડના Criterion સેલમાં "> #01/01/2001#" ટાઈપ કરો. હવે, ક્વેરી ને સેવ કરો અને રન કરો. આઉટપુટ આકૃતિ 3.13 જેવું જ દેખાશે.

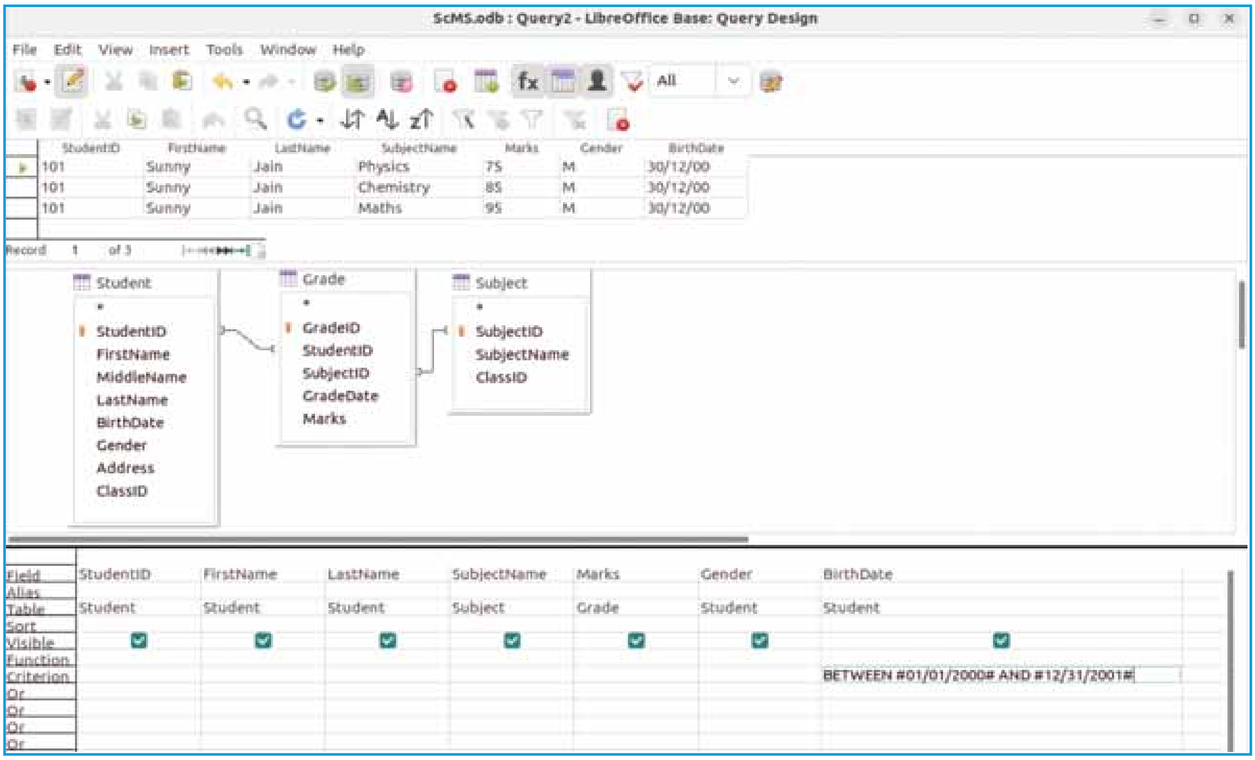


આકૃતિ 3.13 : Date ફીલ્ડમાં ક્રાઈટેરિયા લાગુ કરવો

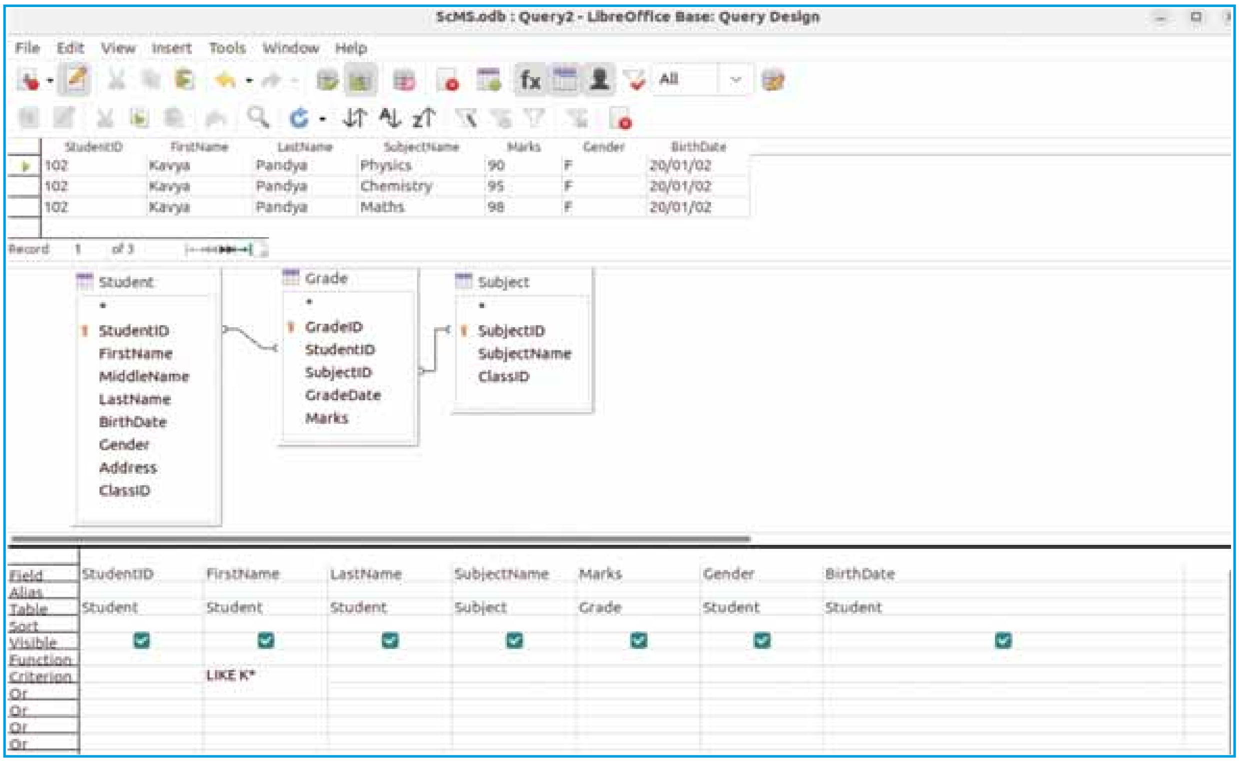
તે જ રીતે, 1st જાન્યુઆરી 2000 અને 31st ડિસેમ્બર 2001ની વચ્ચે આવતા વિદ્યાર્થીઓને ડિસ્પ્લે કરવા માટે, *BirthDate* ફીલ્ડના Criterion ને *">= # 01/01/2000 # AND <= # 12/31/2001 #"*. તરીકે સેટ કરી શકાય છે. આકૃતિ 3.14માં બતાવ્યા પ્રમાણે, આ જ ક્રાઈટેરિયા સેટ કરવા માટે બેઝ *BETWEEN* ઓપરેટર પણ આપે છે.

વાઈલ્ડ કાર્ડ (Wild Card)નો ઉપયોગ

ધારો કે આપણે એવા વિદ્યાર્થીઓની યાદી જોવા માંગીએ છીએ જેમનું પ્રથમ નામ *K* મૂળાક્ષરથી શરૂ થતું હોય. આ કરવા માટે, Student ટેબલનો ઉપયોગ કરીને એક નવી ક્વેરી બનાવો. આકૃતિ 3.15માં બતાવ્યા મુજબ *Criterion* ને *LIKE 'K*'* તરીકે સેટ કરો. આકૃતિ 3.15માં *FirstName* ફીલ્ડના *Criterion* સેલમાં એક્સપ્રેશનમાં ઉપયોગમાં લેવાયેલ એસ્ટેરિસ્ક (*) ને વાઈલ્ડકાર્ડ કેરેક્ટર તરીકે ઓળખવામાં આવે છે. વાઈલ્ડકાર્ડ એક સિમ્બોલ છે કે જે કોઈપણ એક અક્ષર અથવા અક્ષરના સમૂહનું પ્રતિનિધિત્વ કરે છે. *'K*'* એવા શબ્દનું પ્રતિનિધિત્વ કરે છે જેનો પ્રથમ અક્ષર K છે, અને ત્યારબાદ કોઈપણ અક્ષરોનો સમૂહ હોઈ શકે છે. તે જ રીતે, *LIKE '*k'* ક્રાઈટેરીયા એવા વિદ્યાર્થીઓને ડિસ્પ્લે કરશે જેના નામો 'k' અક્ષરથી સમાપ્ત થતા હોય. આવી જ રીતે આપણે એક અક્ષર માટે ? વાઈલ્ડકાર્ડ કેરેક્ટરનો ઉપયોગ કરી શકીએ.



આકૃતિ 3.14 : Between ઑપરેટરનો ઉપયોગ



આકૃતિ 3.15 : વાઈલ્ડ કાર્ડ્સ

પેરામીટર (Parameters) સાથેની ક્વેરી

અત્યાર સુધી, આપણે જે ક્વેરી બનાવી છે તેમાં નિશ્ચિત માપદંડનો ઉપયોગ થતો હતો. એકવાર આપેલ માપદંડ ક્વેરીના દરેક અમલ (એક્ઝિક્યુશન) વખતે બદલાશે નહીં. જોકે, ક્વેરીનું આઉટપુટ ટેબલમાં રહેલ ડેટાના આધારે બદલાઈ શકે છે.

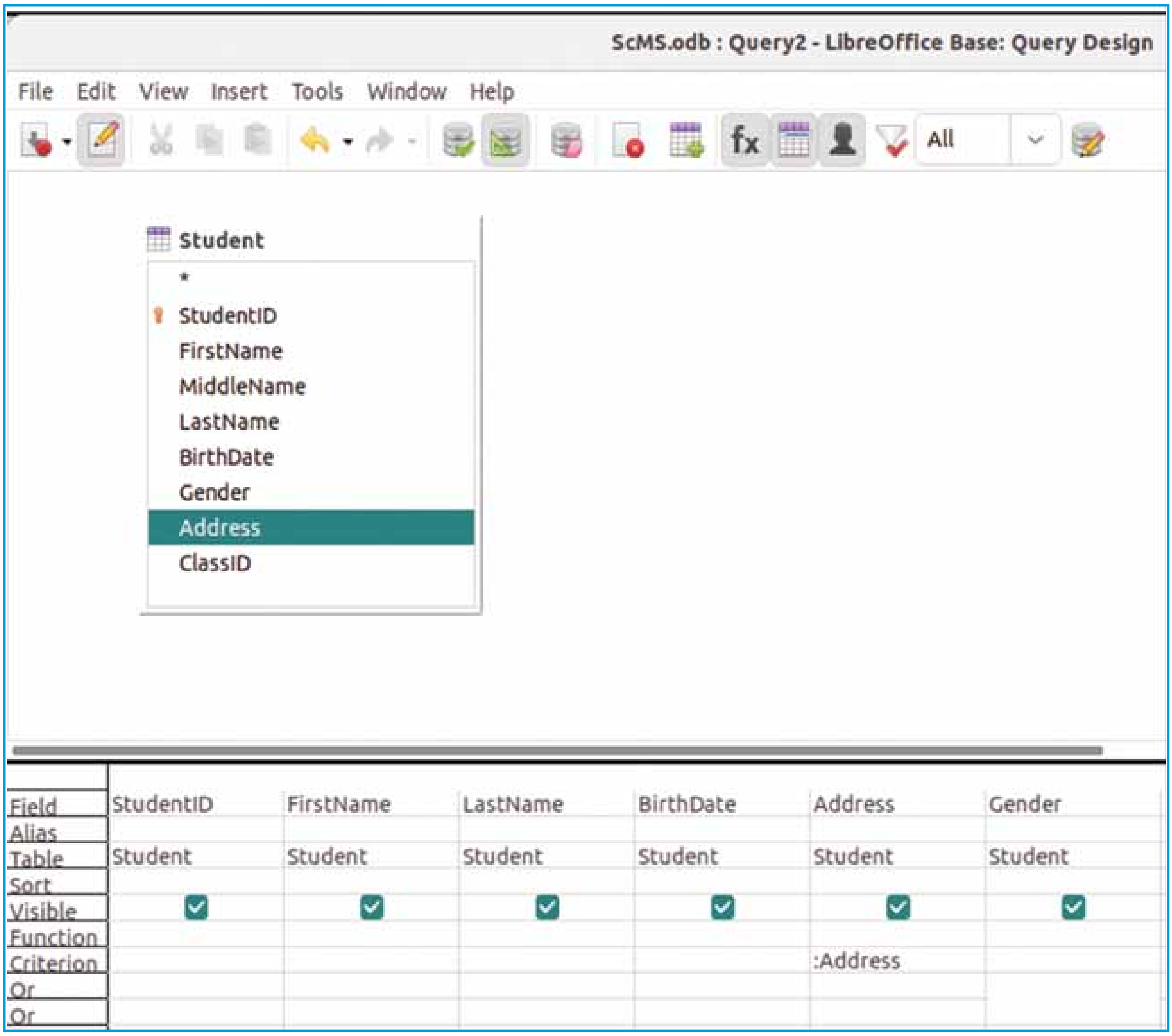
પેરામીટર ક્વેરીને રન ટાઈમ યુઝર પાસેથી વેલ્યુ સ્વીકારવા માટે ડિઝાઈન કરવામાં આવે છે. સામાન્ય રીતે, જ્યારે આપણે પેરામીટર ક્વેરી રન કરીએ છીએ, ત્યારે તે યુઝરને પેરામીટરની વેલ્યુ એન્ટર કરવા માટે એક ડાયલોગ બોક્ષ ડિસ્પ્લે કરશે. આ વેલ્યુ પછી ડેટા પુનઃપ્રાપ્ત કરવા માટે માપદંડની (ક્રાઈટેરીયન) વેલ્યુ તરીકે એસાઈન કરવામાં આવે છે.

ચાલો, આપણે Student ટેબલમાં ઉપલબ્ધ વિદ્યાર્થીઓની વિગતો દર્શાવવા માટે એક ક્વેરી બનાવીએ. નીચે આપેલા પગલાંનો ઉપયોગ કરવાથી આપણને ઇચ્છિત પરિણામ મળશે.

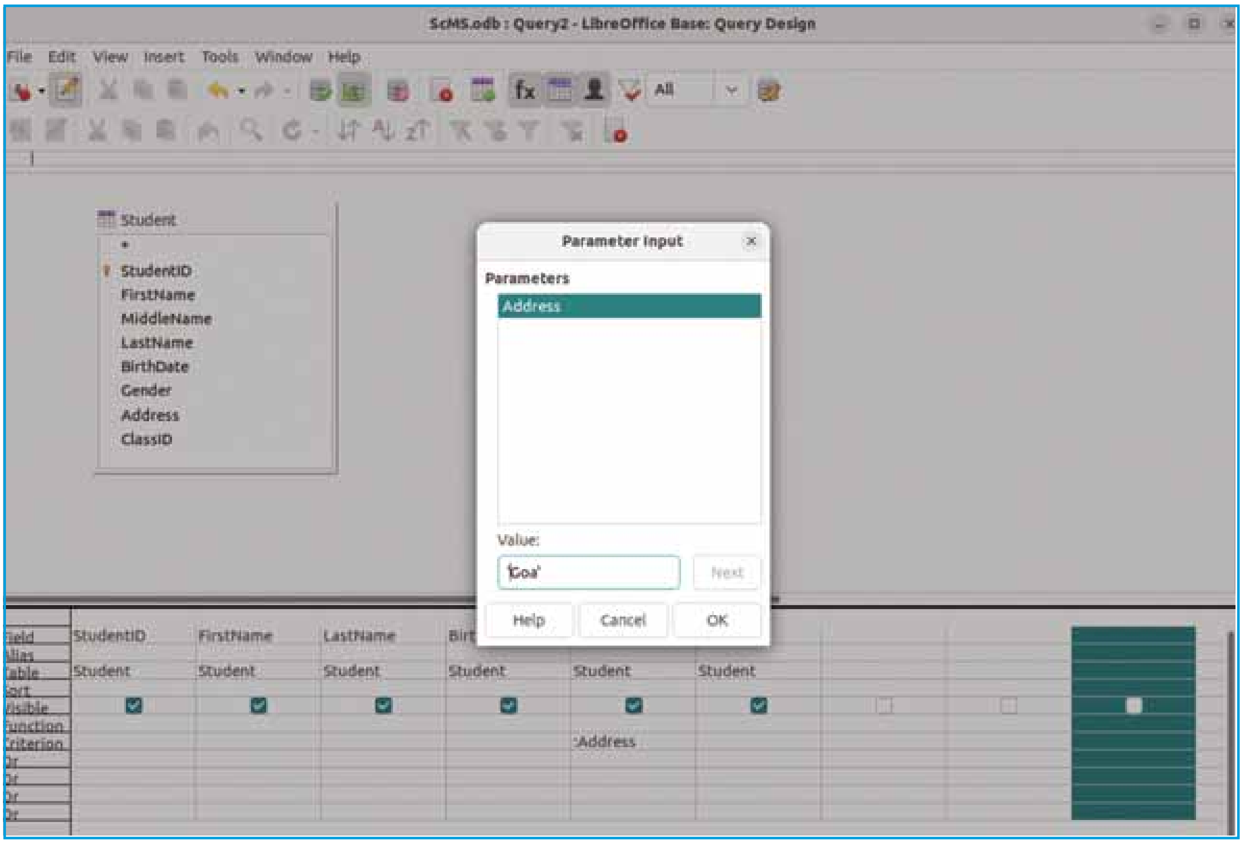
- નવી ક્વેરી ડિઝાઈન વ્યૂમાં ખોલો.
- Student ટેબલ ઉમેરો.
- Address ફીલ્ડના Criterion સેલમાં : *Address* અથવા [Address] ટાઈપ કરો.

ક્વેરી આકૃતિ 3.16માં દર્શાવ્યા પ્રમાણે દેખાશે. નોંધ લો કે માપદંડ માટેના પેરામીટરની પહેલાં કોલન સિમ્બોલ (:) હોવો જરૂરી છે.

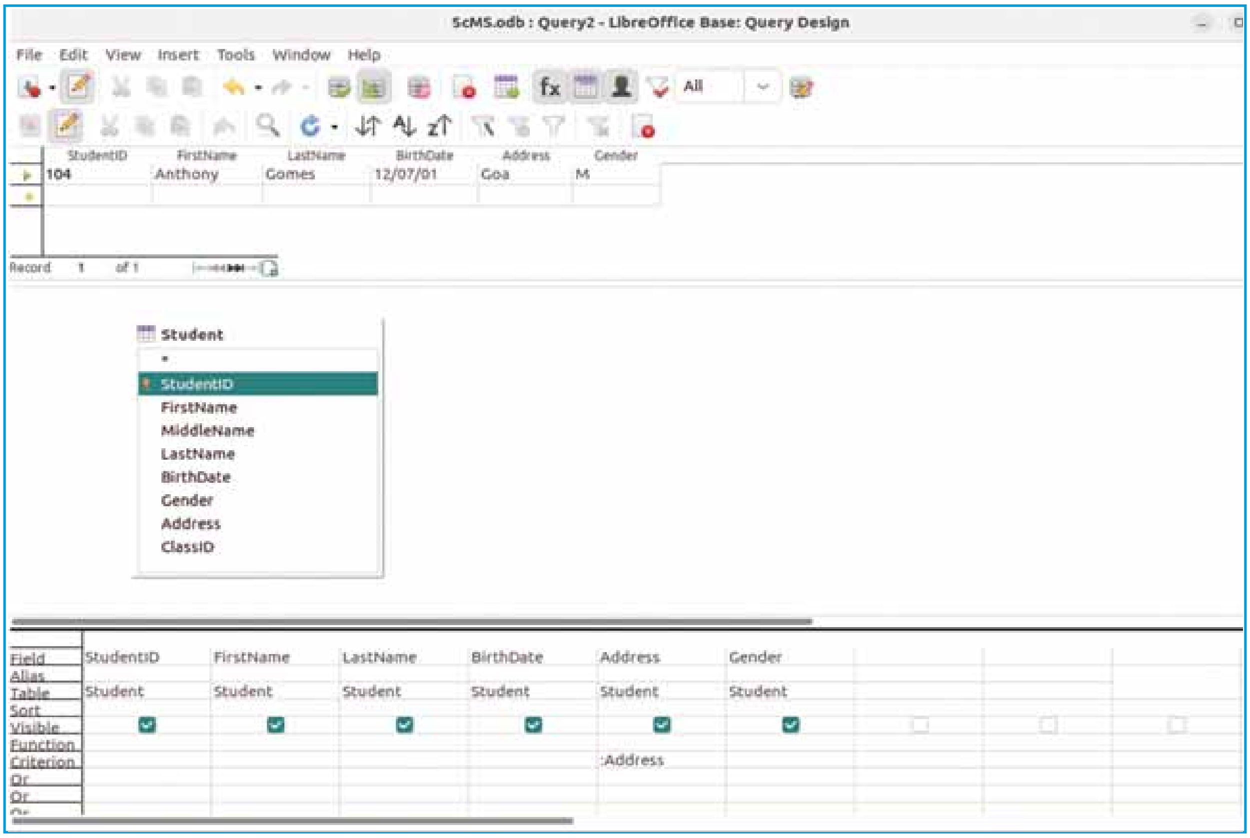
- ક્વેરીનું પરિણામ જોવા માટે *Run* બટન પર ક્લિક કરો. બેઝ આકૃતિ 3.17માં દર્શાવ્યા પ્રમાણે એક ડાયલોગ બોક્ષ ડિસ્પ્લે કરશે.
- *Value* લેબલની નીચે આપેલા ટેક્સ્ટબોક્સમાં *'Goa'* ટાઈપ કરો અને *OK* પર ક્લિક કરો. આકૃતિ 3.18માં દર્શાવ્યા પ્રમાણે આપણને *'Goa'* માં રહેતા વિદ્યાર્થીઓની યાદી મળશે.



આકૃતિ 3.16 : પેરામીટરવાળી ક્વેરી



આકૃતિ 3.17 : પેરામીટર વેલ્યુ



આકૃતિ 3.18 : આઉટપુટ

સારાંશ

એક ડેટાબેઝ સિસ્ટમ મોટા પ્રમાણમાં ડેટા સંગ્રહિત કરવાની એક પદ્ધતિ પૂરી પાડે છે. ડેટાબેઝ સિસ્ટમમાંથી અંતિમ યુઝર માટે ઉપયોગી ડેટા પુનઃપ્રાપ્ત કરવો એક પડકારજનક કાર્ય છે. ડેટાબેઝ ક્વેરી યુઝરને ડેટાબેઝમાંથી અર્થપૂર્ણ માહિતી પુનઃપ્રાપ્ત કરવાની મંજૂરી આપે છે. SQL એ ડેટાબેઝ ટેબલમાંથી ડેટા પુનઃપ્રાપ્ત કરવાની એક ભાષા છે. SQLનો ઉપયોગ કરીને, આપણે એક કરતાં વધુ ટેબલમાંથી પણ ડેટા પુનઃપ્રાપ્ત કરી શકીએ છીએ. લિબ્રેઓફીસ બેઝ ડેટાબેઝ ક્વેરી લખવા માટે યુઝર-ફ્રેન્ડલી ઇન્ટરફેસ પ્રદાન કરે છે. ક્વેરી વિઝાર્ડ વ્યૂ ડેટાબેઝ ક્વેરી બનાવવા માટે સ્ટેપ-બાય-સ્ટેપ માર્ગદર્શિત અભિગમ પૂરો પાડે છે. બીજી બાજુ, ડિઝાઇન વ્યૂ વધુ જટિલ ક્વેરી લખવા માટે એક કાર્યક્ષમ પદ્ધતિ પૂરી પાડે છે, જેના દ્વારા ઉપયોગી માહિતી પુનઃપ્રાપ્ત કરવા માટે મલ્ટીપલ ટેબલને જોડી શકાય છે.

સ્વાધ્યાય

1. લિબ્રેઓફીસ બેઝમાં ક્વેરીનો ઉપયોગ કરીને ટેબલમાંથી ચોક્કસ રેકોર્ડ પુનઃપ્રાપ્ત કરવાની પ્રક્રિયા સમજાવો.

2. મલ્ટીપલ ટેબલમાંથી ડેટા કેવી રીતે પુનઃપ્રાપ્ત કરી શકાય તે વર્ણવો?

3. ડિઝાઇન વ્યૂનો ઉપયોગ કરીને ક્વેરી કેવી રીતે બનાવી શકાય તે સમજાવો.

4. લિબ્રેઓફીસ બેઝમાં ક્વેરીનો હેતુ શું છે?

5. સિમ્પલ ક્વેરી અને પેરામીટર ક્વેરી વચ્ચેનો તફાવત સમજાવો.

6. લિબ્રેઓફીસ બેઝમાં કયો વ્યૂ ક્વેરીને વિઝ્યુઅલ રીતે બનાવવા માટે GUI પ્રદાન કરે છે?

7. Students નામના ટેબલમાંથી બધા ફીલ્ડ સિલેક્ટ કરવા માટેનું SQL સ્ટેટમેન્ટ લખો.

8. આ ક્વેરીનું આઉટપુટ શું છે?

SELECT "Name", "Age" FROM "Employees" WHERE "Age" > 30;

9. ક્વેરીમાં વાઇલ્ડકાર્ડનો ઉપયોગ શું છે?

10. ક્વેરી લખવામાં ક્રાઈટેરિયાનો ઉપયોગ શું છે?

11. સાચું કે ખોટું જણાવો.

(1) લિબ્રેઓફીસ બેઝમાં ક્વેરી ફક્ત એક જ ટેબલ માંથી ડેટા પુનઃપ્રાપ્ત કરી શકે છે.

(2) આપણે લિબ્રેઓફીસ બેઝમાં ડેટા પુનઃપ્રાપ્ત કરવા માટે SQL સ્ટેટમેન્ટ્સનો ઉપયોગ કરી શકીએ છીએ.

(3) પેરામીટર ક્વેરી યુઝરને રનટાઇમમાં વેલ્યુ એન્ટર કરીને ડેટાને ફિલ્ટર કરવાની મંજૂરી આપે છે.

(4) ક્વેરીમાં ડેટા પુનઃપ્રાપ્ત કરતી વખતે પેટર્ન મેચિંગ માટે % જેવા વાઇલ્ડકાર્ડ કેરેક્ટરનો ઉપયોગ કરી શકાય છે.

(5) લિબ્રેઓફીસ બેઝમાં ક્વેરીને સેવ કરી શકાય છે અને તેનો ઉપયોગ વારંવાર કરી શકાય છે.

12. ખાલી જગ્યા પૂરો.

(1) લિબ્રેઓફીસ બેઝમાં એક અથવા વધુ ટેબલમાંથી ચોક્કસ માહિતી પ્રાપ્ત કરવા માટે ____________ નો ઉપયોગ થાય છે.

(2) લિબ્રેઓફીસ બેઝમાં ____________ વ્યૂ યુઝરને SQL કોડ લખ્યા વિના વિઝ્યુઅલ રીતે ક્વેરી બનાવવાની સવલત આપે છે.

(3) પેરામીટર ક્વેરીમાં, યુઝર ઇનપુટ ____________ કૌંસનો ઉપયોગ કરીને પૂછવામાં આવે છે.

(4) SQLમાં કોઈપણ અક્ષરોના ક્રમને મેચ કરવા માટે વાઇલ્ડકાર્ડ કેરેક્ટર ____________ નો ઉપયોગ થાય છે.

(5) ક્વેરીના પરિણામો ____________ની જેમ ટેબલ ફોર્મેટમાં ડિસ્પ્લે કરી શકાય છે.

13. બહુવિકલ્પી પ્રશ્નો. સૌથી યોગ્ય જવાબ પસંદ કરો.

(1) ડેટાબેઝમાંથી ચોક્કસ માહિતી પુનઃપ્રાપ્ત કરવા માટે લિબ્રેઓફિસ બેઝમાં કયા ઑબ્જેક્ટનો ઉપયોગ થાય છે?

(a) ટેબલ (Table) (b) ફોર્મ (Form) (c) ક્વેરી (Query) (d) રીપોર્ટ (Report)

(2) SQL ક્વેરીમાં * સિમ્બોલ શું દર્શાવે છે?

(a) તમામ રો (b) તમામ કોલમ (c) તમામ ટેબલ (d) પ્રાઈમરી કી

(3) લિબ્રેઓફિસ બેઝમાં નીચેનામાંથી કયો વ્યૂ ક્વેરીને ગ્રાફિકલ રીતે બનાવવાની સવલત આપે છે?

(a) SQL વ્યૂ (b) ફોર્મ વ્યૂ (c) ડીઝાઈન વ્યૂ (d) ટેબલ વ્યૂ

(4) પેરામીટર ક્વેરીમાં, યુઝરને ઇનપુટ માટે પ્રોમ્પ્ટ કરવા માટે શું વપરાય છે?

(a) છગડિયો કૌંસ {} (b) કૌંસ ()

(c) મોટો કૌંસ [] (d) અવતરણ ચિહ્ન " "

(5) લિબ્રેઓફિસ બેઝમાં ડેટા પુનઃપ્રાપ્તિ માટે નીચેનામાંથી કોનો ઉપયોગ થાય છે?

(a) ફોર્મ (b) ક્વેરી (c) રીલેશન (d) ટેબલ ડીઝાઈન

(6) લિબ્રેઓફિસ બેઝમાં કયા વાઇલ્ડકાર્ડ કેરેક્ટર અક્ષરોના કોઈપણ ક્રમ સાથે મેચ થાય છે?

(a) – (b) # (c) * (d) ?

(7) ડેટાબેઝમાંથી ડેટા મેળવવા માટે કયા SQL કમાન્ડનો ઉપયોગ થાય છે?

(a) UPDATE (b) INSERT (c) DELETE (d) SELECT

(8) 'A' અક્ષરથી શરૂ થતા નામો મેળવવા માટે, નીચેનામાંથી શેનો ઉપયોગ કરવામાં આવશે?

(a) LIKE 'A_' (b) LIKE '_A%' (c) LIKE 'A*' (d) LIKE '%A'

(9) SELECT "City" FROM "Customers" WHERE "Country" = 'India' આ ક્વેરી શું પરિણામ આપશે?

(a) બધા દેશોના શહેરો (b) ભારતમાં રહેલા ગ્રાહકોના શહેરો

(c) બધા ગ્રાહકોના દેશો (d) ભારતમાં રહેલા ગ્રાહકોના નામો

(10) જો ક્વેરી કોઈ રો પરિણામ સ્વરૂપે ડિસ્પ્લે ન કરે તો લિબ્રેઓફીસ બેઝ શું ડિસ્પ્લે કરશે?

(a) એરેર મેસેજ (b) NULL

(c) ખાલી પરિણામની ગ્રીડ (d) 0

પ્રાયોગિક સ્વાધ્યાય

1. નીચેના ટેબલમાં બતાવ્યા પ્રમાણે BookList નામનું ડેટાબેઝ ટેબલ બનાવો.

BookID	BookTitle	Author	Genre	Ratings
1	The Hobbit	J. R. R. Talkien	Adventure	4
2	The Adventures of Sherlock Holmes	Sir Arthur Conan Doyle	Suspense	5
3	Oliver Twist	Charles Dickens	Drama	3
4	Adventures of huckleberry finn	Mark Twain	Children Fiction	4

2. ક્વેરી વિઝાર્ડનો ઉપયોગ કરીને, સ્વાધ્યાય 1 માં બનાવેલ ટેબલ BookListમાંથી મૂળાક્ષરોના ક્રમમાં ગોઠવાયેલા પુસ્તકના નામ મેળવવા માટે ક્વેરી બનાવો.

3. ડિઝાઇન વ્યૂનો ઉપયોગ કરીને, ઓછામાં ઓછા 4 રેટિંગ ધરાવતા પુસ્તકોની યાદી ડિસ્પ્લે કરવા માટે ક્વેરી બનાવો.

4. "Genre" પેરામીટર સાથે ક્વેરી બનાવો, જેથી યુઝર આ પ્રકાર માટેના પુસ્તકો શોધી શકે.

ફોર્મ અને રિપોર્ટ સાથે કાર્ય

પરિચય

અગાઉના પ્રકરણમાં આપણે ડેટાબેઝ ડિઝાઇન અને ટેબલ ડેટાશીટ વ્યૂ (Table Datasheet View)નો ઉપયોગ કરીને મૂળભૂત ટેબલ ઑપરેશન તેમજ ક્વેરી ડેટાશીટ વ્યૂ (Query Datasheet View) દ્વારા માહિતીની પુનઃપ્રાપ્તિ (information retrieval) વિશે ચર્ચા કરી. આ પદ્ધતિઓ અસરકારક છે, પણ ડેટાશીટ વ્યૂનું પ્રમાણભૂત રો અને કૉલમ સ્વરૂપ - જેમાં સ્પ્રેડશીટ જેવી ડેટા એન્ટ્રી હોય છે - તે ઉપયોગકર્તાની દૃષ્ટિએ આકર્ષક નથી અને નીરસ લાગી શકે છે. આ 'બ્લેક એન્ડ વ્હાઇટ' પ્રેઝન્ટેશન ઘણીવાર ડેટા સાથેની ક્રિયા-પ્રતિક્રિયાને અરસિક અને કંટાળાજનક બનાવી દે છે. ઉપરાંત, ડેટાબેઝ મેનેજમેન્ટ સિસ્ટમની નામકરણ પદ્ધતિઓ (naming conventions)ને કારણે આપણે ઘણીવાર ટેબલ ડિઝાઇન કરતી વખતે શોર્ટ ફોર્મ્સ (initials) અથવા સંક્ષિપ્ત નામો (abbreviated names)નો ઉપયોગ કરીએ છીએ. જોકે, આ સંક્ષિપ્ત નામ ધરાવતા ફિલ્ડ કેટલીકવાર અસ્પષ્ટ હોય છે અને હંમેશા આપમેળે સમજી શકાય તેવા હોતા નથી.

આ પ્રકરણ ડેટાબેઝની માહિતી દાખલ કરવા અને પ્રદર્શિત કરવા માટે વૈકલ્પિક અને વધારે યુઝર-ફ્રેન્ડલી અભિગમ રજૂ કરે છેઃ ફોર્મ્સ (Forms) અને રિપોર્ટ્સ (Reports). આપણે જોઈશું કે રિપોર્ટ્સનો ઉપયોગ કરીને ડેટાને કેવી રીતે ફોર્મેટ કરેલી (formatted) અને સાહજિક (intuitive) રીતે રજૂ કરવો. ટેબલ અને ક્વેરીની જેમ જ, ફોર્મ અને રિપોર્ટ પણ ઑબ્જેક્ટ્સ છે જે ડેટાબેઝ વિન્ડોની ડાબી બાજુના વિભાગમાં દેખાય છે.

ફોર્મ (Forms)

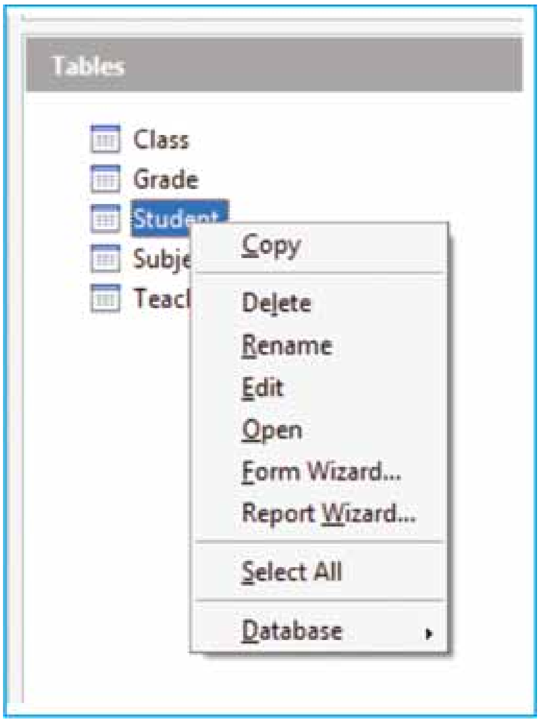
તમે સંભવતઃ ઘણાં ફોર્મનો ઉપયોગ કર્યો હશે, જેમ કે શાળામાં પ્રવેશ માટેનું ફોર્મ, ઓર્ડર ફોર્મ અથવા પૂછપરછ માટેનું ફોર્મ. આ ઉદાહરણો દર્શાવે છે કે ફોર્મ ઉપયોગકર્તા પાસેથી ડેટા એકત્રિત કરવાનું કેટલું સરળ બનાવે છે. ડેટાબેઝમાં ફોર્મ એક એવું ઘટક છે જે આપણને ટેબલમાં સંગ્રહિત ડેટા સાથે ક્રિયા-પ્રતિક્રિયા (interact) કરવાની પરવાનગી આપે છે. ફોર્મની કલ્પના ડેટાબેઝમાં એક યુઝર-ફ્રેન્ડલી વિન્ડો તરીકે કરી શકાય, જે ડેટાને નિહાળવાનું, દાખલ કરવાનું, સુધારવાનું અને કાઢી નાખવાનું ઘણું સરળ બનાવે છે. ટેબલમાં સીધો જ ફેરફાર કરવાને બદલે, ફોર્મ ડેટાને વધુ વ્યવસ્થિત, આકર્ષક અને સાહજિક રીતે રજૂ કરે છે. ફિલ્ડને ઘણીવાર સ્પષ્ટપણે લેબલ આપવામાં આવે છે, અને ફોર્મના લેઆઉટને વાસ્તવિક દુનિયાના દસ્તાવેજો અથવા પ્રક્રિયાઓ પ્રતિબિંબિત કરવા માટે ડિઝાઇન કરી શકાય છે. ડેટાબેઝની પરિભાષામાં, ફોર્મ ડેટા એન્ટ્રી અને એડિટિંગ માટે ફ્રન્ટ-એન્ડ તરીકે કાર્ય કરે છે. વિવિધ સ્ટાઈલ, રંગો, શીર્ષકો, નામ અને લોગો સાથે ફોર્મને ડિઝાઇન કરી શકાય છે, જે પ્રક્રિયાને ઘણી આકર્ષક બનાવે છે.

મોટાભાગના ફોર્મ એક અથવા વધુ ટેબલ પર આધારિત હોય છે. જ્યારે આપણે ફોર્મમાં ડેટા દાખલ કરીએ છીએ અથવા સુધારીએ છીએ, ત્યારે આ ફેરફારો સંબંધિત ટેબલમાં પ્રતિબિંબિત થાય છે. ફોર્મ ક્વેરી-આધારિત પણ હોઈ શકે છે. જ્યારે ડેટાના સબસેટને અથવા બહુવિધ ટેબલોમાંથી સંયુક્ત ડેટાને પ્રદર્શિત કરવા, દાખલ કરવા અથવા સુધારવા માંગતા હોઈએ ત્યારે આ પદ્ધતિ ઉપયોગી છે. ફોર્મ બનાવવાની બે મુખ્ય રીતો છેઃ *ફોર્મ વિઝાર્ડ (Form Wizard)*નો ઉપયોગ કરીને અને *ડિઝાઇન વ્યૂ (Design View)*નો ઉપયોગ કરીને.

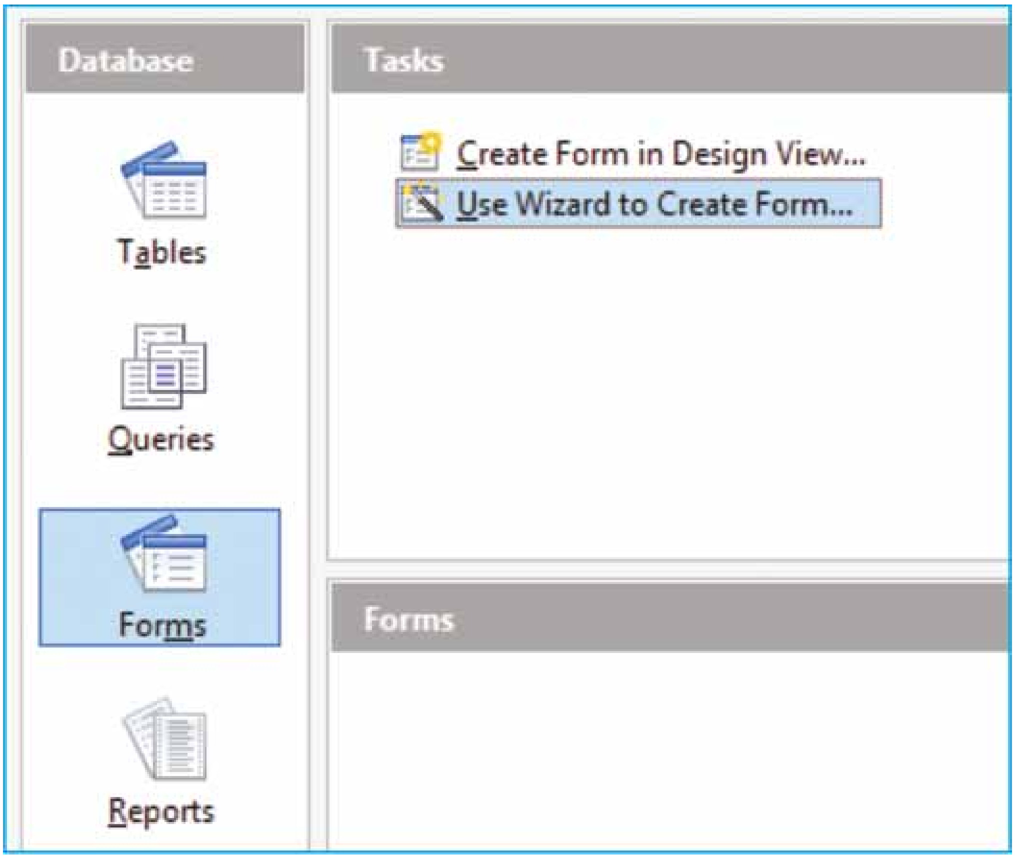
ફોર્મ વિઝાર્ડનો ઉપયોગ કરીને ફોર્મ બનાવવું સરળ અને ઝડપી છે. વિઝાર્ડ એકથી વધુ પગલામાં આપવામાં આવતા પ્રશ્નોની શ્રેણી દ્વારા માર્ગદર્શન આપીને, ફોર્મ બનાવનારનાં પ્રયત્નોમાં ઘટાડો કરીને, ફોર્મ-રચનાની મોટા

ભાગની પ્રક્રિયાને સ્વચાલિત કરે છે. ચાલો, આપણે ફોર્મ વિઝાર્ડનો ઉપયોગ કરીને અગાઉ બનાવેલા એક ટેબલ પર આધારિત ફોર્મ બનાવીએ. લિબ્રેઓફિસ બેઝ ખોલો અને પછી બીજા પ્રકરણમાં બનાવેલ ડેટાબેઝ ખોલો. હવે, Student ટેબલ પર આધારિત ફોર્મ બનાવવા માટે નીચેના પગલાં અનુસરો:

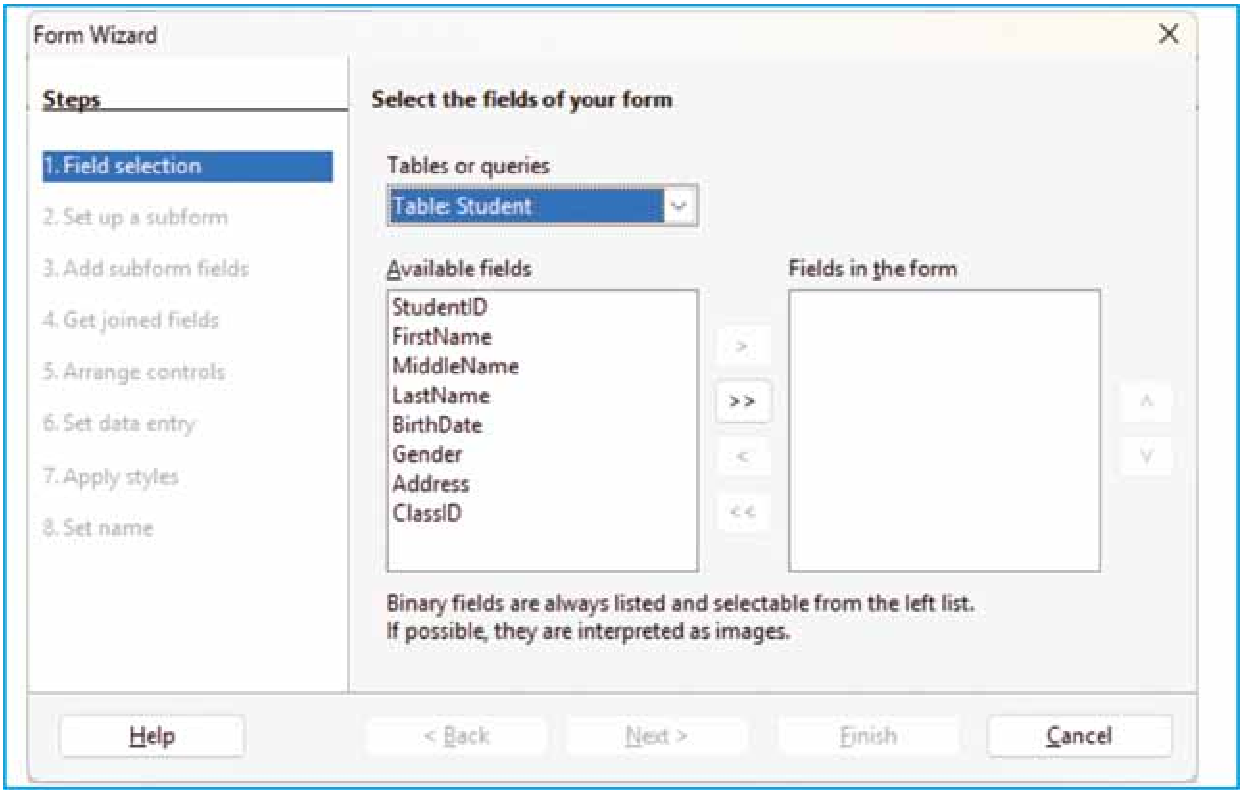
- Student ટેબલ પર રાઈટ ક્લિક કરો અને કન્ટેક્સ્ટ મેનૂમાંથી આકૃતિ 4.1માં દર્શાવ્યા પ્રમાણે *Form Wizard* વિકલ્પ પર ક્લિક કરો. આમ કરવાથી *Form Wizard* ખુલશે, જે ફોર્મ બનાવવા માટે આઠ પગલાં દર્શાવે છે.
- વૈકલ્પિક રીતે, બેઝ ઇન્ટરફેસની ડાબી બાજુ આવેલ પેનલમાંથી ફોર્મ વિભાગ ખોલી શકાય છે અને આકૃતિ 4.2માં બતાવ્યા પ્રમાણે *Use Wizard to create a Form* લિંક પર ક્લિક કરી શકાય છે.



આકૃતિ 4.1 : Form Wizard દર્શાવતું કંટેક્સ્ટ મેનૂ



આકૃતિ 4.2 : Tasks વિભાગ દ્વારા Form Wizardનો ઉપયોગ

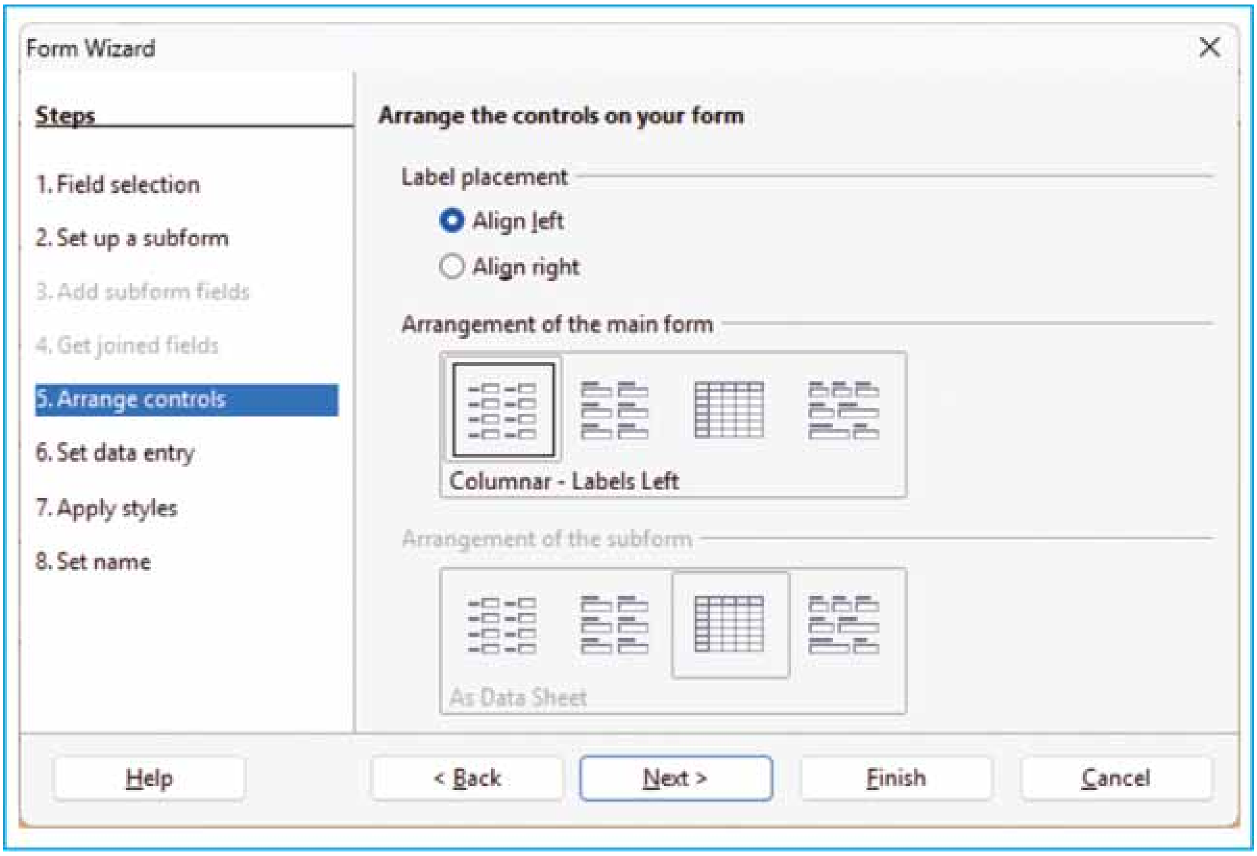


આકૃતિ 4.3 : ફોર્મ માટે ટેબલ અને ફિલ્ડની પસંદગી

- આકૃતિ 4.3માં દર્શાવ્યા મુજબ ફોર્મ વિઝાર્ડ રજૂ કરવામાં આવશે, જેમાં ફોર્મ બનાવવા માટેના આઠ પગલાંનું માર્ગદર્શન આપવામાં આવ્યું છે.
- સૌપ્રથમ, જે ટેબલ પરથી ફોર્મની રચના કરવાની છે તેને પસંદ કરવાની જરૂર પડશે. *Tables or Queries* ડ્રોપડાઉન મેનૂમાંથી *Student* ટેબલ પસંદ કરો. આમ કરવાથી *Student* ટેબલના તમામ ફિલ્ડ *Available Fields* યાદીમાં દેખાશે.
- ફોર્મમાં કોઈ ચોક્કસ ફિલ્ડ ઉમેરવા માટે *Available Fields* યાદીમાં તેનું નામ પસંદ કરો અને '>' બટન પર ક્લિક કરો. બેઝ પસંદ કરેલા ફિલ્ડને *Fields in the form* યાદીમાં ખસેડશે.
- જો ટેબલમાં આવેલ બધા જ ફિલ્ડનો સમાવેશ ફોર્મમાં કરવા માંગતા હોઈએ તો '>>' બટનનો ઉપયોગ કરી શકાય છે.
- ફોર્મમાંથી પસંદ કરેલા કોઈ એક ફિલ્ડને દૂર કરવા માટે '<' બટનનો ઉપયોગ કરી શકાય તથા *Fields in the form* યાદીમાંથી બધા ફિલ્ડ દૂર કરવા માટે '<<' બટન પર ક્લિક કરી શકાય છે.
- પસંદ કરેલ ફિલ્ડનો ક્રમ પણ બદલી શકાય છે. *Fields in the form* યાદીમાં ફિલ્ડને પસંદગીના ક્રમમાં ગોઠવવા માટે *Up* અને *Down* એરો બટનનો ઉપયોગ કરવામાં આવે છે.

એકવાર ટેબલ અને ફિલ્ડ પસંદ કરવામાં આવે ત્યાર પછી ફોર્મ વિઝાર્ડના બીજા પગલા પર જવા માટે Next બટન પર ક્લિક કરો.

- ફોર્મ વિઝાર્ડનું બીજું પગલું સબફોર્મ બનાવવાની સુવિધા આપે છે. પરંતુ હાલ પૂરતું આપણે તે છોડી દઈશું. આકૃતિ 4.4માં દર્શાવ્યા મુજબ સીધા પાંચમા પગલા પર આગળ વધવા માટે *Next* પર ક્લિક કરો.



આકૃતિ 4.4 : ફોર્મમાં કંટ્રોલની ગોઠવણી

- ફોર્મ વિઝાર્ડના પાંચમા પગલામાં મુખ્ય ફોર્મની લેઆઉટ (ગોઠવણી) પસંદ કરવા પર ધ્યાન કેન્દ્રિત કરીશું. આ પગલું આપણે પસંદ કરેલા ફિલ્ડ ફોર્મ પર કેવી રીતે પ્રદર્શિત થશે તેના માટે વિવિધ વિકલ્પો રજૂ કરે છે. આ માટે સામાન્ય રીતે ઉપલબ્ધ લેઆઉટ્સ આ પ્રમાણે છે:

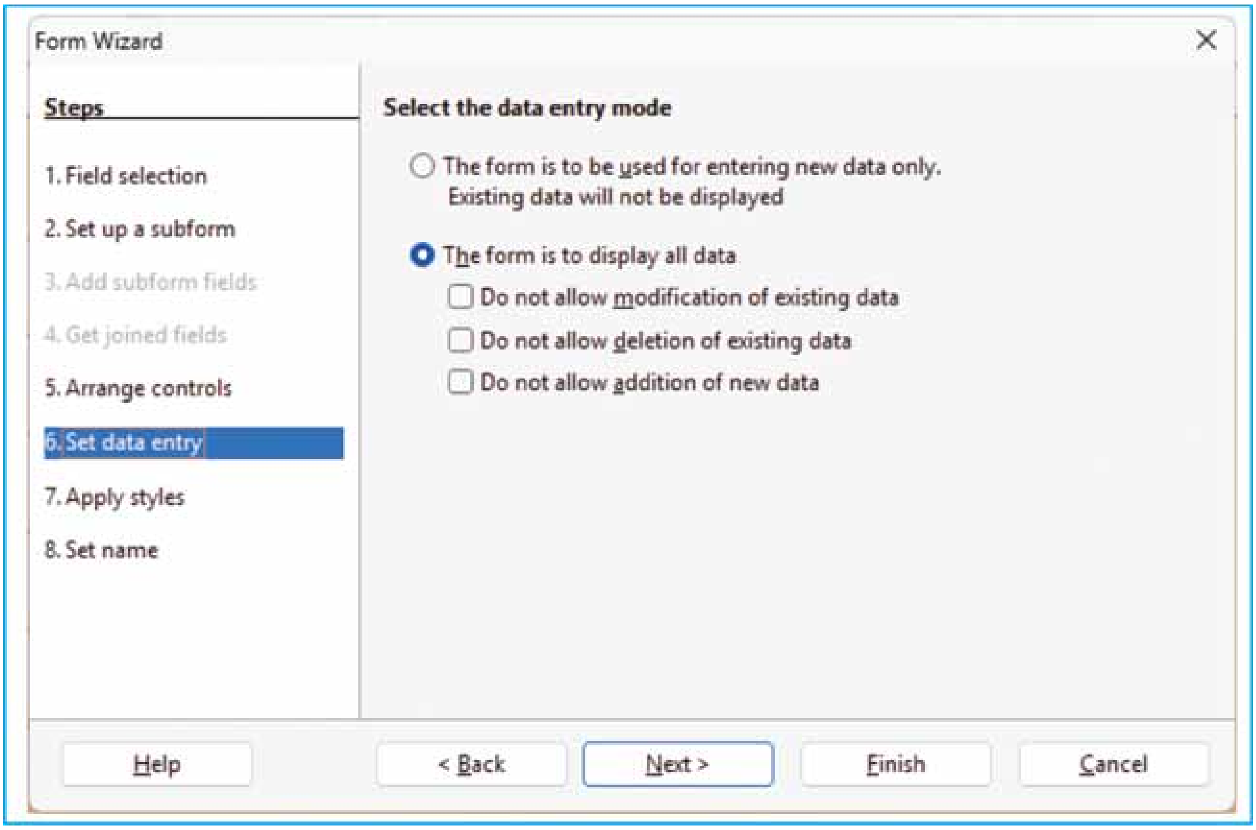
  - **Columnar:** આ ડેટા એન્ટ્રી ફોર્મ માટે એક ખૂબ જ સામાન્ય પસંદગી છે. દરેક ફિલ્ડ તેની પોતાની લાઈન પર દર્શાવવામાં આવે છે, જેમાં ફિલ્ડનું લેબલ (નામ) ડાબી બાજુએ હોય છે અને ઈનપુટ બોક્સ જમણી બાજુએ હોય છે. આ લેઆઉટ એક સમયે એક જ રેકોર્ડ દર્શાવવા માટે ઉત્તમ છે, જે યુઝર માટે વ્યક્તિગત એન્ટ્રીઓ પર ધ્યાન કેન્દ્રિત કરવાનું સરળ બનાવે છે.

  - **Tabular:** આ લેઆઉટ ફિલ્ડને સ્પ્રેડશીટની જેમ એક રોમાં ગોઠવે છે. લેબલ સામાન્ય રીતે કૉલમ હેડિંગ તરીકે દેખાય છે, અને દરેક રેકોર્ડ નવી રોમાં દર્શાવવામાં આવે છે. સીધા ટેબલમાં જ કાર્ય કરવા માટે જ્યારે એકસાથે બહુવિધ રેકોર્ડ્સ જોવા અથવા દાખલ કરવા માંગતા હોઈએ ત્યારે આ ઉપયોગી છે.

  - **Datasheet:** આ વિકલ્પ ડેટાશીટ વ્યૂમાં ટેબલ જેવું જ દેખાતું અને કાર્ય કરતુ ફોર્મ બનાવે છે. તે ગ્રીડ ફોર્મેટમાં એકસાથે અનેક રેકોર્ડ પ્રદર્શિત કરે છે, જે ઝડપી ડેટા એન્ટ્રી અને સમીક્ષા માટેની સુવિધા આપે છે.

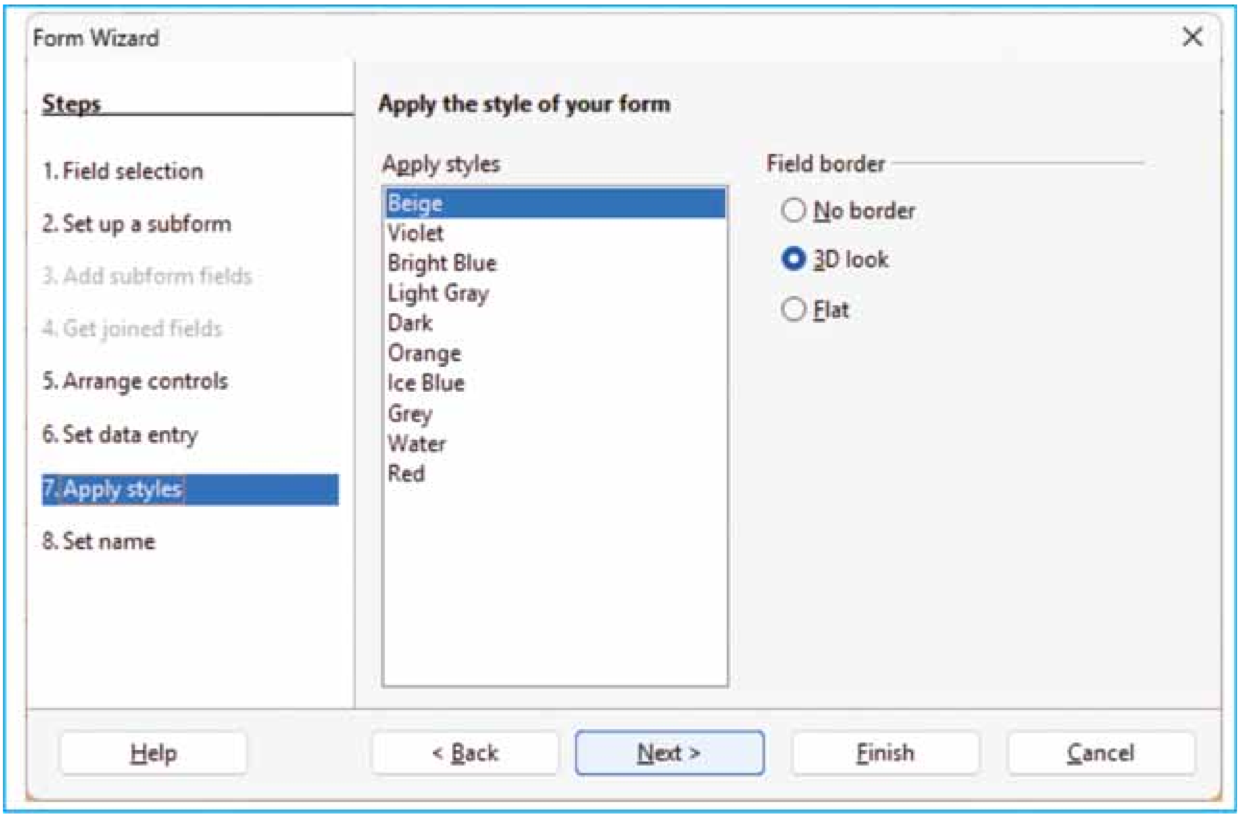
  - **Justified:** આ લેઆઉટ ફિલ્ડને વધુ કોમ્પેક્ટ એવા મલ્ટી-કૉલમ ફોર્મેટમાં ગોઠવીને ફોર્મમાં ઉપલબ્ધ જગ્યા ભરવાનો પ્રયાસ કરે છે. લેબલ ટેક્સ્ટ બોક્સની ઉપર દેખાઈ શકે છે અને ફિલ્ડ એકસાથે નજીકમાં ગોઠવાયેલા હોય છે.

આપેલ વિકલ્પોમાંથી પસંદગીનું લેઆઉટ પસંદ કરો, અને તેને ફોર્મ વિઝાર્ડની પાછળ ફોર્મ ઘટકો પર તુરંત લાગુ થયેલું જોઈ શકાશે. આકૃતિ 4.5માં દર્શાવ્યા મુજબ છઠ્ઠા પગલા પર આગળ વધવા માટે *Next* બટન ક્લિક કરો.



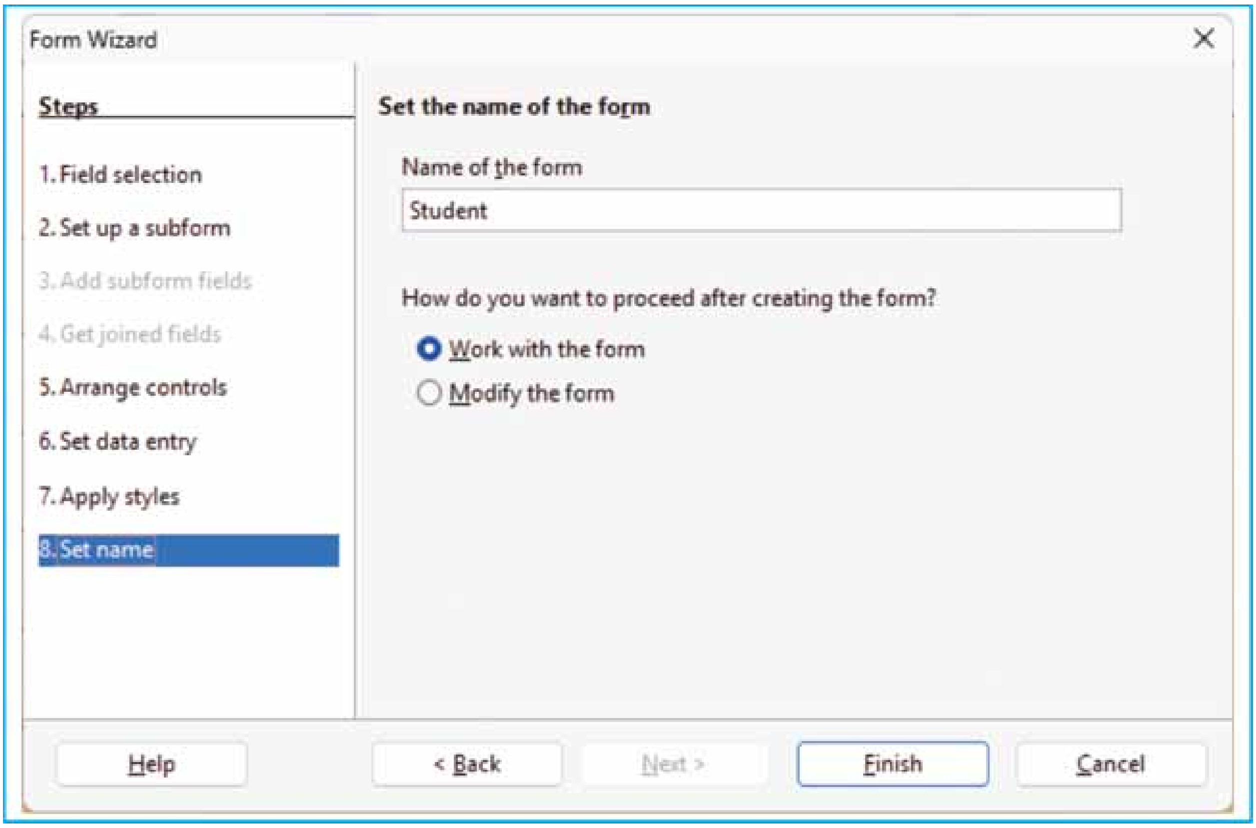
આકૃતિ 4.5 : ડેટા એન્ટ્રી મોડ સેટ કરવો

- *Set Data Entry*ના પગલા દ્વારા ફોર્મમાં ડેટાની ક્રિયા-પ્રતિક્રિયા માટેની પદ્ધતિ અને પરવાનગીઓ વ્યાખ્યાયિત કરી શકાય છે. આ પગલું નક્કી કરે છે કે યુઝર આ ફોર્મ દ્વારા રેકોર્ડ્સને કેવી રીતે જોઈ, ઉમેરી, સુધારી અને કાઢી શકશે. Set Data Entry પગલું બે વિકલ્પો પ્રદાન કરે છે:
  - o **The form is to be used for entering new data only. Existing data will not be displayed:** આ એક એવું ફોર્મ બનાવે છે જે ફક્ત નવા રેકોર્ડ્સ ઉમેરવા માટે હોય છે. હાલનો કોઈપણ ડેટા દર્શાવવામાં આવશે નહીં.
  - o **The form is to display all data:** આ એક વધુ સામાન્ય હેતુ માટેનું ફોર્મ બનાવે છે જે યુઝરને હાલના અને નવા બંને ડેટા સાથે ક્રિયા-પ્રતિક્રિયા કરવાની મંજૂરી આપે છે. વધુમાં, આપણે કેટલાક ચેકબોક્સનો ઉપયોગ કરી શકીએ છીએ જે આપણને હાલના ડેટા સાથે યુઝર શું કરી શકે છે તે બારીકાઈથી ગોઠવવાની મંજૂરી આપે છે.
- ડિફૉલ્ટ સેટિંગ્સને જેમ છે તેમ રહેવા દઈશું અને આકૃતિ 4.6માં દર્શાવ્યા મુજબ ફોર્મ વિઝાર્ડના આગલા પગલા પર જવા માટે *Next* પર ક્લિક કરો.



આકૃતિ 4.6 : ફોર્મને સ્ટાઈલ લાગુ કરવી

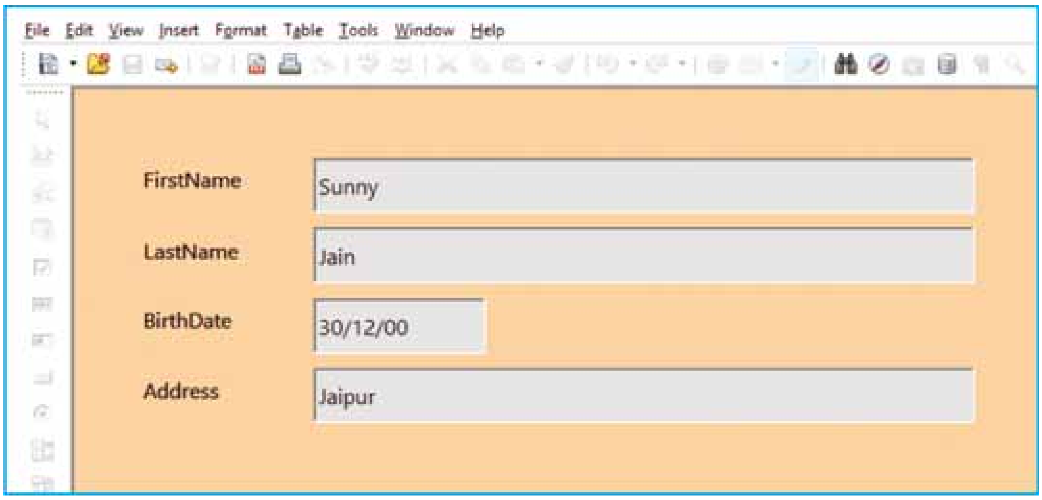
- આ પગલું આપણા ફોર્મના દેખાવને બદલવા માટે વપરાય છે. અહીં ફોર્મના દેખાવ અને ડેટા એન્ટ્રીના અનુભવને કસ્ટમાઇઝ કરી શકાય છે, તેને વધુ આકર્ષક અને વપરાશકર્તા-મૈત્રીપૂર્ણ બનાવી શકીએ છીએ. આ મુખ્યત્વે બે વિકલ્પો આપે છે:
  - o **Apply Styles:** આ ફોર્મ માટે પૂર્વ-નિર્ધારિત રંગ વિકલ્પોની પસંદગી પૂરી પાડે છે.
  - o **Field Border:** આ વિકલ્પ ડેટા એન્ટ્રી ફિલ્ડની આસપાસની બોર્ડરના દેખાવને નિયંત્રિત કરે છે. નીચેના વિકલ્પો ઉપલબ્ધ છે:
    - ♦ **No Border :** ફિલ્ડ કોઈપણ દશ્યમાન આઉટલાઇન વગર દેખાય છે.
    - ♦ **3D Look :** આ ફિલ્ડને ઉપસેલો અથવા દબાયેલો દેખાવ આપે છે.
    - ♦ **Flat :** ફિલ્ડ એક સરળ, સપાટ કિનારી ધરાવે છે.
- પસંદગીના વિકલ્પો સાથે ફોર્મ વિઝાર્ડના અંતિમ પગલા પર આગળ વધવા માટે *Next* પર ક્લિક કરો. ફોર્મ વિઝાર્ડનું અંતિમ પગલું આકૃતિ 4.7માં દર્શાવેલ છે.



આકૃતિ 4.7 : ફોર્મને નામ આપવું

- *Set name* પગલું એ ફોર્મ વિઝાર્ડનો અંતિમ તબક્કો છે. અહીં ફોર્મને એક અનન્ય ઓળખ આપીને અને વિઝાર્ડ બંધ થયા પછી તરત જ તમે શું કરવા માંગો છો તે નક્કી કરીને ફોર્મ બનાવવાની પ્રક્રિયાને અંતિમ રૂપ આપવામાં આવે છે.
- આ પગલું નીચેના બે વિકલ્પો પણ આપે છે:
  - o **Work with the form:** જો આપણે આ વિકલ્પ પસંદ કરીએ, તો વિઝાર્ડ બંધ થશે અને નવું બનાવેલું ફોર્મ *ફોર્મ વ્યૂ (Form View)* માં ખુલશે.
  - o **Modify the form:** જો આ વિકલ્પ પસંદ કરવામાં આવે તો વિઝાર્ડ બંધ થશે અને ફોર્મ *ડિઝાઈન વ્યૂ (Design View)* માં ખુલશે.

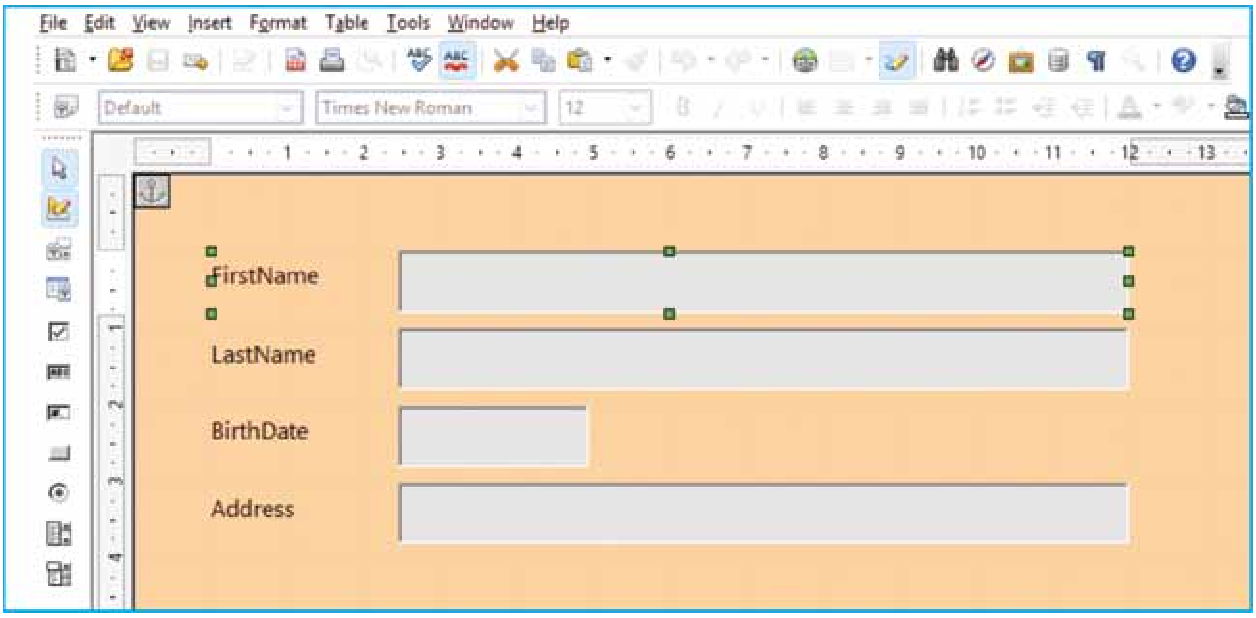
Student ટેબલ માટે બનાવવામાં આવેલું ફોર્મ આકૃતિ 4.8માં દર્શાવ્યું છે.



આકૃતિ 4.8 : Student ટેબલ પરથી બનાવવામાં આવેલું ફોર્મ

ફોર્મમાં સુધારા કરવા (Modifying Form)

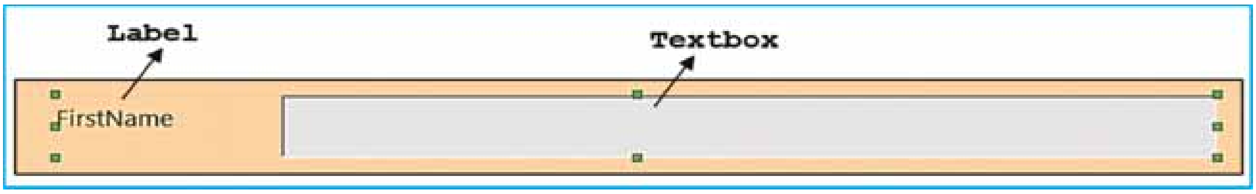
એકવાર ફોર્મ બની જાય, પછી તેના કંટ્રોલ (control)ના વિવિધ પરિમાણોને બદલવા માટે ફોર્મની ડિઝાઇનને સરળતાથી બદલી શકાય છે. ફોર્મની ડિઝાઇન સુધારવા માટે, ફોર્મના નામ (Student) પર રાઈટ ક્લિક કરો અને *Edit* વિકલ્પ પસંદ કરો. આનાથી ફોર્મ ડિઝાઇન વ્યૂ *(Design View)* માં ખુલશે, જે આકૃતિ 4.9માં દર્શાવેલ છે.



આકૃતિ 4.9 : ફોર્મ ડિઝાઈન વ્યૂ

લેબલ અને ટેક્સ્ટબોક્સ પસંદ કરો (Selecting Label and Textbox)

ફોર્મ ડિઝાઇન વ્યૂમાં, કંટ્રોલ સામાન્ય રીતે બે ભાગનું બનેલું હોય છે : લેબલ (Label) અને ટેક્સ્ટબોક્સ (Textbox). તેના પર માત્ર ક્લિક કરીને કંટ્રોલને પસંદ કરી શકાય છે. એકવાર પસંદ કર્યા પછી કંટ્રોલની આસપાસ એડિટ પોઇન્ટ્સ નામે ઓળખાતા નાના લીલા ચોરસ (small green squares) દેખાશે. જે આકૃતિ 4.10માં દર્શાવેલ છે.

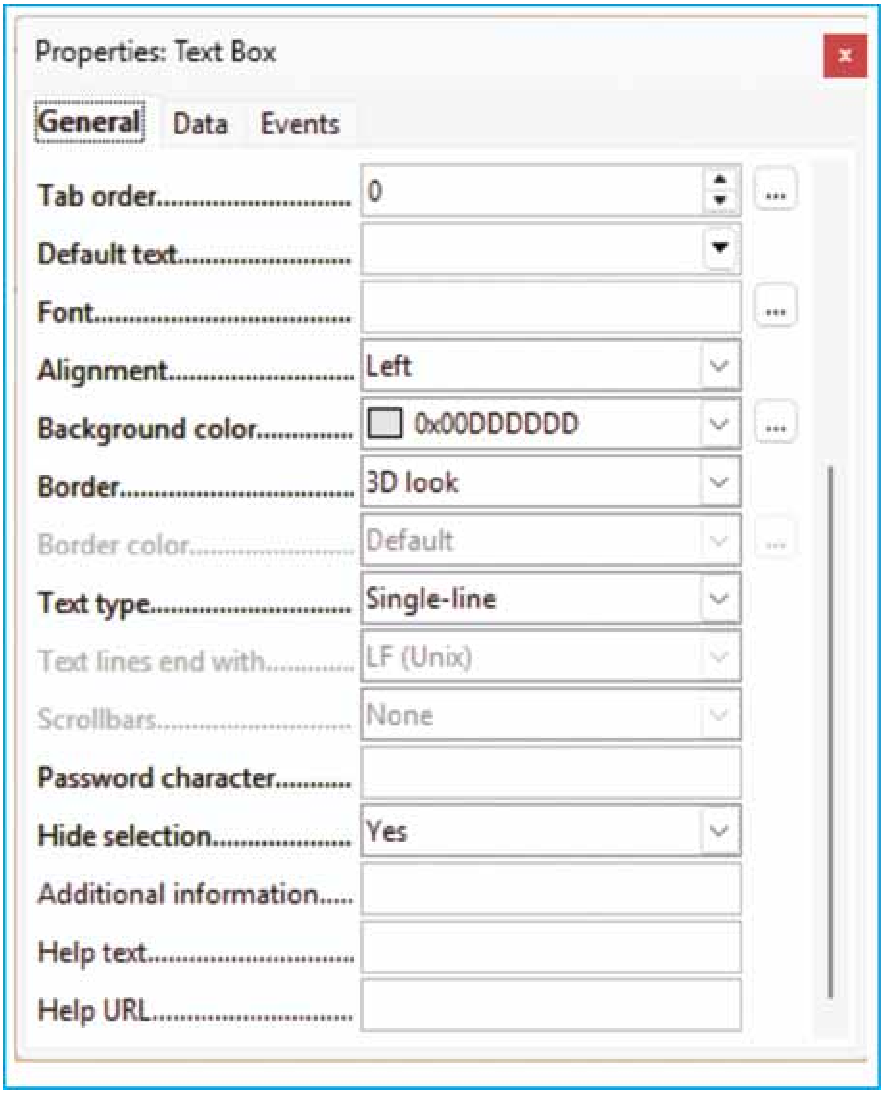


આકૃતિ 4.10 : ફોર્મ ડિઝાઈન વ્યૂમાં કંટ્રોલની પસંદગી

લેબલ અથવા ટેક્સ્ટબોક્સમાંથી કોઈપણ એકને સ્વતંત્ર રીતે અલગથી પસંદ કરવા માટે કંટ્રોલ કીને દબાવી રાખો અને પછી ઇચ્છિત સ્વતંત્ર ઘટક પર ક્લિક કરો. આ ક્રિયા ફક્ત લેબલને અથવા ફક્ત ટેક્સ્ટબોક્સને જ પસંદ કરશે, જે આકૃતિ 4.11માં દર્શાવવામાં આવ્યું છે.

આકૃતિ 4.11 : કંટ્રોલના સ્વતંત્ર ઘટકને પસંદ કરવો

એકવાર આપણે કંટ્રોલના સ્વતંત્ર ઘટક (જેમ કે લેબલ અથવા ટેક્સ્ટબોક્સ)ને પસંદ કરી લઈએ, પછી તેને સરળતાથી ખસેડી અથવા તેનું કદ બદલી શકાય છે. ઘટકને ખસેડવા માટે તેને માત્ર ડ્રેગ કરવામાં આવે છે. કદ બદલવા માટે ચોક્કસ ઘટકની આસપાસના એડિટ પોઇન્ટ્સ (નાના લીલા ચોરસ)ને ડ્રેગ કરવામાં આવે છે.

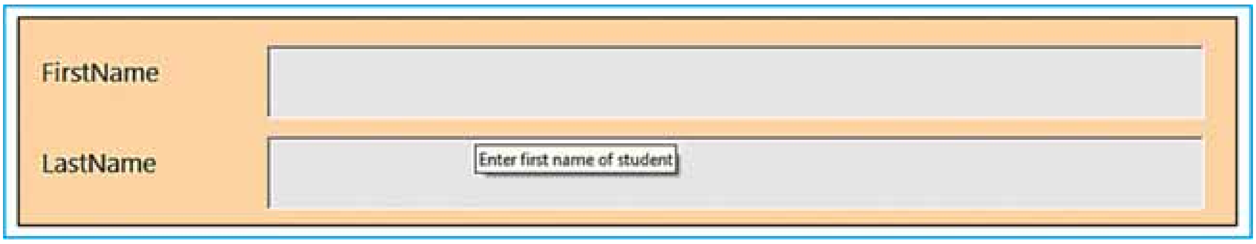


આકૃતિ 4.12 : ટેક્સ્ટબોક્સ માટે Properties ડાયલોગ બોક્સ

ફોર્મ ડિઝાઇન વ્યૂમાં કંટ્રોલના ગુણધર્મો (properties) માં ફેરફાર કરવા માટે, સૌપ્રથમ ચોક્કસ ટેક્સ્ટબોક્સ (textbox)ને પસંદ કરો. આ માટે કંટ્રોલ કી દબાવી રાખીને અને ટેક્સ્ટબોક્સ પર ક્લિક કરો. ત્યારબાદ, પસંદ કરેલા ટેક્સ્ટબોક્સ પર રાઈટ ક્લિક કરો અને જે કન્ટેક્સ્ટ મેનુ ખુલે તેમાંથી *Control...* વિકલ્પ પસંદ કરો. આ ક્રિયા *Properties* ડાયલોગ બોક્સ ખોલવામાં આવશે, જે આકૃતિ 4.12 માં દર્શાવેલ છે. અહીં ટેક્સ્ટબોક્સ માટેના વિવિધ કન્ફિગરેબલ ગુણધર્મો દર્શાવ્યા છે.

ચાલો, આ ટેક્સ્ટબોક્સ માટે એક પ્રોપર્ટી બદલીએ. પ્રોપર્ટીની સૂચિમાં નીચેની તરફ *Help Text* પ્રોપર્ટી દેખાય ત્યાં સુધી સ્ક્રોલ કરો. તેની બાજુના ટેક્સ્ટબોક્સમાં *"Enter first name of student."* ટાઈપ કરો. આ પ્રોપર્ટી કંટ્રોલમાં એક સરળ ટૂલટીપ ઉમેરે છે.

કાર્ય પૂર્ણ થાય ત્યારે *Properties* ડાયલોગ બોક્સ બંધ કરો. ફોર્મની ડિઝાઈન સેવ કરો, અને ફોર્મ ખોલો. હવે, જ્યારે જ્યારે માઉસ પોઇન્ટર *FirstName* ફિલ્ડના ટેક્સ્ટબોક્સ પર રાખવામાં આવશે, ત્યારે આકૃતિ 4.13માં બતાવ્યા મુજબ ટૂલટીપ દર્શાવવામાં આવશે.

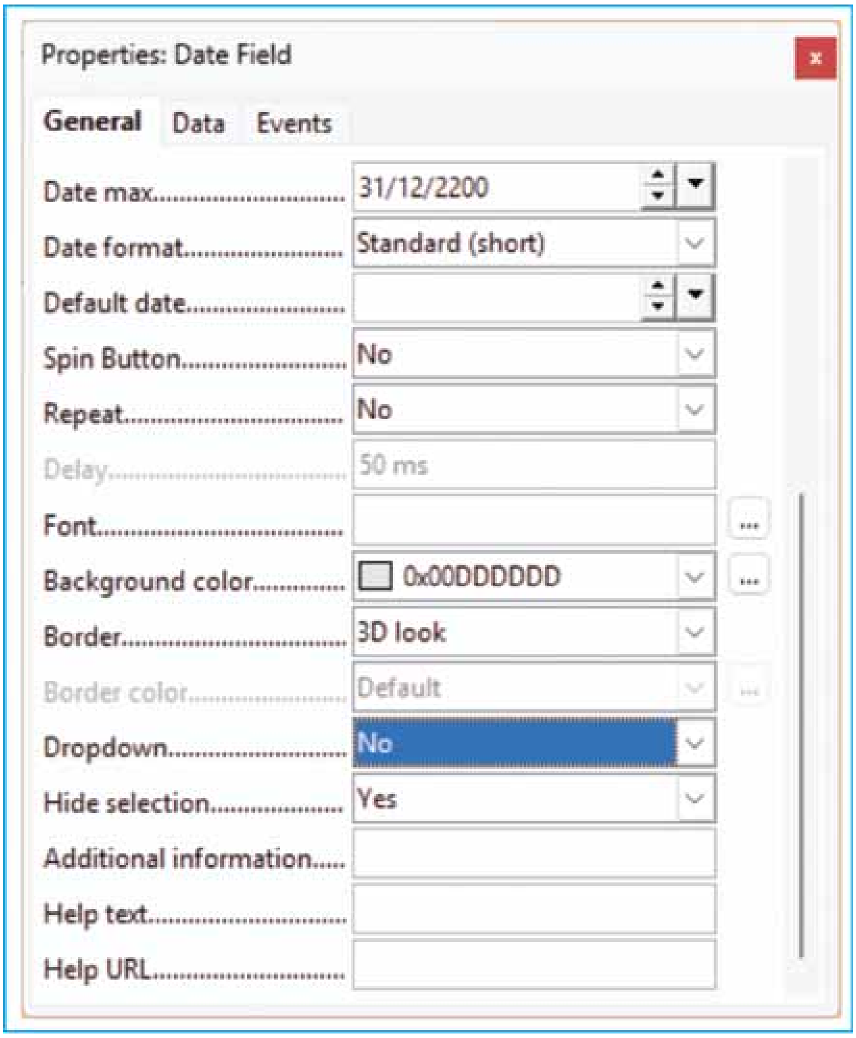


આકૃતિ 4.13 : ટેક્સ્ટબોક્સ માટે ટૂલટીપ દર્શાવતી Help text પ્રોપર્ટી

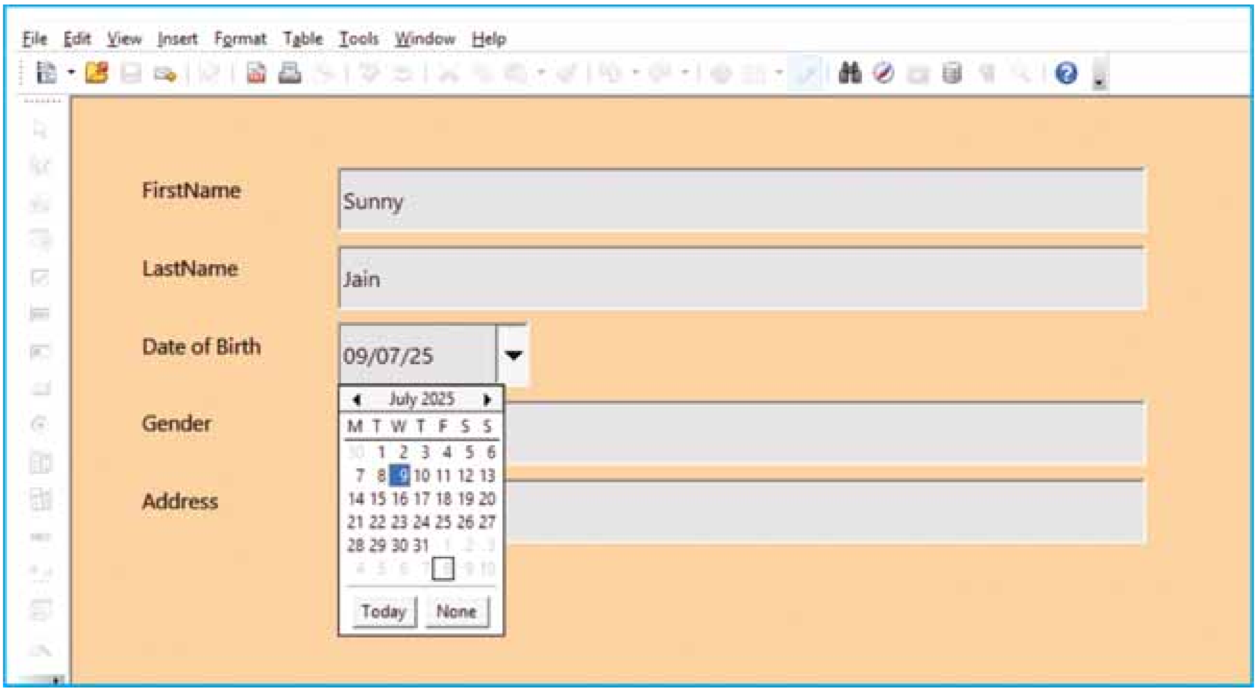
હવે, આપણે ફોર્મની બીજી પ્રોપર્ટી બદલીશું. Birthdate ટેક્સ્ટબોક્સ પસંદ કરી તેના પર રાઈટ ક્લિક કરો અને *Control*... વિકલ્પ પર ક્લિક કરો. તેથી BirthDate ફિલ્ડની પ્રોપર્ટી આકૃતિ 4.14માં દર્શાવ્યા મુજબ ખૂલશે. *Dropdown* પ્રોપર્ટીની કિંમત 'No' થી બદલીને 'Yes' કરો.

હવે, જ્યારે ફોર્મ વ્યૂ (*Form view*) માં વિદ્યાર્થીની જન્મ-તારીખ દાખલ કરવામાં આવશે, ત્યારે યુઝરને કેલેન્ડર દેખાશે. પ્રોપર્ટી ડાયલોગ બોક્સ બંધ કરો, ફોર્મના ડિઝાઈન વ્યૂને સેવ કરો અને ફોર્મ ખોલો.

BirthDate ફિલ્ડમાં તારીખ દાખલ કરવાની ચકાસણી કરો. આ પ્રોપર્ટીનું આઉટપુટ આકૃતિ 4.15માં દર્શાવેલું છે.



આકૃતિ 4.14 : Birthdate ફિલ્ડની Dropdown પ્રોપર્ટી

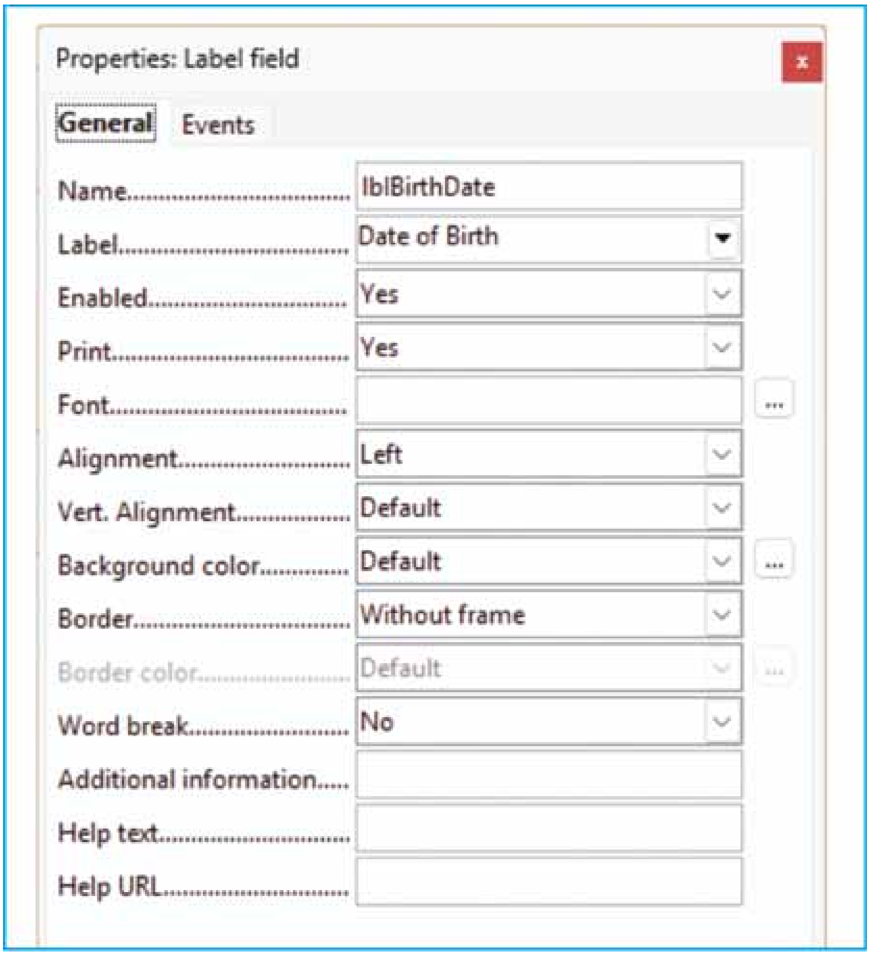


આકૃતિ 4.15 : કેલેન્ડર દર્શાવતું Birthdate ફિલ્ડ

આપણે લેબલનું લખાણ પણ બદલી શકીએ છીએ, જે સામાન્ય રીતે લેબલ બોક્સમાં દર્શાવેલ ફિલ્ડનું નામ હોય છે. *BirthDate* લેબલ પર રાઈટ-ક્લિક કરો અને *Control...* વિકલ્પ પસંદ કરીને તેનું *Properties* ડાયલોગ બોક્સ ખોલો.

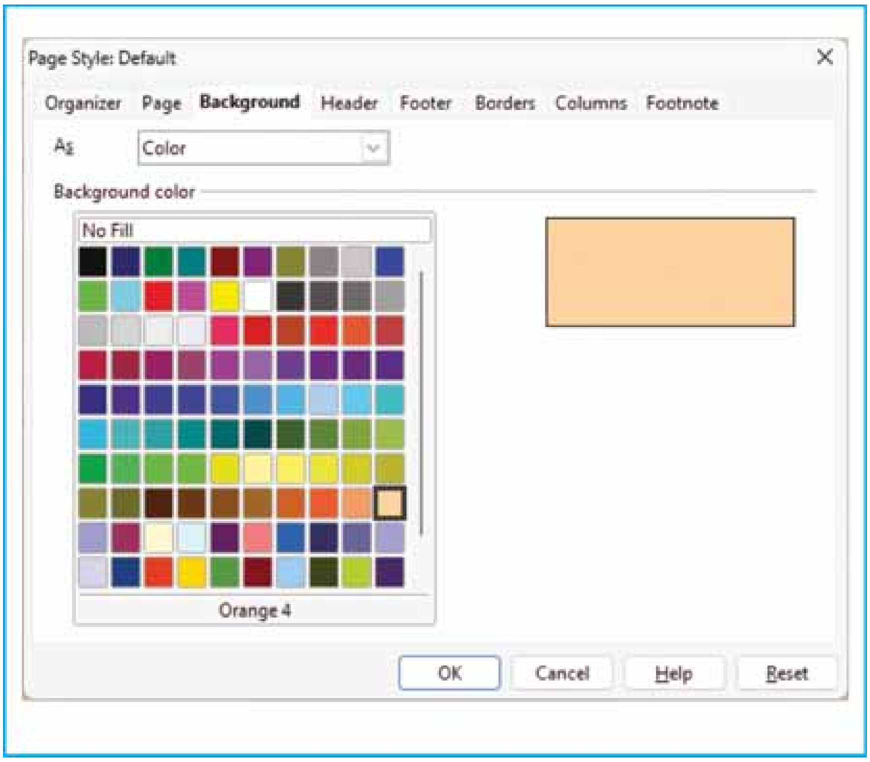
Properties ડાયલોગ બોક્સમાં, *Label* પ્રોપર્ટી શોધો અને આકૃતિ 4.16માં દર્શાવ્યા મુજબ તેમાં *BirthDate*ની જગ્યાએ *Date of Birth* લખો. આ ફેરફાર કર્યા પછી, *BirthDate* ફિલ્ડ માટે સુધારેલું લખાણ લેબલ તરીકે જોવા માટે આઉટપુટ તપાસો.

ફોર્મ ડિઝાઈન વ્યૂમાં ફોર્મનું બેકગ્રાઉન્ડ પણ બદલી શકાય



આકૃતિ 4.16 : Birthdate ફિલ્ડનું લેબલ બદલવું

છે. ફોર્મની અંદરની કોઈપણ ખાલી જગ્યા પર રાઈટ ક્લિક કરો. કોન્ટેક્સ્ટ મેનૂમાંથી, *Page* વિકલ્પ પસંદ કરો. આમ કરવાથી આકૃતિ 4.17માં દર્શાવ્યા મુજબ *Page Style* ડાયલોગ બોક્સ ખુલશે. હવે, *Background* ટેબ પર જાઓ અને સોલિડ કલર લાગુ કરવા માટે આપેલા વિવિધ રંગોમાંથી કોઈપણ રંગની પસંદગી કરો. *OK* બટન ક્લિક કરો. આથી ફોર્મનું બેકગ્રાઉન્ડ પસંદ કરેલ રંગ સાથે બદલાયેલું જણાશે.



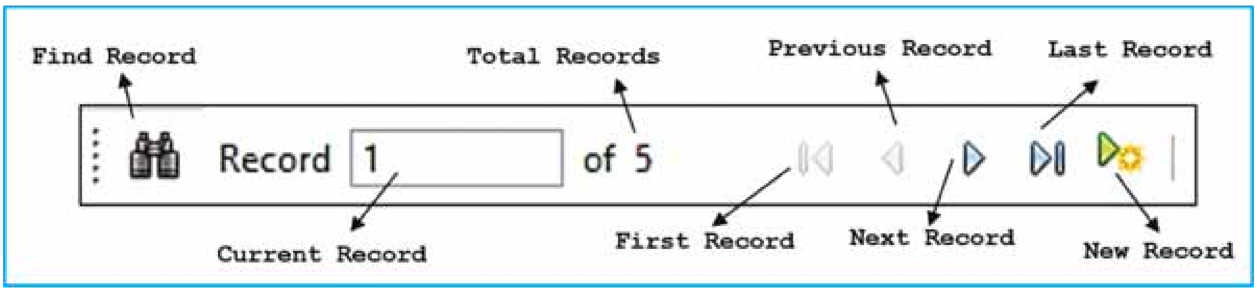
આકૃતિ 4.17 : ફોર્મનો બેકગ્રાઉન્ડ રંગ બદલવો

બેકગ્રાઉન્ડ તરીકે કોઈ ઈમેજ મૂકવા માટે, *As* ડ્રોપડાઉન મેનૂમાં *Graphic* પસંદ કરો. *Browse* બટન પર ક્લિક કરો અને ઈચ્છિત ઈમેજ ફાઈલ પસંદ કરો. આથી, પસંદ કરેલું ચિત્ર બેકગ્રાઉન્ડ તરીકે દેખાશે.

તમારા શિક્ષકના માર્ગદર્શન હેઠળ, તમે *Properties* ડાયલોગ બોક્સમાં અન્ય ફિલ્ડ પ્રોપર્ટીઝ વિશે જાણી શકો છો અને તેમાં ફેરફાર પણ કરી શકો છો.

ફોર્મની મદદથી રેકોર્ડ ઉમેરવા (Insert) અને દૂર (Delete) કરવા

આપણે જાણીએ છીએ તેમ, ફોર્મનો મુખ્યત્વે ઉપયોગ ટેબલમાં ડેટા દાખલ કરવા માટે થાય છે. જ્યારે આપણે કોઈ ફોર્મને ફોર્મ વ્યૂમાં ખોલીએ છીએ, ત્યારે સ્ક્રીનની નીચેના ભાગમાં આકૃતિ 4.18માં દર્શાવ્યા મુજબ નેવિગેશન બાર (navigation bar) દર્શાવશે. આ બારમાં એવા બટન છે જે આપણને આપણા ટેબલમાં રહેલા રેકોર્ડ્સ વચ્ચે સરળતાથી ફરવાની (આગળ-પાછળ જવાની) સુવિધા આપે છે.



આકૃતિ 4.18 : ફોર્મ વ્યૂમાં નેવિગેશન બાર

ફોર્મનો ઉપયોગ કરીને ટેબલમાં નવો રેકોર્ડ ઉમેરવા માટે નેવિગેશન બાર પરના *New Record* બટન પર ક્લિક કરો. એક ખાલી રેકોર્ડ દેખાશે, અને પછી દરેક ફિલ્ડ માટેના ટેક્સ્ટબોક્સમાં મૂલ્યો દાખલ કરી શકીએ છીએ.

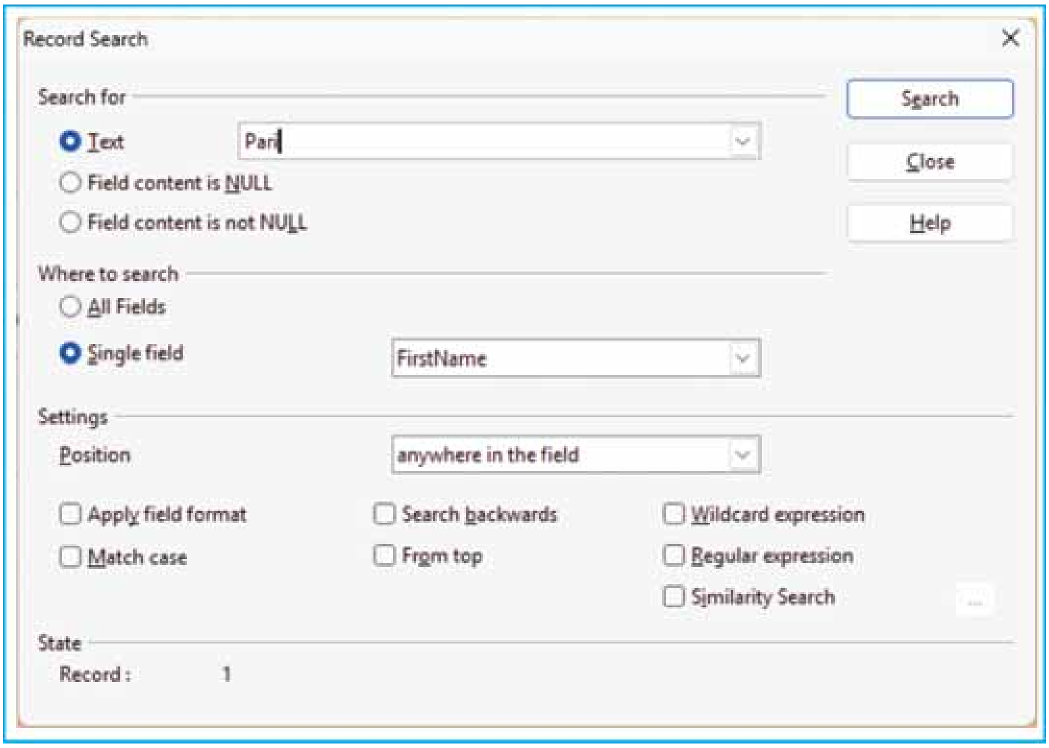
ટેબલમાંથી રેકોર્ડ દૂર કરવા માટે, સૌ પ્રથમ જે રેકોર્ડ દૂર કરવો છે તે શોધો. પછી, નેવિગેશન બારની બાજુમાં આવેલા *Delete Record* બટન પર ક્લિક કરો. એક ચેતવણી સંદેશ પોપ અપ થશે જે રેકોર્ડ ડિલીટ કરવાની પુષ્ટિ કરશે . *Yes* પર ક્લિક કરવાથી તે રેકોર્ડ ટેબલમાંથી કાયમ માટે દૂર થઈ જશે.

ફોર્મનો ઉપયોગ કરીને રેકોર્ડ શોધવો (Searching Record using Form)

અવારનવાર, આપણને ફોર્મના કોઈ ચોક્કસ રેકોર્ડને શોધવાની જરૂર પડતી હોય છે. સદભાગ્યે બેઝમાં બિલ્ટ-ઇન સર્ચ સુવિધા શામેલ છે જે ટેબલમાંથી રેકોર્ડ શોધવામાં મદદ કરે છે.

ફોર્મમાં રેકોર્ડ શોધવાના પગલાં અહીં આપેલા છે:

- નેવિગેશન બાર પરના *Find Record* બટન પર ક્લિક કરો. આનાથી આકૃતિ 4.19માં દર્શાવ્યા મુજબ *Record Search* ડાયલોગ બોક્સ ખુલશે.
- *Search for* બોક્સમાં જે શબ્દ શોધવા માંગીએ છીએ તે *સર્ચ ટર્મ (search term)* દાખલ કરો.
- *Where to search* વિભાગમાં બે વિકલ્પો આપવામાં આવ્યા છે: કોઈ ચોક્કસ ફિલ્ડમાં શોધવા માટે *Single field* પસંદ કરો અને ડ્રોપડાઉન મેનૂમાંથી ફિલ્ડનું નામ પસંદ કરો. ટેબલના બધા ફિલ્ડ્સમાં શોધ ચલાવવા માટે *All Fields* વિકલ્પ પસંદ કરો.
- આપણે *Position* ડ્રોપડાઉન મેનૂનો ઉપયોગ કરીને, રેકોર્ડની અંદર સર્ચ ટર્મ ક્યાં દેખાવી જોઈએ તે પણ સ્પષ્ટ કરી શકીએ છીએ.
- શોધ શરૂ કરવા માટે *Search* બટન પર ક્લિક કરો. ત્યાર બાદ, બેઝ સર્ચ ટર્મને સમાવતા રેકોર્ડને પ્રદર્શિત કરશે.



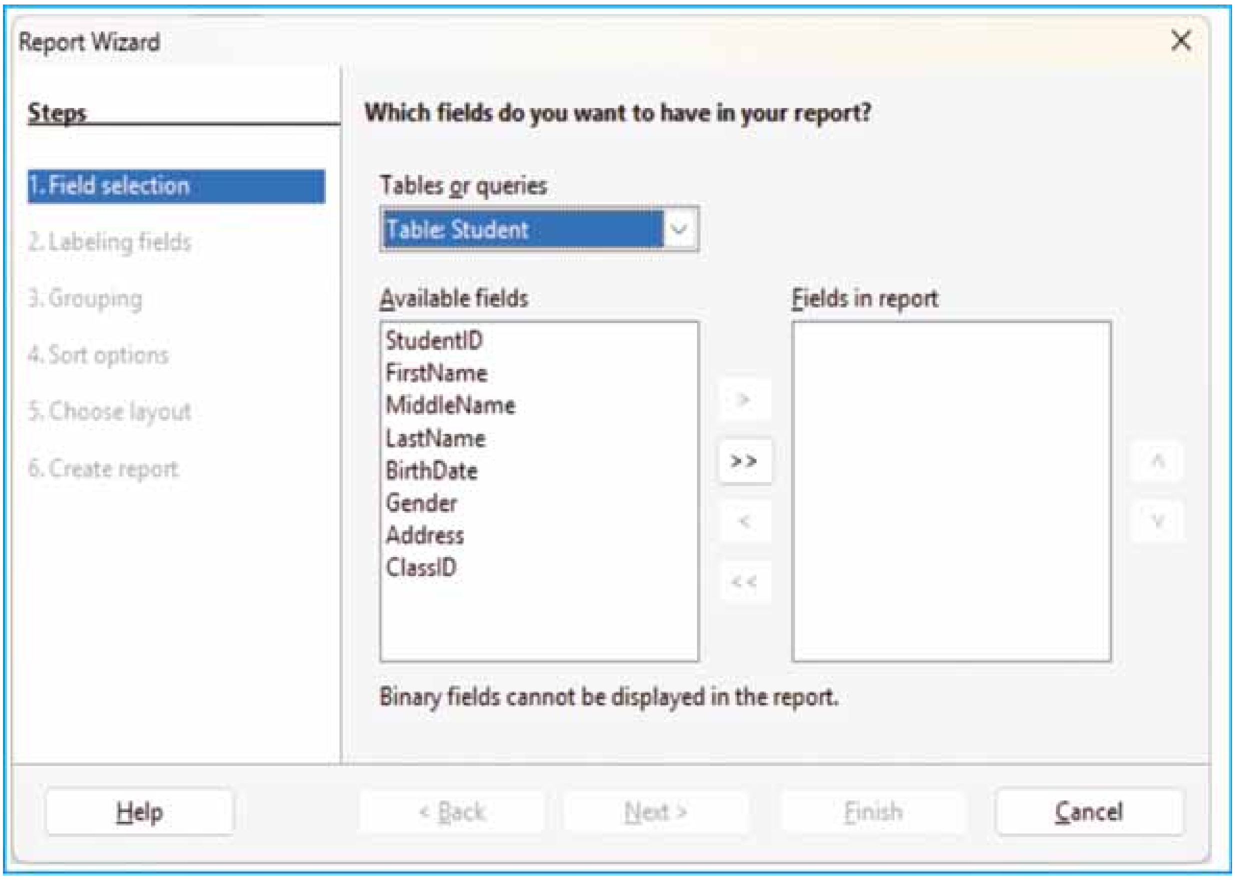
આકૃતિ 4.19 : રેકોર્ડની શોધ

રિપોર્ટસ્ (Reports)

રિપોર્ટ પ્રાપ્ત કરેલી માહિતીને આકર્ષક અને વ્યવસ્થિત ફોર્મેટમાં રજૂ કરવાની એક ઉત્તમ રીત છે. તેનો મુખ્ય હેતુ ઘણીવાર હાર્ડકોપી (છાપેલી નકલ) બનાવવાનો હોય છે. સામાન્ય રીતે રિપોર્ટની રચના તે પ્રિન્ટમાં કેવો દેખાશે તેને પ્રાધાન્ય આપે છે.

આપણે રીપોર્ટની રચના ક્વેરી, ટેબલ અથવા બંનેના સંયોજન પર આધારિત કરી શકીએ છીએ. જો આપણા રિપોર્ટમાં એકથી વધુ ટેબલના ફિલ્ડનો સમાવેશ કરવાની જરૂર હોય તો તે બધા ફિલ્ડને એકસાથે સમાવતી એક ક્વેરી સૌ પ્રથમ બનાવવી તે સારી પ્રથા છે. ત્યારબાદ, તે સંયુક્ત ક્વેરીનો ઉપયોગ કરીને અહેવાલ બનાવી શકીએ છીએ.

ચાલો, Student ટેબલનો ઉપયોગ કરીને એક રિપોર્ટની રચના કરીએ. સૌ પ્રથમ, બેઝ લેઆઉટની ડાબી બાજુની પેનલમાં *Reports* આઇકન પર ક્લિક કરો. ત્યાર પછી, *Use Wizard to Create Report...* વિકલ્પ દેખાશે. તેના પર ક્લિક કરો, અને આકૃતિ 4.20માં દર્શાવ્યા મુજબ *Report Wizard* રજૂ કરવામાં આવશે, જે રિપોર્ટને તૈયાર કરવા માટે છ અલગ-અલગ પગલાંઓમાં આપણને માર્ગદર્શન આપશે.

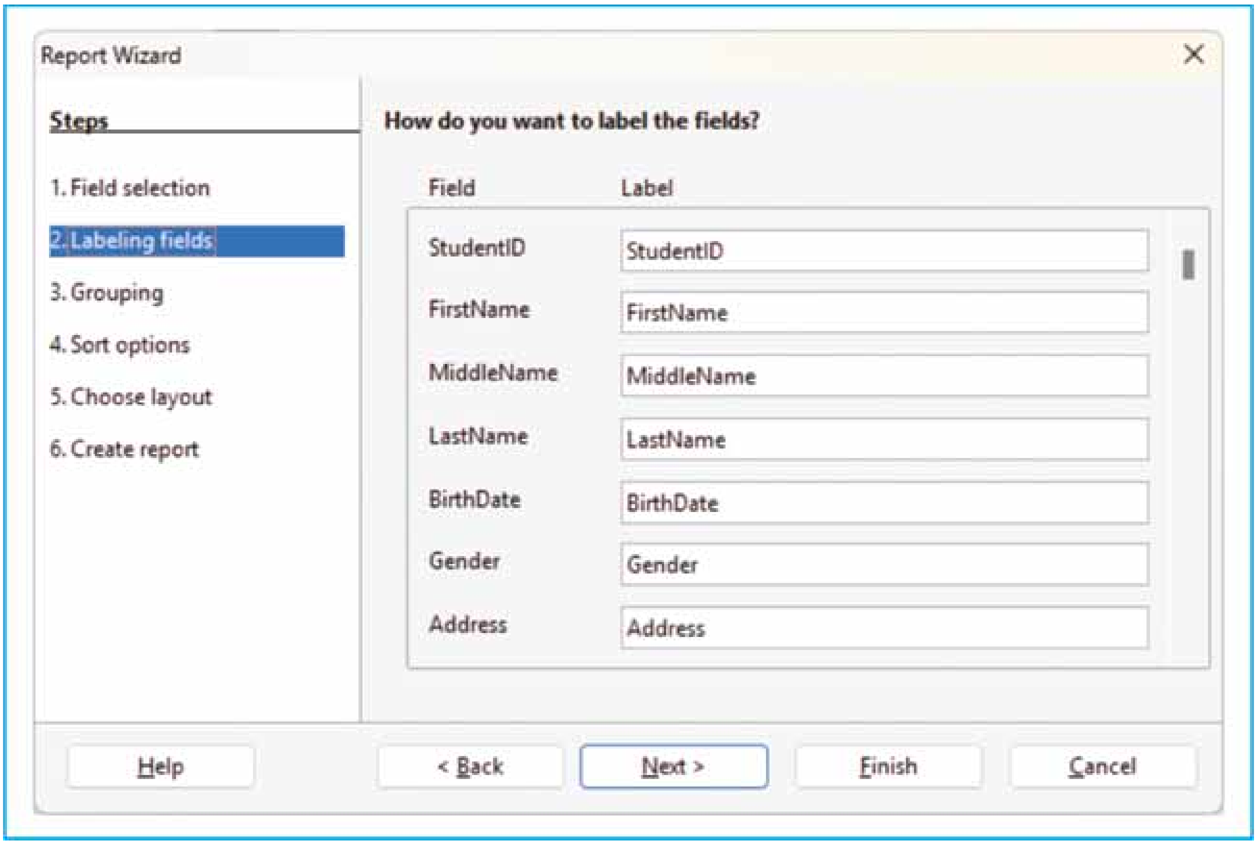


આકૃતિ 4.20 : રિપોર્ટ વિઝાર્ડ

રિપોર્ટ બનાવવાનું શરૂ કરવા માટે રિપોર્ટ વિઝાર્ડમાં પ્રથમ પગલું ડેટા સોર્સ (data source) ને પસંદ કરવાનું છે. *Tables or Queries* યાદીમાંથી Student ટેબલ પસંદ કરો. ટેબલ પસંદ કર્યા પછી રિપોર્ટમાં સમાવવા માંગતા હોય તેવા ચોક્કસ ફિલ્ડ પસંદ કરવા પડશે. આ ફિલ્ડને ઉપલબ્ધ યાદીમાંથી આપણા અહેવાલમાં ઉમેરવા માટે '>' બટનનો ઉપયોગ કરો.

ફોર્મ વિઝાર્ડની જેમ જ રિપોર્ટમાં પણ ફિલ્ડનું વ્યવસ્થાપન કરવા માટે ઘણા વિકલ્પો છે. >> બટન બધા ઉપલબ્ધ ફિલ્ડને ઝડપથી આપણા અહેવાલમાં ઉમેરવાની સુવિધા આપે છે. પસંદ કરેલા ફિલ્ડને રિપોર્ટમાંથી દૂર કરવા માટે < બટનનો ઉપયોગ કરી શકાય છે. જો રિપોર્ટમાંથી એકસાથે બધા ફિલ્ડ દૂર કરવા માંગતા હોઈએ તો << બટનનો ઉપયોગ કરી શકાય.

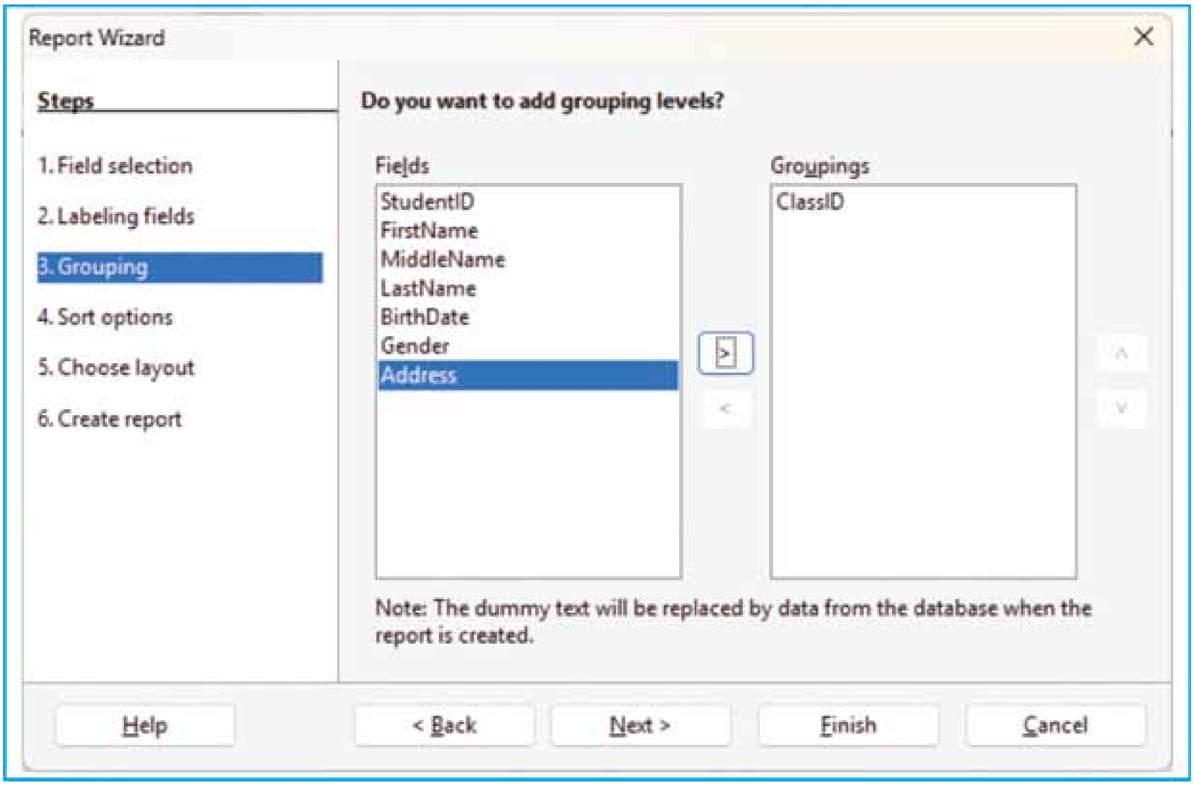
ફિલ્ડની પસંદગીઓ ગોઠવ્યા પછી રિપોર્ટ વિઝાર્ડના *Labeling Fields* પગલા પર આગળ વધવા માટે *Next* બટન પર ક્લિક કરો. આકૃતિ 4.2માં રીપોર્ટ વિઝાર્ડનું બીજું પગલું દર્શાવવામાં આવ્યું છે.



આકૃતિ 4.21 : ફિલ્ડના લેબલ બદલવા

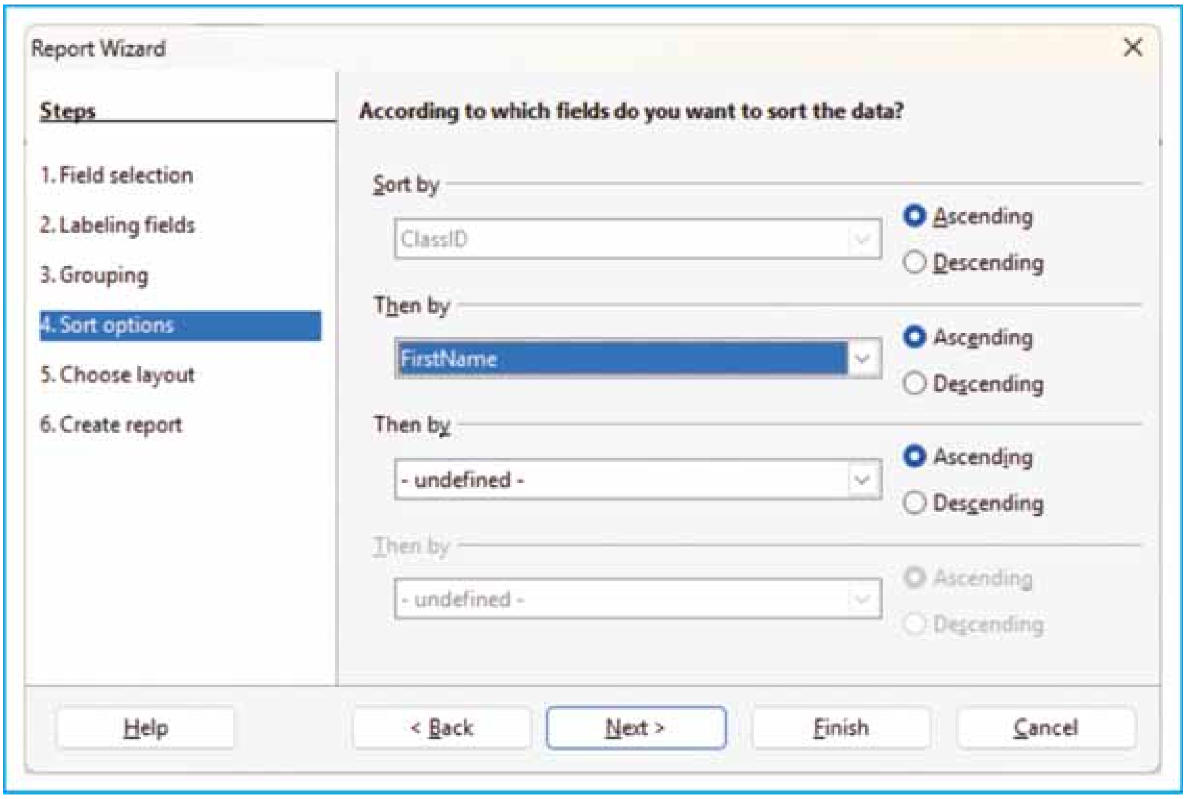
આ પગલું ફિલ્ડ લેબલ્સ તરીકે અંતિમ અહેવાલમાં કેવી રીતે દેખાશે તેમાં ફેરફાર કરવાની છૂટ આપે છે. મૂળભૂત રીતે બેઝ રિપોર્ટમાં લેબલ તરીકે ટેબલ અથવા ક્વેરીમાંથી ચોક્કસ ફિલ્ડનેમનો ઉપયોગ કરશે. આ પગલું આપણને તે લેબલને યુઝર્સ માટે વધુ મૈત્રીપૂર્ણ અને વાંચી શકાય તેવા બનાવવાની તક આપે છે. લેબલના લખાણ વચ્ચે જગ્યાઓ ઉમેરી શકાય છે, કેપિટલાઇઝેશન (capitalization) બદલી શકીએ છીએ અથવા વધુ વર્ણનાત્મક શબ્દોનો ઉપયોગ કરી શકીએ છીએ.

આકૃતિ 4.22માં દર્શાવ્યા મુજબ રિપોર્ટ વિઝાર્ડના *Grouping* પગલા પર આગળ વધવા માટે *Next* પર ક્લિક કરો.



આકૃતિ 4.22 : ફિલ્ડના જૂથ બનાવવા - Grouping

*Grouping*માં એક ચોક્કસ ફિલ્ડમાં સમાન મૂલ્ય ધરાવતી ડેટાની તમામ રોને ભેગી કરીને તેને એકસાથે રજૂ કરવામાં આવે છે. જ્યારે ડેટાનું ગ્રુપિંગ કરીએ છીએ ત્યારે રિપોર્ટ પસંદ કરેલ ગ્રુપિંગ ફિલ્ડમાં દરેક અનન્ય મૂલ્ય

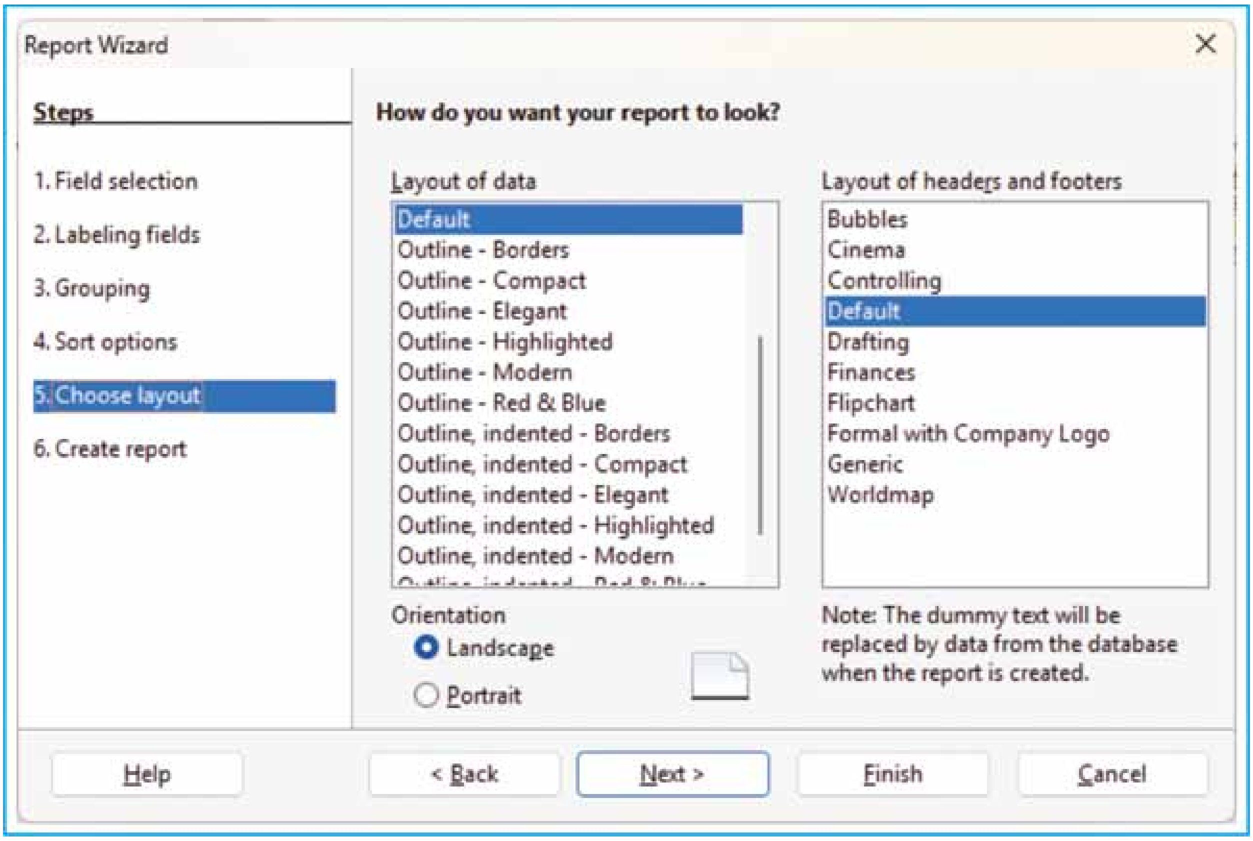


આકૃતિ 4.23 : સોર્ટ વિકલ્પો

માટે એક હેડર પ્રદર્શિત કરશે, જેના પછી તે જૂથ સાથે સંબંધિત તમામ વિગતવાર રેકોર્ડ્સ આવશે. ઉપલબ્ધ યાદીમાંથી એક અથવા વધુ ફિલ્ડ પસંદ કરો અને તેમને *'Groupings'* યાદીમાં ખસેડો. અહીં, *'ClassID'* ફિલ્ડને ગ્રુપિંગ ફિલ્ડ તરીકે ઉમેરવામાં આવ્યું છે.

આકૃતિ 4.23માં દર્શાવ્યા મુજબ રિપોર્ટ વિઝાર્ડના પછીના પગલાં – *Sort Options* પર જવા માટે *Next* પર ક્લિક કરો.

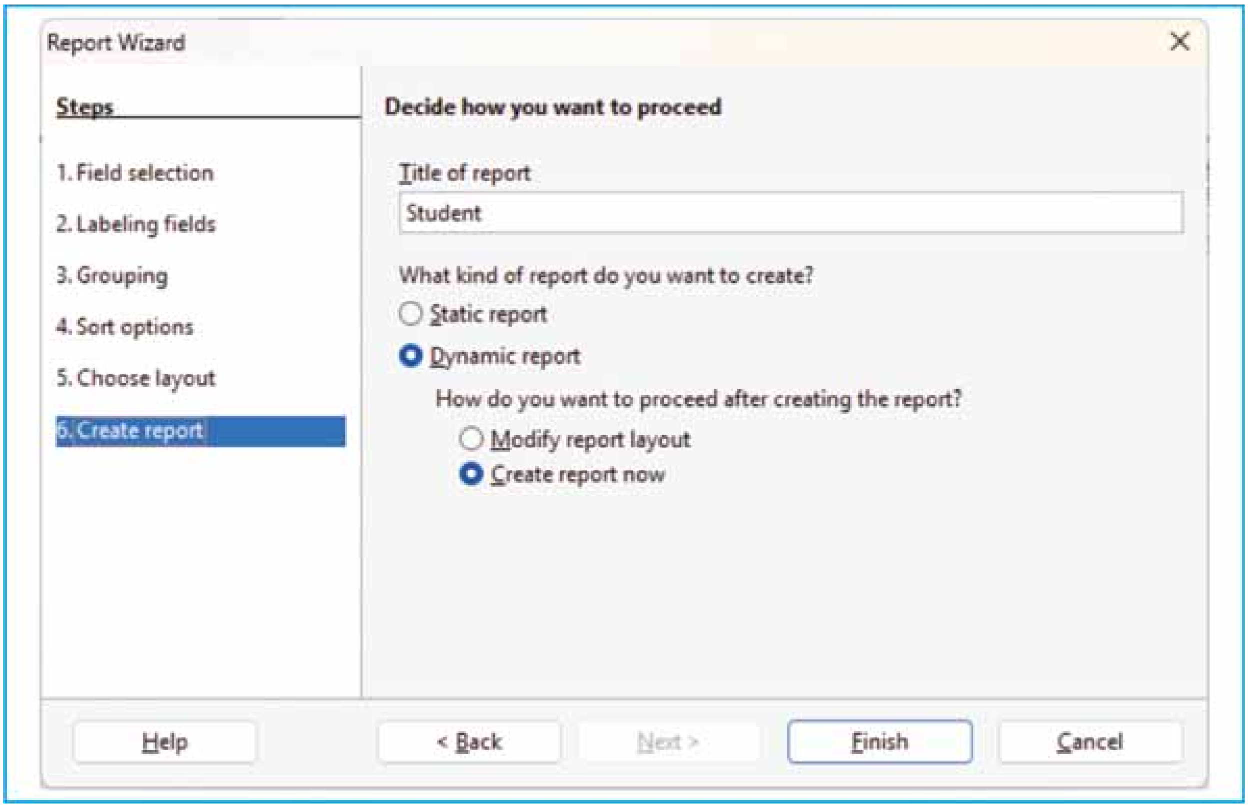
આ પગલામાં રિપોર્ટની અંદરના અને ખાસ કરીને વ્યાખ્યાયિત કરેલા કોઈપણ ગ્રુપની અંદરના વિગતવાર રેકોર્ડને કયા ક્રમમાં રજૂ કરવા તે નક્કી કરી શકાય છે. આ પગલું રિપોર્ટના ડેટાને તાર્કિક રીતે વ્યવસ્થિત અને અનુસરવામાં સરળ બનાવવા માટે ખૂબ જ મહત્વપૂર્ણ છે. *Sort by* ડ્રોપડાઉન મેનૂમાં *FirstName* ફિલ્ડ પસંદ કરો. ત્યારપછી, આકૃતિ 4.24માં બતાવ્યા મુજબ રિપોર્ટ વિઝાર્ડના *Choose layout* પગલા પર આગળ વધવા માટે *Next* પર ક્લિક કરો.



આકૃતિ 4.24 : લેઆઉટ પસંદ કરવો

રિપોર્ટ વિઝાર્ડનું પાંચમું પગલું આપણને રિપોર્ટના લેઆઉટ (layout)ને કસ્ટમાઇઝ કરવાની મંજૂરી આપે છે. તમે વિવિધ રંગ સંયોજનોમાંથી પસંદગી કરી શકો છો અને *Layout of data* પસંદગી યાદીનો ઉપયોગ કરીને ફિલ્ડની સ્થિતિ અને લખાણની ગોઠવણમાં ફેરફાર કરી શકો છો. આ પગલું રિપોર્ટનું ઓરિએન્ટેશન *Landscape* (આડું) અથવા *Portrait* (ઊભું) સેટ કરવા અને *Layout of headers and footers* યાદી બોક્સમાંથી એક સામાન્ય હેડર અને ફૂટર લેઆઉટ પસંદ કરવાની પણ મંજૂરી આપે છે.

એકવાર ઇચ્છિત રિપોર્ટ લેઆઉટ પસંદ કરવામાં આવે, પછી રિપોર્ટ વિઝાર્ડના અંતિમ પગલાં – *Create Report* પર આગળ વધવા માટે *Next* પર ક્લિક કરો, જે આકૃતિ 4.25માં દર્શાવવામાં આવ્યું છે.



આકૃતિ 4.25 : રિપોર્ટ તૈયાર કરવો

રિપોર્ટ વિઝાર્ડના અંતિમ પગલામાં નીચેના વિકલ્પો રજૂ કરવામાં આવે છે:

- **Title of Report:** રિપોર્ટને નામ આપવું.
- **What kind of report do you want to create?:** અહીં બે વિકલ્પો રજૂ કરવામાં આવશે:
  - **Static report:** ડેટાબેઝમાંના ટેબલ અથવા ક્વેરી નવી માહિતી સાથે અપડેટ કરવામાં આવે તો પણ એકવાર બનાવ્યા પછી સ્ટેટિક રિપોર્ટમાં પ્રદર્શિત ડેટા બદલાતો નથી.
  - **Dynamic report:** જ્યારે પણ ડાયનેમિક રિપોર્ટ ખોલવામાં આવે અથવા રિફ્રેશ કરવામાં આવે, ત્યારે તે ડેટાબેઝમાંના વર્તમાન ડેટાને પ્રતિબિંબિત કરી રિપોર્ટને અદ્યતન બનાવે છે.

રિપોર્ટ બનાવવાનું પૂર્ણ કરવા માટે *Finish* પર ક્લિક કરો. આકૃતિ 4.26 અંતિમ રિપોર્ટને સમાન દૃશ્ય દર્શાવે છે.

Author:
Date: 8/9/25

ClassID 11

StudentID	FirstName	MiddleName	LastName	BirthDate	Gender	Address
102	Kavya	B	Pandya	01/20/02	F	Ahmedabad
105	Pari	V	Naik	01/21/01	F	Mumbai
101	Sunny	A	Jain	07/09/25	M	Jaipur

ClassID 12

StudentID	FirstName	MiddleName	LastName	BirthDate	Gender	Address
104	Anthony	R	Gomes	07/12/01	M	Goa
103	Rafiq	M	Memon	02/11/01	M	Hyderabad

આકૃતિ 4.26 : Student ટેબલ પર આધારિત રિપોર્ટ

ખાસ નોંધ લો કે આ રિપોર્ટ Student ટેબલના ડેટાને ClassID દ્વારા ગ્રુપ કરે છે અને તેને FirstName દ્વારા સૉર્ટ કરે છે. એકવાર રિપોર્ટ બની જાય, પછી તેની ડિઝાઈનને રિપોર્ટના નામ પર રાઈટ-ક્લિક કરી, *Edit* વિકલ્પ પસંદ કરવાથી સરળતાથી સુધારી શકાય છે. રીપોર્ટના નામ પર રાઈટ ક્લિક કરીને Edit વિકલ્પ પસંદ કરો. આમ કરવાથી *Report Design View* ખુલશે, જ્યાં ફોર્મની જેમ જ ફેરફારો કરી શકાય છે.

File → *Print* વિકલ્પ દ્વારા રિપોર્ટની હાર્ડકૉપી (છાપેલી નકલ) પણ મેળવી શકાય છે. વૈકલ્પિક રીતે, PDF કૉપી બનાવવા માટે, સ્ટાન્ડર્ડ ટૂલબાર પર આપવામાં આવેલા *Export directly as PDF* બટન પર ક્લિક કરી શકાય છે.

સારાંશ

કોઈપણ અસરકારક ડેટાબેઝ મેનેજમેન્ટ સિસ્ટમ માટે ફોર્મ (Form) અને રિપોટ (Report) ખૂબ જ મહત્ત્વપૂર્ણ છે. તેઓ જુદી-જુદી છતાં એકબીજાને પૂરક ભૂમિકાઓ ભજવે છે, જે અંતિમ-યુઝર્સ માટે રો (Raw) ડેટાને સુગમ, ઉપયોગી અને મૂલ્યવાન માહિતીમાં પરિવર્તિત કરે છે. આ પ્રકરણમાં ડેટાબેઝ માટે ફોર્મ અને રિપોર્ટ બંનેની રચનાનો અભ્યાસ કર્યો. આપણે જોયું કે ફોર્મ યુઝર ઇન્ટરફેસ તરીકે કાર્ય કરે છે, જે ટેબલ અથવા ક્વેરીમાં ફેરફાર કર્યા વિના ડેટાબેઝ સાથે સીધી ક્રિયા-પ્રતિક્રિયા કરવાની મંજૂરી આપે છે. આપણે રિપોર્ટ પણ તૈયાર કર્યા, જે ડેટાને વ્યવસ્થિત અને અર્થપૂર્ણ રીતે રજૂ કરવા અને તેનો સારાંશ આપવા માટે આવશ્યક સાધન છે. તે રો (Raw) ડેટાને નિર્ણય લેવા માટે મહત્ત્વપૂર્ણ એવા કાર્યવાહી કરી શકાય તેવા ડોક્યુમેન્ટમાં ફેરવે છે. આ સાથે, આપણે ડેટાબેઝ મેનેજમેન્ટની મૂળભૂત બાબતોનો અભ્યાસ પૂરો કરીએ છીએ.

સ્વાધ્યાય

1. ડેટાબેઝ ડેટા સાથે ક્રિયા-પ્રતિક્રિયા (interacting) કરવા માટે ફોર્મ શા માટે વધુ અનુકૂળ છે?
2. બેઝમાં ફોર્મની રચના કરવા માટે કયા વિકલ્પો ઉપલબ્ધ છે?
3. Form Wizardના કોઈપણ ત્રણ પગલાં સમજાવો.
4. ફોર્મનું બેકગ્રાઉન્ડ કેવી રીતે બદલી શકાય?
5. ફોર્મમાં આવેલ કંટ્રોલનું માપ કેવી રીતે બદલી શકાય છે?

6. રિપોર્ટ એટલે શું? ડેટાની રજૂઆતમાં તેની ભૂમિકા શું છે?

7. Form Design viewમાં કંટ્રોલની Label પ્રોપર્ટીને સુધારવા માટેના પગલાં જણાવો.

8. સ્ટેટિક (Static) અને ડાયનેમિક (Dynamic) રિપોર્ટ વચ્ચે શું તફાવત છે?

9. ફોર્મના નેવિગેશન બાર (navigation bar) પર કયા બટન દર્શાવવામાં આવે છે?

10. ફોર્મમાં કોઈ ચોક્કસ રેકોર્ડ કેવી રીતે શોધી શકાય છે?

11. સાચું કે ખોટું જણાવો.

(1) ફોર્મ વ્યૂમાં ડેટામાં ફેરફાર કરવો શક્ય છે.

(2) ટેબલ અથવા ક્વેરી પરથી ફોર્મ બનાવી શકાય છે.

(3) ફોર્મ વિઝાર્ડ દ્વારા ફોર્મ બનાવતી વખતે સબ-ફોર્મ બનાવવું જરૂરી છે.

(4) રિપોર્ટ્સ સામાન્ય રીતે રીડ-ઓન્લી હોય છે.

(5) રિપોર્ટ્સને પીડીએફ ફાઇલ સ્વરૂપે એક્સપોર્ટ કરી શકાય છે.

12. ખાલી જગ્યા પૂરો.

(1) ફોર્મ ____________ અથવા ____________ નો ઉપયોગ કરીને બનાવી શકાય છે.

(2) ____________ નો ઉપયોગ કરીને ફોર્મની રચના સરળ અને ઝડપી થઈ શકે છે.

(3) ફોર્મ વિઝાર્ડમાં કુલ ____________ પગલાં આપવામાં આવ્યા છે.

(4) ટેબલના આગળ કે પાછળના રેકોર્ડને જોવા માટે ____________ નો ઉપયોગ કરવામાં આવે છે.

(5) ____________ એ પ્રાપ્ત કરીલે માહિતીને આકર્ષક અને વ્યવસ્થિત ફોર્મેટમાં રજૂ કરવાની એક ઉત્તમ રીત છે.

13. બહુવિકલ્પી પ્રશ્નો. સૌથી યોગ્ય જવાબ પસંદ કરો.

(1) કયું ઘટક ટેબલમાં સંગ્રહિત ડેટા સાથે સંવાદ (interaction) કરી શકે છે?

(a) ડિઝાઇન વ્યૂ (Design view) (b) સંબંધ (Relationship)

(c) ફોર્મ (Form) (d) રિપોર્ટ (Report)

(2) ફોર્મ બનાવવા માટે કયા ડેટાબેઝ ઑબ્જેક્ટનો ઉપયોગ કરી શકાય છે?

(a) ટેબલ (Table) (b) ક્વેરી (Query)

(c) રિપોર્ટ (Report) (d) ટેબલ અથવા ક્વેરી

(3) ફોર્મ વિઝાર્ડ (Form Wizard)માં કેટલા પગલાં સામેલ છે?

(a) 6 (b) 7 (c) 8 (d) 10

(4) ફોર્મ વિઝાર્ડના કયા પગલામાં આપણે મૂળભૂત ટેબલ અથવા ક્વેરી પસંદ કરીએ છીએ?

(a) પ્રથમ (b) ત્રીજું (c) ચોથું (d) આઠમું

(5) ટેબલના તમામ ઉપલબ્ધ ફિલ્ડને ફોર્મમાં ઉમેરવા માટે કયા બટનનો ઉપયોગ કરવામાં આવે છે?

(a) > (b) >> (c) < (d) <<

(6) લેબલ અથવા ટેક્સ્ટબોક્સમાંથી કોઈ એક કંટ્રોલ પસંદ કરતી વખતે કઈ કી દબાવી રાખવી જોઈએ?

(a) શિફ્ટ (Shift) (b) કંટ્રોલ (Control) (c) ઓલ્ટર (Alter) (d) સ્પેસ (Space)

(7) ફોર્મ ડિઝાઇન વ્યૂમાં કઈ પ્રોપર્ટીનો ઉપયોગ કંટ્રોલમાં ટૂલટીપ ઉમેરવા માટે થાય છે?

(a) ટૂલટીપ (Tooltip)

(b) એડ ટેક્સ્ટ (Add text)

(c) હેલ્પ ટેક્સ્ટ (Help text)

(d) આવી કોઈ પ્રોપર્ટી નથી

(8) રિપોર્ટ વિઝાર્ડમાં રિપોર્ટમાંથી પસંદ કરેલ ફિલ્ડને દૂર કરવા માટે કયા બટનનો ઉપયોગ થાય છે?

(a) > (b) >> (c) < (d) <<

(9) રિપોર્ટ વિઝાર્ડના કયા પગલામાં આપણે પેજ લેઆઉટ ઓરિએન્ટેશન પસંદ કરીએ છીએ?

(a) ત્રીજું (b) ચોથું (c) પાંચમું (d) છઠ્ઠું

(10) કયા પ્રકારનો રિપોર્ટ મૂળભૂત ટેબલ અપડેટ થાય તો પણ બદલાતો નથી?

(a) સ્ટેટિક (Static) (b) ડાયનેમિક (Dynamic)

(c) પ્રોગ્રામેટિક (Programmatic) (d) ઑટોમેટિક (Automatic)

પ્રાયોગિક સ્વાધ્યાય

1. અગાઉના પ્રકરણોમાં બનાવેલા તમામ ટેબલ માટે ફોર્મ બનાવો.
2. નિર્દિષ્ટ જરૂરિયાતો અનુસાર ફોર્મમાં ફેરફાર કરો:
   - કોઈપણ એક ફોર્મનું બેકગ્રાઉન્ડ બદલો.
   - કોઈપણ એક ફોર્મમાં ઓછામાં ઓછા બે કંટ્રોલ્સ માટે ટૂલટીપ દાખલ કરો.
   - ફોર્મમાં રહેલા 'Date' પ્રકારના કંટ્રોલને ડ્રોપડાઉન મેનુમાં બદલો.
   - કોઈપણ એક ફોર્મમાં ઓછામાં ઓછા બે કંટ્રોલ્સના લેબલ બદલો.
   - જરૂરિયાત મુજબ તમામ ફોર્મ પરના તમામ કંટ્રોલનું માપ યોગ્ય રીતે બદલો.
3. વિદ્યાર્થીનું નામ, જન્મતારીખ અને સરનામું આપીને વિદ્યાર્થીના રેકોર્ડને શોધો.
4. બે અલગ-અલગ ટેબલના આધારે સ્ટેટિક (static) અને ડાયનેમિક (dynamic) રિપોર્ટ તૈયાર કરો.

5 સમસ્યા ઉકેલ : ફ્લોચાર્ટ અને અલ્ગોરિધમ

પરિચય

પથ્થર યુગથી લઈને આધુનિક ટેક્નોલોજીના યુગ સુધી, મનુષ્યે જીવનભર દરરોજ વિવિધ પ્રકારની સમસ્યાઓનો સામનો કરવો પડ્યો છે. આમાં જુદી જુદી વસ્તુઓની ગણતરી કરવી, જુદા જુદા ભાવની અને જુદા જુદા જથ્થામાં ખરીદેલી વસ્તુઓના બિલનો સરવાળો કરવો જેવા ખૂબ જ સરળ કાર્યોથી માંડીને ઘરવપરાશની વસ્તુઓ બનાવતી એક મોટી ફેક્ટરીના સ્ટોરમાં સેંકડો વસ્તુઓના સ્ટોકનું સંચાલન કરવા જેવી જટિલ સમસ્યાઓનો સમાવેશ થાય છે. લોકો તેમના રોજિંદા અને સરળ પ્રશ્નોને કેવી રીતે ઉકેલવા તે માટે ટેવાયેલા છે. પરંતુ જ્યારે મોટી અને જટિલ સમસ્યાઓની વાત આવે, જ્યાં સમયસર અને ચોકસાઈ સાથે કાર્ય પૂર્ણ કરવું ફરજિયાત હોય, ત્યારે કમ્પ્યૂટર મદદ માટે આવે છે.

કમ્પ્યૂટર વિજ્ઞાન આપણને સમસ્યાઓ ઉકેલવાની અનુકૂળ રીત પૂરી પાડે છે અથવા આપણે કહી શકીએ કે સમસ્યાનો ઉકેલ (નિરાકરણ) એ કમ્પ્યૂટર વિજ્ઞાનના કેન્દ્રમાં છે. આપણે જે કામ કરવા માંગીએ છીએ તેના માટે પ્રોગ્રામ લખીને કરી શકાય છે. એકવાર જે તે કામ માટે પ્રોગ્રામ લખાઈ જાય, પછી તે સમસ્યાને વારંવાર ઉકેલ્યા વિના લાંબા સમય સુધી તેને વાપરી શકાય છે. ઉદાહરણ તરીકે: બેંકિંગ કામગીરી કરવી, ઓનલાઈન વસ્તુની પસંદગી કરવી અથવા ઓનલાઈન વસ્તુઓનો ઓર્ડર આપવો અને UPI (Unified Payment Interface – યુનીફાઈડ પેમેન્ટ ઇન્ટરફેસ)નો ઉપયોગ કરીને ઓનલાઈન ચુકવણી કરવી.

આ પ્રકરણમાં, આપણે સમસ્યા ઉકેલના ખ્યાલો સમજવા પર અને ત્યારબાદ બે વ્યાપકપણે ઉપયોગમાં લેવાતી પદ્ધતિઓ, (1) ફ્લોચાર્ટ અને (2) અલ્ગોરિધમ – ને સંખ્યાબંધ ઉદાહરણો સાથે શીખવા પર ધ્યાન કેન્દ્રિત કરીશું. આ પ્રકરણ શીખ્યા પછી, વિદ્યાર્થીઓ આપવામાં આવેલ સમસ્યા કે પ્રશ્ન સમજી શકશે અને સમસ્યાને સ્પષ્ટતા સાથે ઉકેલવા માટે ફ્લોચાર્ટ બનાવી શકશે અને/અથવા એલ્ગોરિધમ લખી શકશે.

સમસ્યા ઉકેલની સામાન્ય સમજ (Overview of Problem Solving)

સમસ્યા ઉકેલએ કોઈપણ સમસ્યાનું ઉકેલ શોધવા માટે અપનાવાતી ક્રમબદ્ધ પગલાવાર પ્રક્રિયા છે. તેનો અર્થ છે કે, સમસ્યાને સંપૂર્ણપણે સમજવી અને તેને નાના પગલાઓમાં યોગ્ય ક્રમમાં ગોઠવવી. સમસ્યા ઉકેલના મુખ્ય પગલાં નીચે મુજબ હોઈ શકે છે:

- સમસ્યાને તમામ વિગતો સાથે વ્યાખ્યાયિત કરવી
- સમસ્યાને સમજવી
- ઈનપુટ્સ ઓળખવા અથવા મેળવવા
- સમસ્યા હલ કરવા માટે જરૂરી પગલાં લેવા
- આઉટપુટનું પરીક્ષણ કરવું અને માન્ય કરવા

"સમસ્યા હલ કરવા માટે જરૂરી પગલાં લેવા" પગલું સમસ્યા પ્રમાણે બદલાય છે. સરળ સમસ્યાઓ માટે તેમાં ફક્ત થોડા પગલાં જ ક્રમ પ્રમાણે હોય છે. પરંતુ મધ્યમથી મોટી સમસ્યાઓ માટે તેમાં નિર્ણયો લેવાના પગલાં સામેલ થાય છે જેમાં વૈકલ્પિક માર્ગ અપનાવવા પડે, અથવા કેટલાક પગલાં વારંવાર પુનરાવર્તિત કરવા પડે. તેથી આ તબક્કામાં મુખ્યત્વે નીચેના પગલાં વિવિધ સંયોજન સાથે સામેલ થાય છે:

- **ક્રમ (Sequence) :** એક પછી એક પગલું
- **નિર્ણય લેવો (Decision Making) :** બે કે વધુ વિકલ્પમાંથી કોઈ એક પસંદ કરવો
- **પુનરાવર્તન/લૂપિંગ (Repetiition/Looping) :** કેટલાક પગલાં વારંવાર કરવાનું, એટલે કે એકથી વધારે વખત કરવા.

ચાલો હવે ઉદાહરણની મદદથી તેને સમજીએ. બે સંખ્યાનો સરવાળો કરવા માટેના પગલાં નીચે મુજબ છેઃ

1. બે સંખ્યાઓ N1 અને N2 સ્વીકારો
2. તેમનો સરવાળો કરોઃ SUM = N1 + N2
3. પરિણામ એટલે કે SUM પ્રિન્ટ કરો

ઉપરોક્ત સમસ્યા સંપૂર્ણપણે ક્રમબદ્ધ છે અને તેને ઉકેલવા માટે આપણે પગલાં 1, 2 અને 3 ને ક્રમ પ્રમાણે અનુસરવા પડે છે.

બે સંખ્યાઓમાંથી મોટી સંખ્યા શોધવાના પગલાં નીચે મુજબ છેઃ

1. બે સંખ્યાઓ N1 અને N2 સ્વીકારો
2. તેમની સરખામણી કરો, જો N1 > N2 હોય તો N1 મોટી છે, પછી સીધું પગલું-4 પર જાઓ
3. નહીંતર N2 મોટી છે
4. મોટી સંખ્યા પ્રિન્ટ કરો

નિરીક્ષણ કરો કે ઉપરોક્ત સમસ્યામાં પગલું-2 પર નિર્ણય સામેલ છે. પગલું-1 માં N1 અને N2 સ્વીકાર્યા પછી, પગલું-2 પહેલા તેમની ">" દ્વારા સરખામણી કરે છે. જો N1 મોટી હોય તો તે પગલું-3 છોડીને સીધું પગલું-4 પર જઈ N1 પ્રિન્ટ કરે છે. જો સરખામણી ખોટી નીવડે તો પગલું-2 નો બાકીનો ભાગ છોડી દેવામાં આવે છે, એટલે કે N2 મોટી હોવાથી પગલું-3 કરવામાં આવે છે અને ત્યારબાદ પગલું-4 N2 પ્રિન્ટ કરે છે. અહીં મુખ્ય વાત એ છે કે આપણે N1 અને N2ની તુલનાના આધારે બે વિકલ્પોમાંથી એક પસંદ કરીએ છીએ.

1 થી 10 સુધીના સંખ્યાઓનો સરવાળો કરવા માટેના પગલાં નીચે મુજબ છેઃ

1. SUM = 0 થી પ્રારંભ કરો
2. I = 1 નક્કી કરો
3. પગલું-4 અને પગલું-5 ને 10 વખત પુનરાવર્તિત કરો
4. SUM માં I ઉમેરો
5. I ને 1 થી વધારો
6. SUM પ્રિન્ટ કરો

ઉપરોક્ત સમસ્યાના પગલાં ધ્યાનથી જુઓ. પગલું-1માં SUM ને 0 થી શરૂ કરવામાં આવે છે અને પગલું-2માં I ને પ્રથમ મૂલ્ય 1 આપવામાં આવે છે. ત્યારબાદ પગલું-4માં I ની વર્તમાન મૂલ્ય SUM સાથે ઉમેરવામાં આવે છે અને પગલું-5માં I ને 1થી વધારો કરવામાં આવે છે. આ પ્રક્રિયા કુલ 10 વાર થાય છે, એટલે કે I ના મૂલ્યો ક્રમશઃ 1, 2, 3, 5, ..., 10 થાય છે અને દરેક વખતે તે SUM સાથે ઉમેરાય છે. આ રીતે અંતે SUM = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 = 55 મળે છે, જેને પગલું-6 દ્વારા પ્રિન્ટ કરવામાં છે.

ઉપરોક્ત ઉદાહરણમાં સમસ્યા ઉકેલવાના પ્રક્રિયા પગલાંના ત્રણેય મહત્વના ઘટકોનો ઉપયોગ દર્શાવવામાં આવ્યો છે. વાસ્તવિક જીવનમાં આપણી સમક્ષ આવતી સમસ્યાઓ ઉપરના ઉદાહરણ જેટલી સરળ નથી. તેમાં અનેક ગણતરીઓ સામેલ હોય છે અને તે જટિલ હોય છે. આવી સમસ્યાઓ ઉકેલવા માટે આપણે ઉપર જણાવેલ ત્રણ પગલાં - ક્રમ (Sequence), નિર્ણય લેવો (Decision Making) અને પુનરાવર્તન (Repetiition) - નો વિવિધ સંયોજનમાં ઉપયોગ કરવો પડે છે, જે સમસ્યાની જરૂરિયાત પર આધારિત હોય છે.

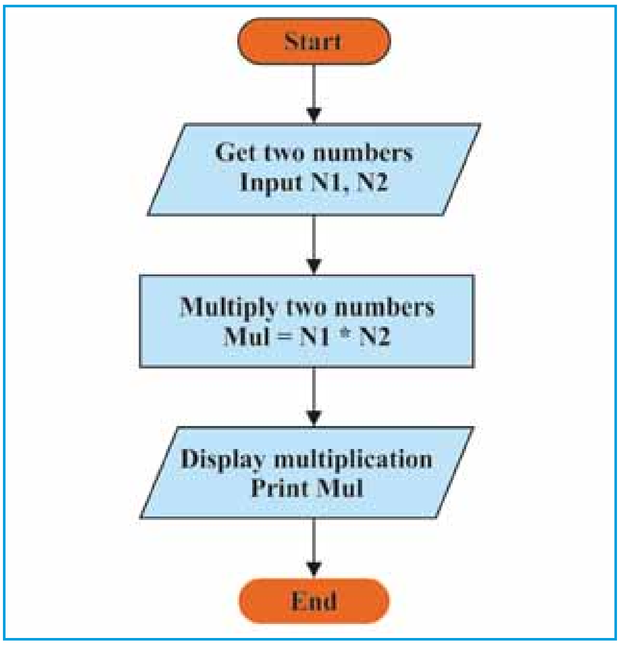
આગળનો વિભાગ તમને બે વ્યાપકપણે વપરાતા સમસ્યા ઉકેલવાના ઉપાયો - ફ્લોચાર્ટ અને અલ્ગોરિધમ - સાથે પરિચિત કરાવશે. આપણે સમજીશું કે કેવી રીતે આ બંનેનો ઉપયોગ કરીને ઘણી ગણતરીઓ ધરાવતી સમસ્યાઓ ઉકેલી શકાય અને અંતે ફ્લોચાર્ટ તથા અલ્ગોરિધમની તુલના પણ કરીશું.

ફ્લોચાર્ટ અને તેના પ્રતીકો (Flowchart and its Symbols)

ચાલો, આપણે ફ્લોચાર્ટને વિગતવાર સમજીએ, જેમાં ફ્લોચાર્ટ બનાવવા માટે ઉપયોગમાં લેવાતા વિવિધ પ્રતીકો તેમના અર્થ અને ઉપયોગનો સમાવેશ થાય છે.

ફ્લોચાર્ટ (Flowchart)

ફ્લોચાર્ટ એ સમસ્યાના ઉકેલનું ચિત્રાત્મક નિરૂપણ છે. તે કોઈ પ્રક્રિયા અથવા કાર્યના પગલાંને સરળ અને દ્રશ્યમાન રીતે દર્શાવે છે. ફ્લોચાર્ટમાં જુદા જુદા પગલાં માટેની વિવિધ ક્રિયાઓ દર્શાવવા માટે અંડાકાર (ovals), લંબચોરસ (rectangles) અને ડાયમંડ (diamonds) જેવા વિવિધ પ્રતીકોનો ઉપયોગ થાય છે. વિવિધ પગલાંને જોડવા માટે એરોનો ઉપયોગ થાય છે. ફ્લોચાર્ટ દ્રશ્ય નિરૂપણનો ઉપયોગ કરતો હોવાથી, તમામ સંભવિત માર્ગો અને પગલાં સ્પષ્ટપણે દેખાય છે, અને તે ઉકેલની સમજણ ખૂબ જ સરળ બનાવે છે. આકૃતિ 5.1 બે સંખ્યાઓનો ગુણાકાર કરવા માટેનો ફ્લોચાર્ટ દર્શાવે છે.

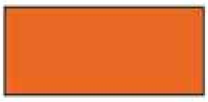


આકૃતિ 5.1 : ગુણાકાર કરવા માટેનો ફ્લોચાર્ટ

આકૃતિ 5.1માં દર્શાવ્યા મુજબ, ફ્લોચાર્ટ “Start” (શરૂઆત) થી શરૂ થાય છે અને “End” (અંત)થી સમાપ્ત થાય છે. બીજું પગલું, જે સમાંતર ચતુષ્કોણ (parallelogram) દ્વારા દર્શાવવામાં આવ્યું છે, તે બે સંખ્યાઓ N1 અને N2 સ્વીકારે છે. ત્રીજું પગલું, લંબચોરસનો ઉપયોગ કરીને તેમનો ગુણાકાર કરવાની પ્રક્રિયા દર્શાવે છે : Mul = N1 * N2. ચોથું પગલું ફરીથી સમાંતર ચતુષ્કોણ છે, જે આઉટપુટ એટલે કે બે આપેલી સંખ્યાઓનો ગુણાકાર (Mul) પ્રિન્ટ કરવા માટે છે. આ ફ્લોચાર્ટ ખૂબ જ સરળ છે કારણ કે સમસ્યા ક્રમિક છે. પ્રવાહ એટલે કે પગલાંનો ક્રમ સ્પષ્ટપણે દેખાય છે, કારણ કે દરેક પગલાં પછીનો એરો આગલા પગલા તરફ નિર્દેશ કરે છે. નિર્ણય લેવા અને પુનરાવર્તન ધરાવતી વિવિધ સમસ્યાઓ માટે ફ્લોચાર્ટ વિકસાવીએ તે પહેલાં, ચાલો આપણે ફ્લોચાર્ટમાં ઉપયોગમાં લેવાતા તમામ પ્રતીકોને સમજીએ.

ફ્લોચાર્ટ પ્રતીકો (Flowchart Symbols)

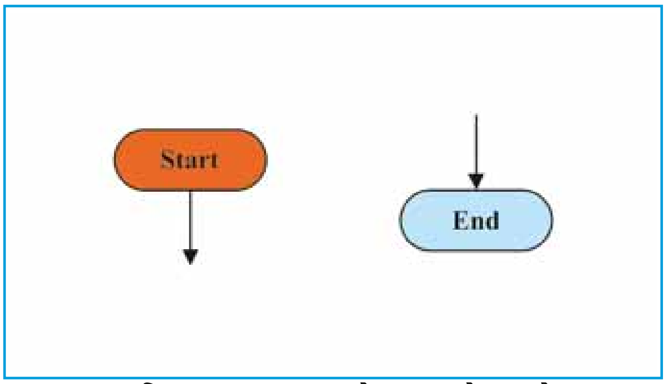
આકૃતિ 5.2માં ફ્લોચાર્ટ બનાવવા માટે ઉપયોગમાં લેવાતા વિવિધ પ્રતીકો દર્શાવેલ છે. આકૃતિમાં તેને સમજવા માટે પ્રતીક, તેનું નામ અને ટૂંકું વર્ણન

Symbol	Name	Description
	Oval	Represents start or end point
	Arrows	Connect two symbols with direction
	Parallelogram	Represents input or output
	Rectangle	Represents process, used to compute values or perform operations
	Diamond	Represents decision making, provides alternative paths
	Circle	Used as connector, used to connect different parts of flowchart

આકૃતિ 5.2 : ફ્લોચાર્ટ પ્રતીકો

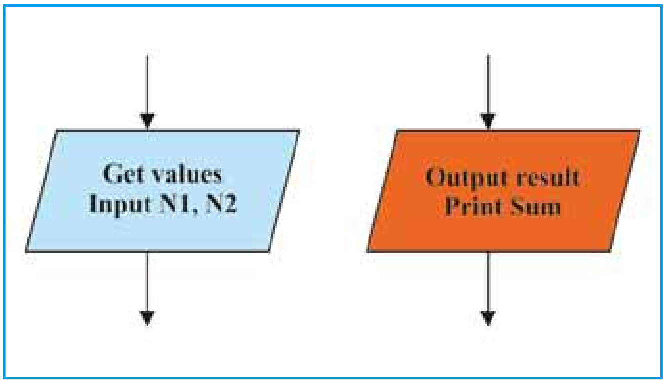
આપેલ છે. ચાલો હવે ફ્લોચાર્ટમાં તેમના ઉપયોગ સાથે તે દરેકને એક પછી એક સમજીએ.

શરૂઆત/અંત (Start/End) : Start અને End એ અંડાકાર (oval) દ્વારા રજૂ થતા ટર્મિનલ પ્રતીકો છે, જે અનુક્રમે ફ્લોચાર્ટની શરૂઆત અને અંત દર્શાવે છે. Start ફ્લોચાર્ટની શરૂઆત દર્શાવે છે અને તે હંમેશાં પ્રથમ પ્રતીક હોય છે. End ફ્લોચાર્ટની પૂર્ણતા અથવા સમાપ્તિ દર્શાવે છે અને તે હંમેશાં છેલ્લું પ્રતીક હોય છે. આકૃતિ 5.1માં બતાવ્યા પ્રમાણે, પ્રક્રિયા અથવા કાર્ય કરવાનાં પગલાં Start અને End પ્રતીકોની વચ્ચે સમાવિષ્ટ હોય છે. આકૃતિ 5.3 એરો સાથે Start અને End પ્રતીકના ઉપયોગને દર્શાવે છે.



આકૃતિ 5.3 : Start અને End નો ઉપયોગ

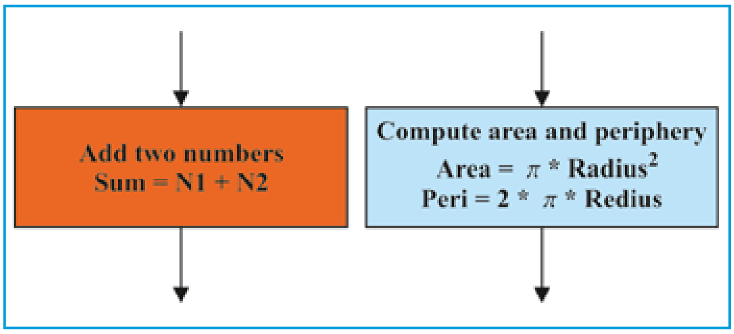
એરો (Arrow) : એરોનો ઉપયોગ પ્રવાહ રેખાઓ દર્શાવવા માટે થાય છે. ફ્લોચાર્ટમાં આગળનું પ્રતીક ક્યાં દોરેલું છે તેના આધારે આપણે તમામ દિશાઓમાં એરોનો ઉપયોગ કરી શકીએ છીએ. આકૃતિ 5.1માં બતાવ્યા પ્રમાણે, સૌથી સામાન્ય ઉપયોગ નીચે તરફનો એરો છે. એરો ફ્લોચાર્ટની અંદર અમલનો માર્ગ નક્કી કરવા માટે વપરાતું સૌથી મહત્વપૂર્ણ પ્રતીક છે, જે સૂચવે છે કે નિયંત્રણ હવે પછી ક્યાં જાય છે. જો કે, ખાસ કરીને એક જ પૃષ્ઠ પર ઘણાં પગલાં ધરાવતા મોટા ફ્લોચાર્ટમાં જરૂર પડે ત્યારે, ડાબી કે જમણી બાજુના એરોનો પણ ઉપયોગ થાય છે.



આકૃતિ 5.4 : ઈનપુટ/આઉટપુટ પ્રતીકોનો ઉપયોગ

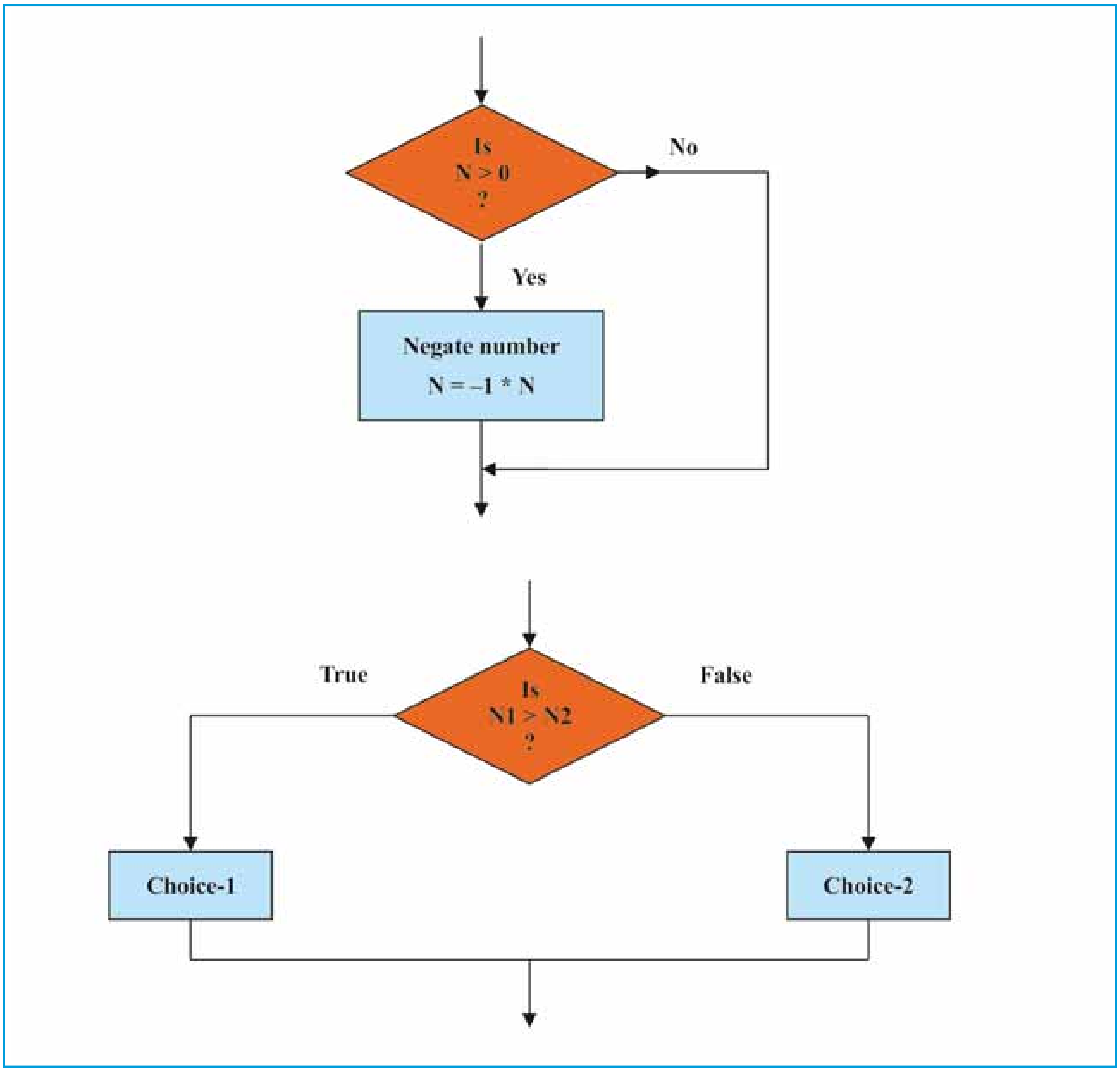
ઈનપુટ/આઉટપુટ (Input/Output) : ઈનપુટ અને આઉટપુટને આકૃતિ 5.2માં બતાવ્યા પ્રમાણે સમાંતર ચતુષ્કોણ દ્વારા રજૂ કરવામાં આવેલ છે. ઈનપુટ પ્રતીકનો ઉપયોગ યુઝર અથવા વાસ્તવિક દુનિયા તરફથી મૂલ્યો અથવા ડેટા મેળવવા માટે થાય છે, જેના પર આગળની પ્રક્રિયા કરવામાં આવે છે. આઉટપુટ પ્રતીક દર્શાવે છે કે ઉકેલ દ્વારા ઉત્પન્ન થયેલું પરિણામ યુઝર અથવા વાસ્તવિક દુનિયાને તેમના ઉપયોગ માટે મોકલવામાં આવે છે. આકૃતિ 5.4 ઈનપુટ અને આઉટપુટ પ્રતીકોનો ઉપયોગ દર્શાવે છે. ઈનપુટ પ્રતીકને "Input N1 and N2" દ્વારા અને આઉટપુટ પ્રતીકને "Print Sum" દ્વારા દર્શાવવામાં આવ્યું છે.

પ્રક્રિયા (Process) : તેના માટે લંબચોરસ (rectangle)નો ઉપયોગ થાય છે. જ્યારે પણ આપણે કોઈ મૂલ્યોની ગણતરી કે પ્રક્રિયા કરવા માંગીએ છીએ, ત્યારે તેનો ઉપયોગ થાય છે. પ્રક્રિયાનું પગલું વિવિધ ગણતરીઓ કરવા માટે વાપરી શકાય છે, જેમ કે અંકગણિતીય ક્રિયાઓ, તાર્કિક ક્રિયાઓ, આપેલા મૂલ્યોના આધારે સૂત્રોનું મૂલ્યાંકન વગેરે. લંબચોરસ દ્વારા રજૂ કરાયેલ એક જ 'પ્રક્રિયા બોક્સ'માં પગલાંના સમૂહનો ઉપયોગ કરીને એક કરતાં વધુ મૂલ્યોની ગણતરી કરી શકાય છે. આકૃતિ 5.5 પ્રક્રિયાના પગલાંના બે ઉદાહરણો દર્શાવે છે: પહેલું બે સંખ્યાઓનો સરવાળો કરવા માટે (Sum = N1 + N2) અને બીજું વર્તુળનું ક્ષેત્રફળ અને પરિઘની ગણતરી કરવા માટે.



આકૃતિ 5.5 : પ્રક્રિયા માટેના પ્રતિકનો ઉપયોગ

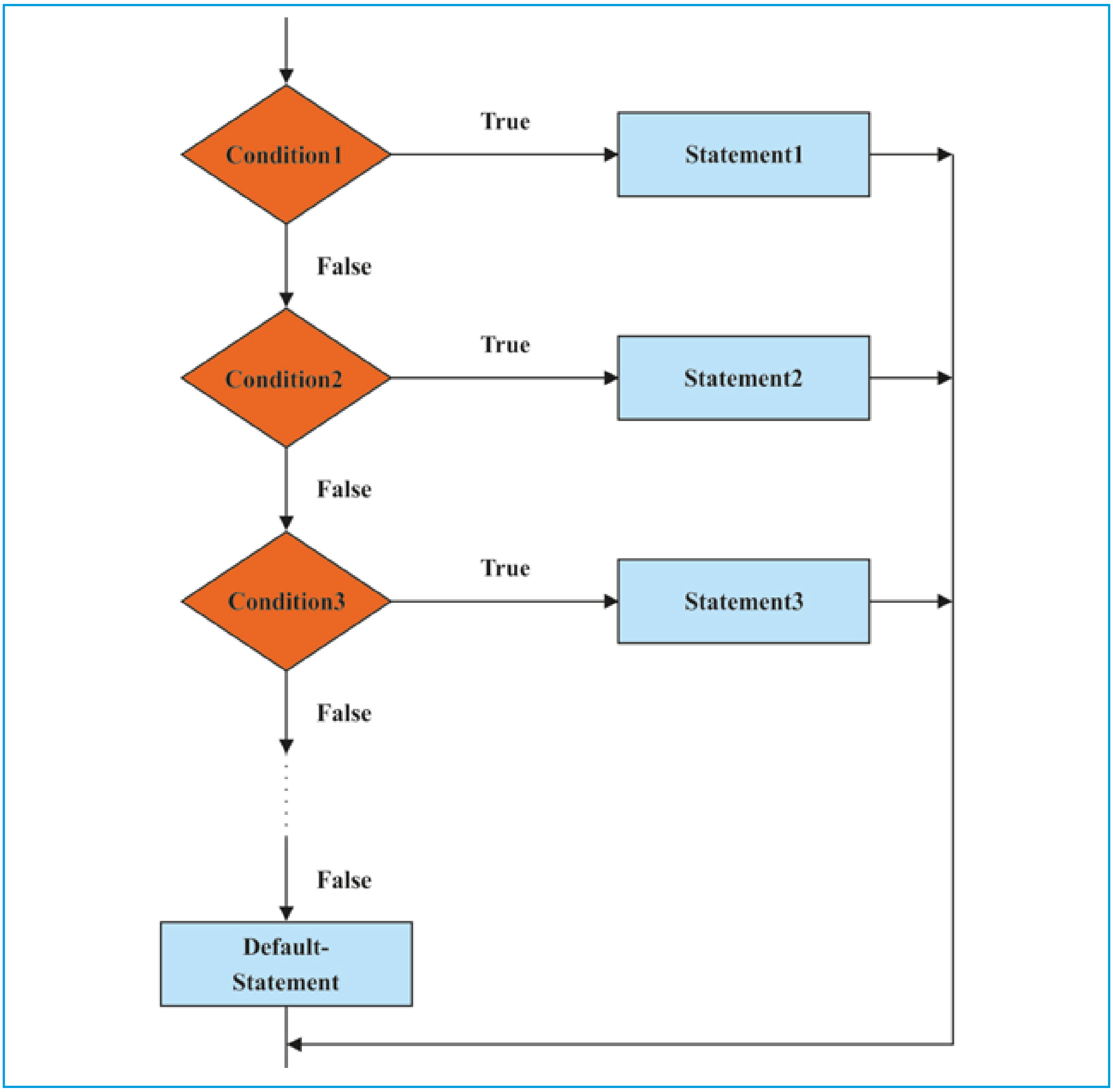
નિર્ણય (Decision) : નિર્ણય લેવો એ સૌથી મહત્વપૂર્ણ છે અને માનવીઓ દ્વારા તેમના જીવનમાં નિયમિતપણે કરવામાં આવે છે. ફ્લોચાર્ટમાં નિર્ણય માટે ડાયમંડ આકારના પ્રતીકનો ઉપયોગ થાય છે. આપણે ડાયમંડની અંદર સામાન્ય રીતે શરતનો ઉપયોગ કરીને વિકલ્પોમાંથી એક નક્કી કરીએ છીએ. ધારો કે, જો સંખ્યા 0 કરતાં મોટી હોય તો આપણે તેને ઋણ સંખ્યા બનાવીએ છીએ, નહીં તો કોઈ પગલું લેતા નથી. બીજું ઉદાહરણ બે સંખ્યાઓની તુલના કરે છે અને જો પ્રથમ સંખ્યા બીજી સંખ્યા કરતાં મોટી હોય તો આપણે વિકલ્પ-1 (choice-1) લઈએ છીએ, નહીં તો આપણે વિકલ્પ-2 (choice-2) લઈએ છીએ. આકૃતિ 5.6માં આ દર્શાવ્યું છે.



આકૃતિ 5.6 : નિર્ણય માટેના પ્રતિકનો ઉપયોગ

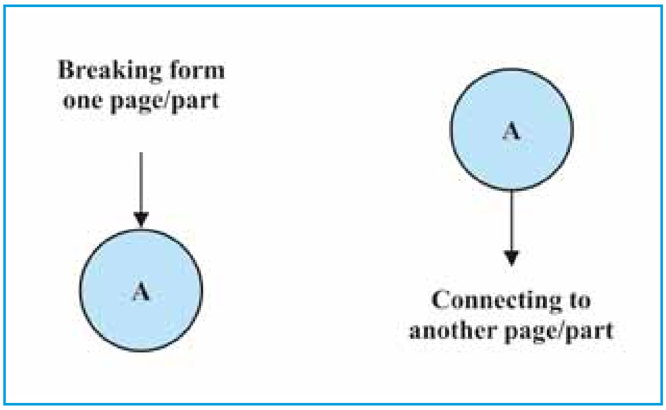
કેટલીકવાર, આપણી પાસે બહુવિધ શરતો હોય છે અને તે દરેક સાથે અમુક વિધાનો (statements) જોડાયેલા હોય છે. આકૃતિ 5.7 આવી પરિસ્થિતિ દર્શાવે છે. આ એક મલ્ટિ-ચોઈસ માળખું છે, કારણ કે કઈ શરત સાચી છે તેના આધારે આપણે એક કરતાં વધારે વિકલ્પોમાંથી એક વિકલ્પ પસંદ કરવો પડે છે. આકૃતિ 5.7માં, સૌપ્રથમ શરત-1 (Condition1) નું પરીક્ષણ થાય છે. જો તે સાચી હોય, તો વિધાન-1 (statement1) ની પ્રક્રિયા થાય છે અને નિયંત્રણ માળખાના અંત તરફ આગળ વધે છે. જો તે ખોટી હોય, તો પછી શરત-2 (Condition2) નું મૂલ્યાંકન થાય છે. જો શરત-2 (Condition2) સાચી હોય તો વિધાન-2 (statement2) ની પ્રક્રિયા થાય છે અને તે અંત તરફ આગળ વધે છે, અન્યથા તે શરત-3 (Condition3) તરફ જાય છે. જ્યાં સુધી બધી શરતોનું

પરીક્ષણ ન થાય ત્યાં સુધી આ પ્રક્રિયા ચાલુ રહે છે. જો બધી શરતો ખોટી હોય તો પછી ડિફોલ્ટ-વિધાન (Default-statement) ની પ્રક્રિયા થાય છે અને માળખું પૂર્ણ થાય છે. ટૂંકમાં, કઈ શરત સાચી છે તેના આધારે ફક્ત એક જ વિધાન અમલમાં મૂકવામાં આવશે અથવા ડિફોલ્ટ-વિધાનનો અમલ થશે.



આકૃતિ 5.7 : નિર્ણય દ્વારા મલ્ટિ-ચોઈસનો ઉપયોગ

કનેક્ટર (Connector) : સિંગલ કેપિટલ અક્ષર ધરાવતું નાનું વર્તુળ કનેક્ટર તરીકે વપરાય છે. જ્યારે કોઈ સમસ્યા પ્રમાણમાં મોટી અને જટિલ હોય છે, ત્યારે ફ્લોચાર્ટ પણ જટિલ બની જાય છે અથવા એક જ પૃષ્ઠમાં સમાઈ શકતો નથી. કનેક્ટર્સનો ઉપયોગ ફ્લોચાર્ટના બે ભાગોને એક જ પૃષ્ઠમાં અથવા અન્ય પૃષ્ઠમાં જોડવા માટે થાય છે. કનેક્ટરનો ઉપયોગ હંમેશાં જોડીમાં થાય છે: ધારો કે, "A" અક્ષર ધરાવતું એક વર્તુળ અંદર તરફ આવતા એરો સાથે (કોઈ



આકૃતિ 5.8 : કનેક્ટર્સ

અન્ય પ્રતીકમાંથી બહાર આવતું) બતાવે છે કે ફ્લોચાર્ટ તે જ પૃષ્ઠના અન્ય ભાગમાં અથવા અન્ય પૃષ્ઠમાં ચાલુ રહે છે. આ જ "A" અક્ષર ધરાવતું બીજું વર્તુળ બહાર તરફ જતા એરો સાથે તે જ પૃષ્ઠમાં અથવા અન્ય પૃષ્ઠમાં ફ્લોચાર્ટના બીજા ભાગ સાથે જોડાય છે. આકૃતિ 5.8માં "A" અક્ષર સાથેના કનેક્ટર્સની જોડી દર્શાવેલ છે.

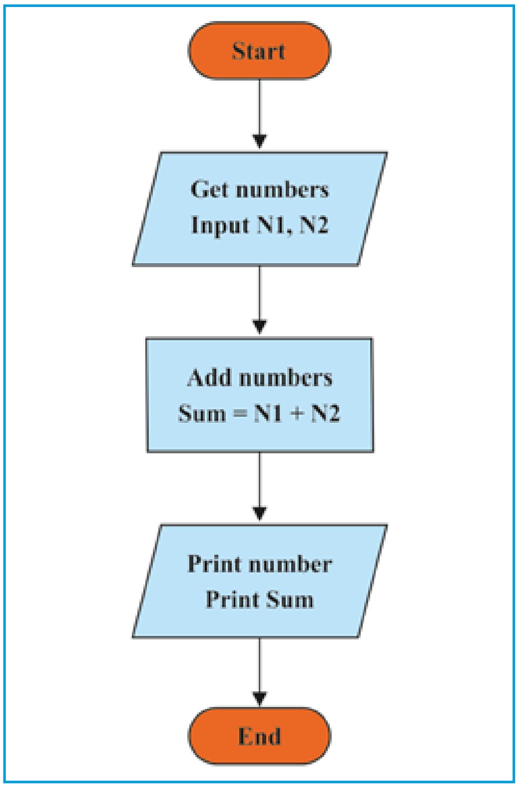
ક્રમિક અને નિર્ણય લેવાના ઉદાહરણો (Sequential and Decision-making Examples)

ચાલો હવે અગાઉના વિભાગમાં શીખેલા પ્રતીકોનો ઉપયોગ કરીને સરળ ક્રમિક પગલાં અને નિર્ણય લેવા સહિતની વિવિધ સમસ્યાઓ માટે ફ્લોચાર્ટ વિકસાવીએ.

પ્રશ્ન 1 : આપેલ બે સંખ્યાઓનો સરવાળો કરવા માટે ફ્લોચાર્ટ દોરો.

ઉકેલ :

સમસ્યા ખૂબ જ સરળ છે અને તેમાં બે સંખ્યાઓ મેળવવાની, તેમનો સરવાળો કરવાની અને અંતે તેમનો સરવાળો દર્શાવવાની જરૂર છે. આકૃતિ 5.9 ફ્લોચાર્ટ દર્શાવે છે.



આકૃતિ 5.9 : બે સંખ્યાનાં સરવાળાનો ફ્લોચાર્ટ

આકૃતિ 5.9માં બતાવ્યા પ્રમાણે, ફ્લોચાર્ટ Start પ્રતીકથી શરૂ થાય છે. પછી તે N1 અને N2 ચલોનો ઉપયોગ કરીને બે સંખ્યાઓ મેળવે છે. ત્રીજું પગલું પ્રક્રિયા છે, જેને લંબચોરસ દ્વારા દર્શાવવામાં આવ્યું છે, જે N1 અને N2 નો સરવાળો કરીને તેને ચલ Sum માં સંગ્રહિત કરે છે. ચોથું પગલું Sum માં સંગ્રહિત સરવાળાને પ્રિન્ટ કરે છે. છેલ્લું પગલું End ફ્લોચાર્ટને સમાપ્ત કરવા માટે છે.

આકૃતિ 5.9માં નાના ફેરફારો કરીને આપણે બાદબાકી માટેનો ફ્લોચાર્ટ પણ સરળતાથી દોરી શકીએ છીએ. પ્રથમ ફેરફાર, ત્રીજા પગલાં (પ્રક્રિયા પગલું) ને બદલીને

Diff = N1 – N2

કરવું. અને પછી ચોથા પગલાંને બદલીને નીચે પ્રમાણે કરો.

output the difference

Print Diff

પ્રશ્ન 2 : સાદું વ્યાજ ગણવા નીચેના સૂત્રનો ઉપયોગ કરી ફ્લોચાર્ટ દોરો.

$I = (P \times R \times N)/100$

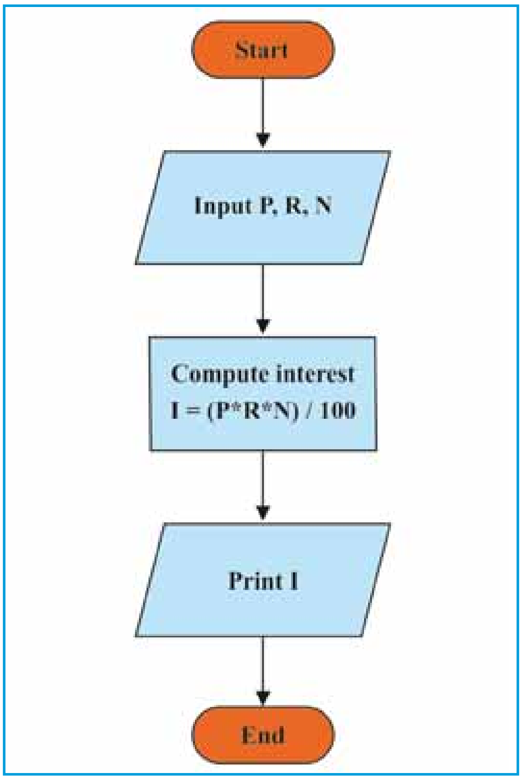
જ્યાં P = મુદ્દલની રકમ, R = વ્યાજનો દર અને N = સમયગાળો દર્શાવે છે.

ઉકેલ :

ચાલો એક ઉદાહરણ લઈએ જ્યાં

P = 100000, R = 7% અને N = 1 વર્ષ.

તો વ્યાજ (Interest) I = (100000 * 7 * 1) / 100 = 7000.



આકૃતિ 5.10 : સાદું વ્યાજ ગણવાનો ફ્લોચાર્ટ

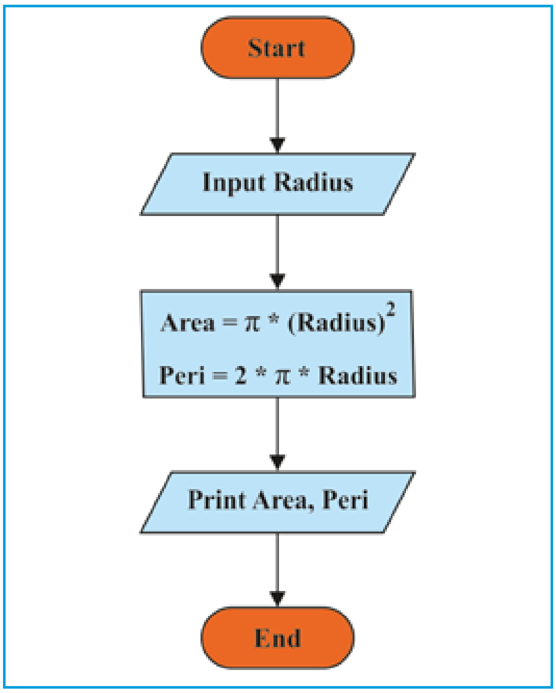
આકૃતિ 5.10 ફ્લોચાર્ટ દર્શાવે છે. ફ્લોચાર્ટમાં જોઈ શકાય છે કે ત્રણ ચલો P, R અને N નો ઉપયોગ ઇનપુટ મેળવવા માટે થાય

છે, અને પછી પ્રોસેસ બોક્સ ઉપર દર્શાવેલા સૂત્રનો ઉપયોગ કરીને વ્યાજ I ની ગણતરી કરે છે, અને અંતે ગણતરી કરેલ વ્યાજ પ્રિન્ટ કરવામાં આવે છે.

પ્રશ્ન 3 : વર્તુળના ક્ષેત્રફળ અને પરિઘ શોધવાનો ફ્લોચાર્ટ દોરો.

ઉકેલ :

વર્તુળનું ક્ષેત્રફળ અને પરિઘ ગણવા માટે, આપણે ત્રિજ્યાને ઇનપુટ તરીકે સ્વીકારવી પડે. આપણે ક્ષેત્રફળ $A = \pi R^2$ અને પરિઘ $P = 2\pi R$ નો ઉપયોગ કરીને ગણી શકીએ. ફ્લોચાર્ટ આકૃતિ 5.11માં દર્શાવેલ છે. ધ્યાન આપો કે આપણે એક જ પ્રોસેસિંગ બોક્ષમાં બે મૂલ્યોની ગણતરી કરી રહ્યા છીએ. જો તેઓ ક્રમમાં હોય તો આપણે એક જ પ્રોસેસ બોક્સમાં ગમે તેટલા મૂલ્યોની ગણતરી કરી શકીએ છીએ.

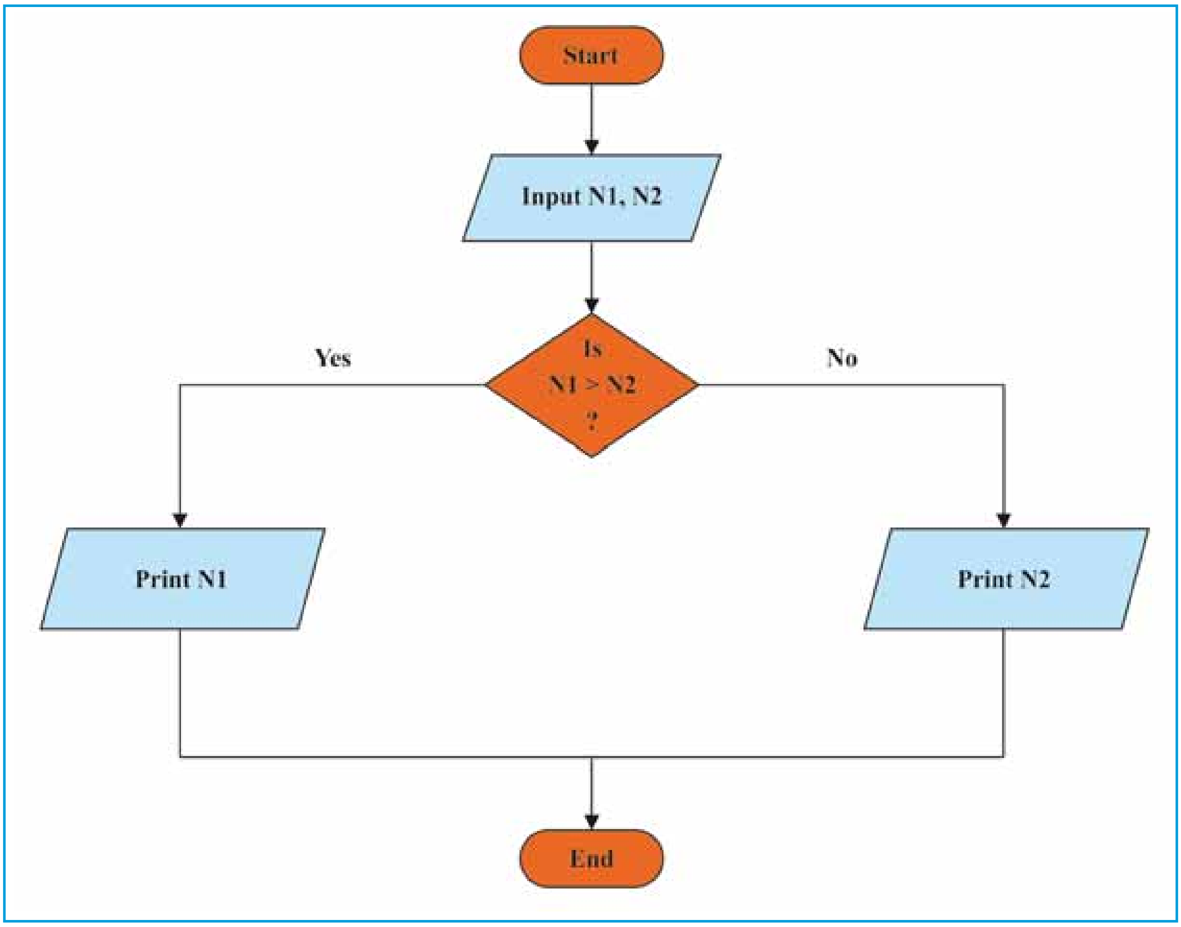


આકૃતિ 5.11 : વર્તુળના ક્ષેત્રફળ અને પરિઘ શોધવાનો ફ્લોચાર્ટ

પ્રશ્ન 4 : બે સંખ્યામાંથી મોટી સંખ્યા શોધવાનો ફ્લોચાર્ટ દોરો.

ઉકેલ :

આ ફ્લોચાર્ટમાં બે સંખ્યાઓની સરખામણી કરવા માટે ડાયમંડ પ્રતીકનો ઉપયોગ થાય છે, જેમાં ">" (કરતાં મોટું) ચિહ્નનો ઉપયોગ કરવામાં આવે છે અને પછી તેમાંથી જે સંખ્યા મોટી હોય તેને પ્રિન્ટ કરવામાં આવે છે. આ ફ્લોચાર્ટ આકૃતિ 5.12માં દર્શાવેલ છે.



આકૃતિ 5.12 : બે સંખ્યામાંથી મોટી સંખ્યા શોધવાનો ફ્લોચાર્ટ

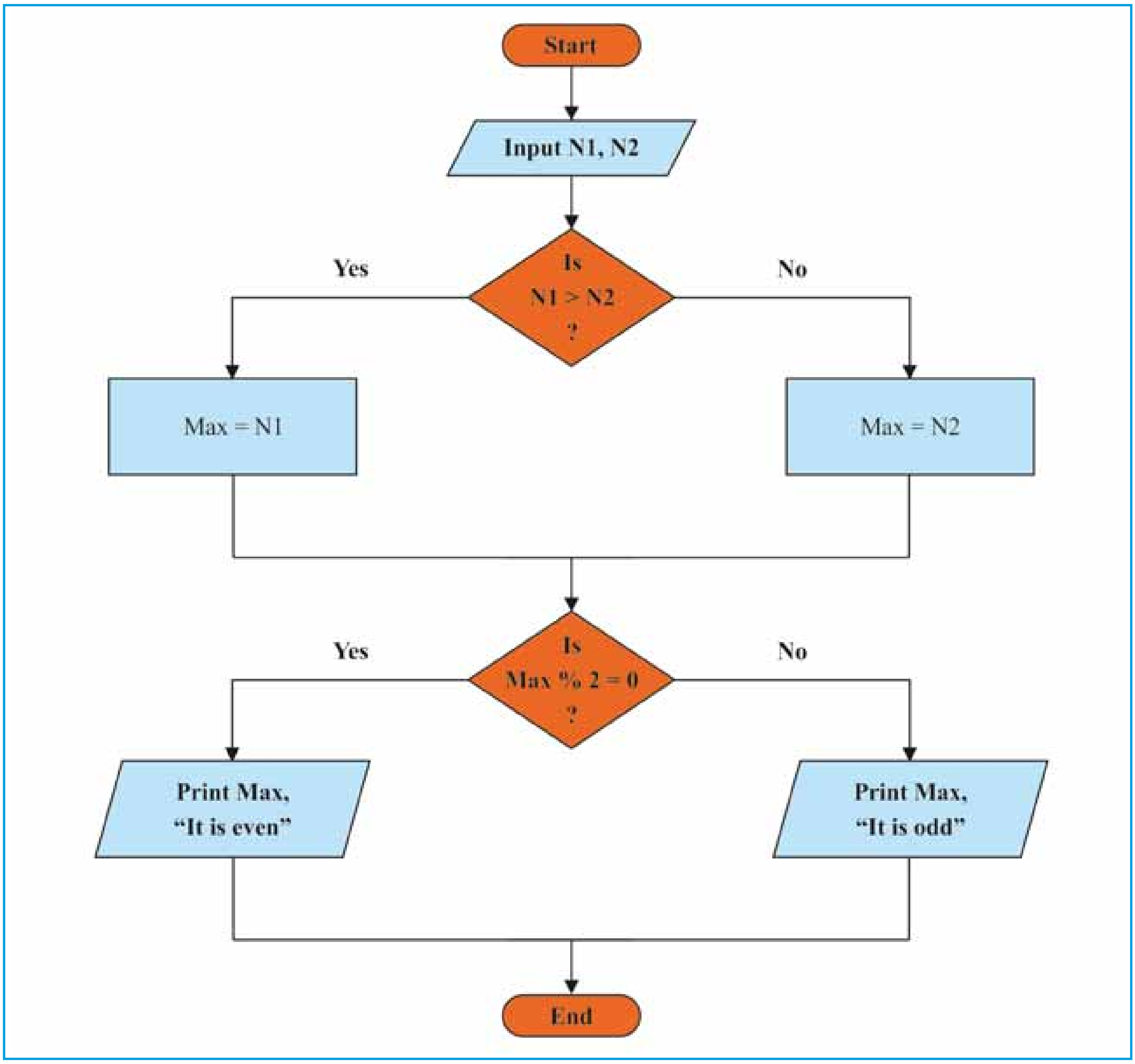
ઉપરના ફ્લોચાર્ટમાં બતાવ્યા પ્રમાણે, N1 અને N2 એમ બે સંખ્યાઓ મેળવ્યા પછી, નિર્ણય બોક્સમાં N1 > N2 શરતનો ઉપયોગ કરીને તેમની સરખામણી કરવામાં આવે છે. નિર્ણય બોક્સ બે માર્ગો બનાવે છે : એક જો શરત સાચી હોય (N1 પ્રિન્ટ કરવું) અને બીજો જો શરત ખોટી હોય (N2 પ્રિન્ટ કરવું). તેમાંથી એક પ્રિન્ટ કર્યા પછી ફ્લોચાર્ટ સમાપ્ત થાય છે.

જો આપણે ફ્લોચાર્ટ આકૃતિ 5.12ના નિર્ણય બોક્સમાં શરતને ફક્ત N1 < N2 માં બદલીએ, તો તે N1 અને N2 માંથી નાની સંખ્યાને પ્રિન્ટ કરશે.

પ્રશ્ન 5 : બે સંખ્યાઓમાંથી મોટી સંખ્યા શોધવા અને તે મોટી સંખ્યા એકી છે કે બેકી તે તપાસવા માટેનો ફ્લોચાર્ટ દોરો.

ઉકેલ :

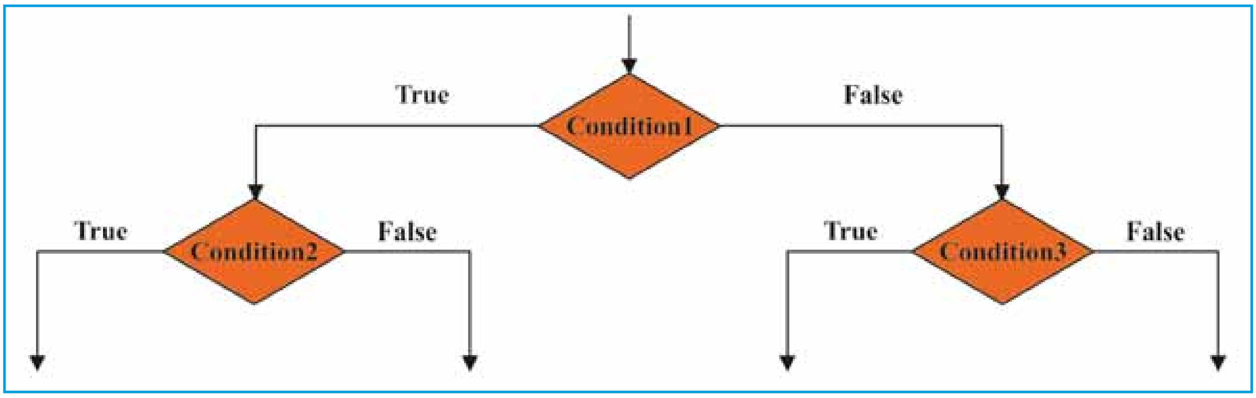
આકૃતિ 5.13માં ફ્લોચાર્ટ દર્શાવેલ છે. જેમ ફ્લોચાર્ટમાં જોઈ શકાય છે, તેમ સૌ પ્રથમ મોટી સંખ્યા શોધવામાં આવે છે (આકૃતિ 5.12 તપાસો) અને તેને Max ચલમાં સંગ્રહિત કરવામાં આવે છે. પછી, નિર્ણય બોક્સનો ઉપયોગ કરીને Max એકી છે કે બેકી તે 2 વડે મોડ્યુલો (% - Modulo) નો ઉપયોગ કરીને ચકાસવામાં આવે છે. (મોડ્યુલો એટલે કે % ઓપરેટર ભાગાકાર પછીની શેષ આપે છે). જો Max%2=0, તો તે બેકી છે, અન્યથા (=1) તે એકી છે. અંતે, આપણે Max ને "Max is Even" અથવા "Max is Odd" લેબલ સાથે પ્રિન્ટ કરીએ છીએ.



આકૃતિ 5.13 : બેમાંથી મોટી સંખ્યા એકી કે બેકી છે તે ચકાસવાનો ફ્લોચાર્ટ

નેસ્ટેડ શરતો અને પુનરાવર્તનના ઉદાહરણો (Nested Conditions and Looping with Examples)

નેસ્ટેડ શરત એટલે શરતની અંદર શરત. ક્યારેક એવું જરૂરી હોય છે કે એક શરત લાગુ કર્યા પછી, આપણે પ્રથમ શરતના પરિણામ (તે સાચું હોય કે ખોટું) ના આધારે બીજી શરત લાગુ કરવાની જરૂર પડે છે. આકૃતિ 5.14માં દર્શાવેલ પરિસ્થિતિને ધ્યાનમાં લો. સૌપ્રથમ, આપણે શરત-1 (Condition1) નું મૂલ્યાંકન કરીએ છીએ. જો તે સાચું હોય, તો આપણે શરત-2 (Condition2) લાગુ કરીએ છીએ અને Condition2 સાચી છે કે ખોટી તેના આધારે આગળ વધીએ છીએ. એ જ રીતે, જો શરત-1 (Condition1) ખોટી હોય, તો આપણે શરત-3 (Condition3) લાગુ કરીશું અને તેના પરિણામ (તે સાચું હોય કે ખોટું) ના આધારે આગળ વધીશું.

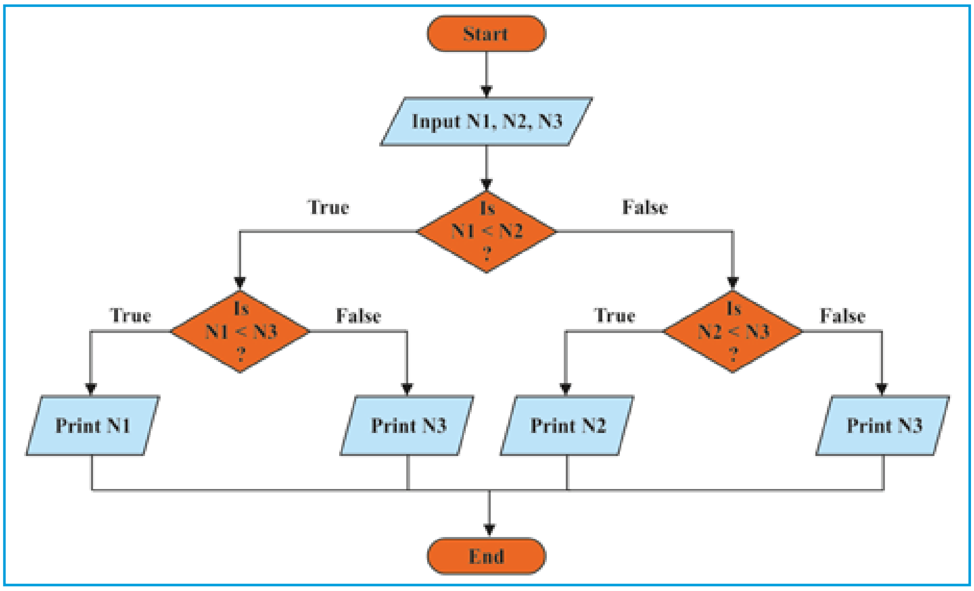


આકૃતિ 5.14 : શરતની અંદર શરત (નેસ્ટેડ શરત)

પ્રશ્ન 6 : ત્રણ સંખ્યાઓમાંથી નાની સંખ્યા શોધવાનો ફ્લોચાર્ટ દોરો.

ઉકેલ :

આકૃતિ 5.15 ત્રણ સંખ્યાઓમાંથી નાની સંખ્યા શોધવા માટેનો ફ્લોચાર્ટ દર્શાવે છે. ફ્લોચાર્ટ સૌપ્રથમ N1, N2 અને N3 ચલનો ઉપયોગ કરીને ત્રણ સંખ્યાઓ સ્વીકારે છે. પછી, નિર્ણય બોક્સ N1 અને N2 ની સરખામણી કરે છે (N1 < N2). જો N1, N2 કરતાં નાનો હોય તો અંદરનો નિર્ણય બોક્સ (ડાબી બાજુ) N1 < N3 નો ઉપયોગ કરીને N1 ની N3 સાથે સરખામણી કરે છે, જેથી N1 નાનો છે કે N3 નાનો છે તે શોધી શકાય. જો N2, N1 કરતાં નાનો હોય તો બીજો અંદરનો નિર્ણય બોક્સ (જમણી બાજુ) N2 < N3 ની ચકાસણી કરે છે, જેથી N2 નાનો છે કે N3 નાનો છે તે શોધી શકાય. અંતે, ત્રણેય સંખ્યાઓમાંથી નાની સંખ્યાને પ્રિન્ટ કરવામાં આવે છે.



આકૃતિ 5.15 : ત્રણ સંખ્યાઓમાંથી નાની સંખ્યા શોધવાનો ફ્લોચાર્ટ

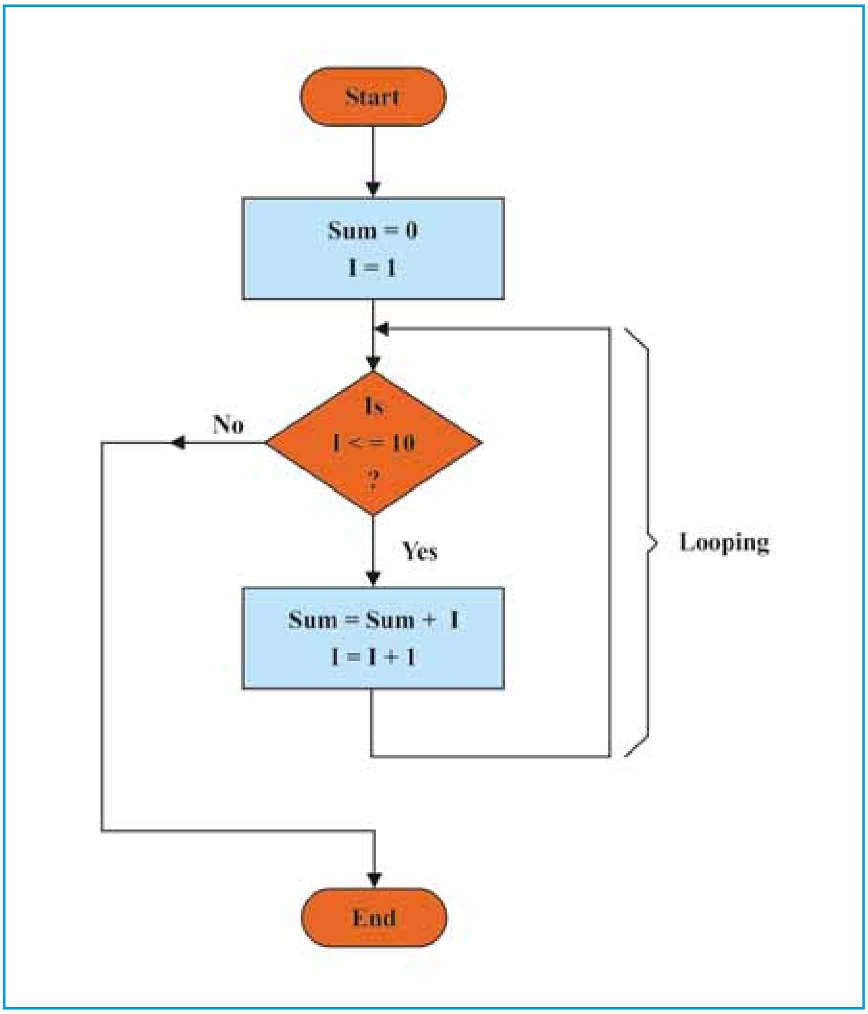
ઉદાહરણ તરીકે, ધારોકે N1 = 5, N2 = 2 અને N3 = 7 છે. શરત N1 < N2 (5 < 2) ખોટી છે. કંટ્રોલ જમણી બાજુ અંદરની શરત તરફ જાય છે જે N2 < N3 (2 < 7) ની ચકાસણી કરે છે, જે સાચી છે. અને અંતે, ત્રણેયમાંથી ન્યૂનતમ સંખ્યા N2 = 2 પ્રિન્ટ થાય છે. તમે N1, N2 અને N3 ના વિવિધ સંયોજનો સાથે આનું પરીક્ષણ કરી શકો છો.

એ પણ શક્ય છે કે કોઈ કાર્યને અમુક શરતને આધારે અમુક પગલાના સમૂહનું પુનરાવર્તન (લૂપિંગ) કરવાની જરૂર હોય. અગાઉ આપણે 1 થી 10 સંખ્યાઓનો સરવાળો કરવાનું ઉદાહરણ લીધું હતું, જેમાં ચલ I નું મૂલ્ય 10 કરતાં ઓછું અથવા તેના બરાબર હોય ત્યાં સુધી દર વખતે ચલ Sum માં આગળનું મૂલ્ય ઉમેરવા અને ચલ I માં 1 નો વધારો કરવાના પગલાનું પુનરાવર્તન કરવું પડતું હતું. ચાલો નીચેના ઉદાહરણ દ્વારા તેને સમજીએ.

પ્રશ્ન 7 : 1 to 10 નો સરવાળો કરવાનો ફ્લોચાર્ટ દોરો.

ઉકેલ :

આકૃતિ 5.16માં ફ્લોચાર્ટ દર્શાવેલ છે. સૌપ્રથમ, પ્રોસેસ બોક્સ Sum = 0 અને I = 1 નું પ્રારંભિક મૂલ્ય સેટ કરે છે. પછી નિર્ણય લેવાની શરત I ≤ 10 લાગુ કરવામાં આવે છે, અને જો તે સાચી હોય, તો Sum = Sum + I અને I = I + 1 ધરાવતું પ્રોસેસ બોક્સનો અમલ કરવામાં આવે છે. ત્યારબાદ એરો ફરીથી નિર્ણય બોક્સ તરફ નિર્દેશ કરે છે અને I ≤ 10 શરત ફરીથી નવી કિંમત સાથે ચકાસવામાં આવે છે. જ્યાં સુધી I ≤ 10 હોય ત્યાં સુધી આ પુનરાવર્તિત થાય છે. જ્યારે I = 11 થાય છે ત્યારે શરત ખોટી થાય છે અને કંટ્રોલ લૂપ પછીના આગળના પગલા પર જાય છે અને સમાપ્ત થાય છે.



આકૃતિ 5.16 : 1 to 10 નો સરવાળો કરવાનો ફ્લોચાર્ટ

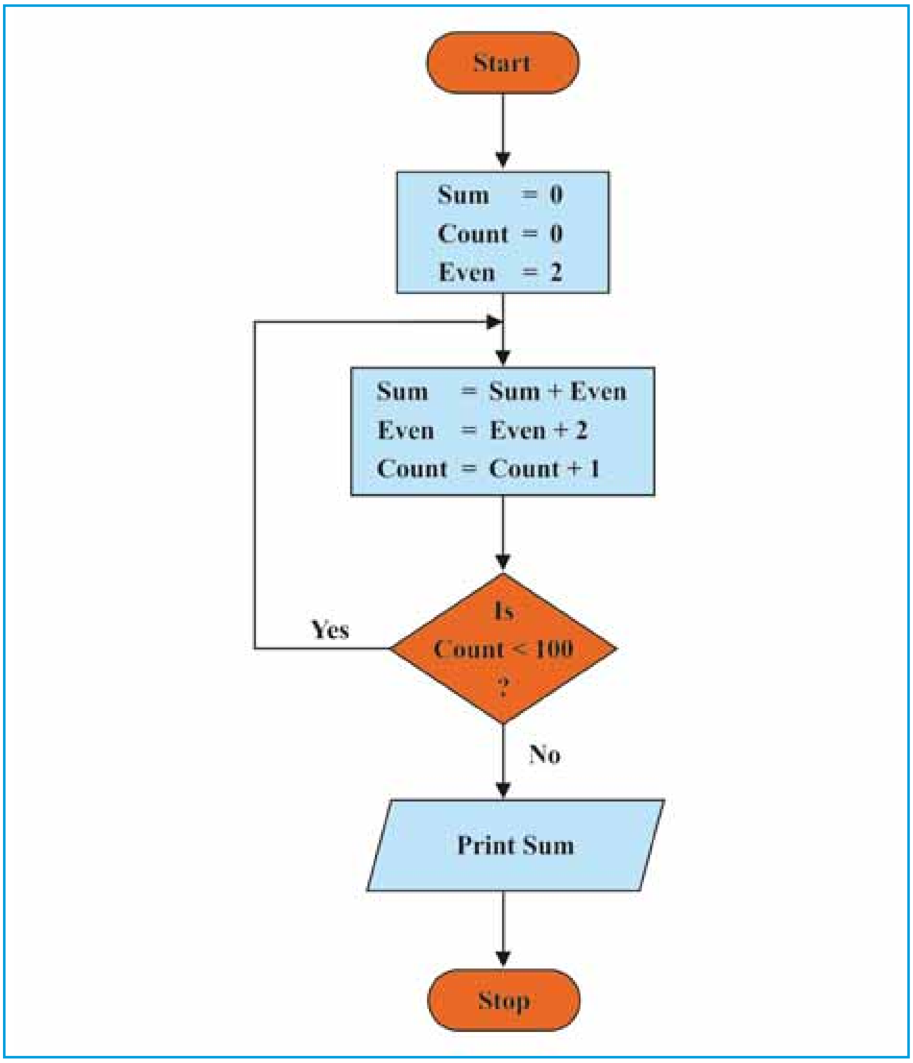
પ્રશ્ન 8 : પ્રથમ 100 બેકી સંખ્યાઓનો સરવાળો કરવાનો ફ્લોચાર્ટ દોરો.

ઉકેલ :

પ્રથમ 100 બેકી સંખ્યાઓ 1 થી 200 ની રેન્જમાં આવરી લેવાય છે અને તેમનો સરવાળો કરવાનો અર્થ છે:

$$2 + 4 + 6 + \ldots + 200$$

આકૃતિ 5.17માં ફ્લોચાર્ટ દર્શાવેલ છે. આપણને ત્રણ ચલની જરૂર છે: બધી બેકી સંખ્યાઓનો સરવાળો સંગ્રહવા માટે Sum, કાર્ય ક્યારે સમાપ્ત કરવું તે નક્કી કરવા માટે Count અને આગામી બેકી સંખ્યા સંગ્રહવા માટે Even. તેઓને Sum=0, Count=0 અને Even=2 (પ્રથમ બેકી સંખ્યા) થી પ્રારંભ કરવામાં આવે છે. આપણે Even માં રહેલી બેકી સંખ્યાને ચલ Sum માં ઉમેરીશું, અને પછી Even માં 2 નો ઉમેરો કરીશું અને Count માં 1 નો વધારો કરીશું. આ પ્રક્રિયા કર્યા પછી, ફ્લોચાર્ટ તપાસે છે કે શું Count < 100 છે? જો હા, તો આપણે પ્રક્રિયાનું પુનરાવર્તન કરીશું, અન્યથા કાર્ય સમાપ્ત થાય છે.



આકૃતિ 5.17 : પ્રથમ 100 બેકી સંખ્યાઓનો સરવાળો કરવાનો ફ્લોચાર્ટ

અલ્ગોરિધમનો ટૂંકો પરિચય (Overview of Algorithm)

અલ્ગોરિધમ એ આપેલ સમસ્યાનો તાર્કિક ક્રમમાં પગલાં સ્વરૂપે ઉકેલ છે. તે બરાબર એવું જ છે જેમ કોઈ પણ વાનગીની રેસીપી, જે ચોક્કસ ક્રમમાં શું કરવું તે સમજાવે છે. અલ્ગોરિધમ વર્ણનાત્મક શબ્દસમૂહો અથવા સ્યુડો કોડ્સ (pseudo codes)/સૂચનાઓનો ઉપયોગ કરીને લખી શકાય છે. સ્યુડો કોડ્સ ક્રિયાઓ અથવા પગલાં

દર્શાવવા માટે શબ્દો અને પ્રતીકોનો ઉપયોગ કરે છે. તે ખૂબ જ સઘન અને સમજવામાં સરળ હોય છે. અલ્ગોરિધમ સમસ્યાનું નાના પગલાંઓમાં વિઘટન કરે છે, જે સહેલાઈથી સમજી શકાય તેવા હોય છે. અલ્ગોરિધમ કમ્પ્યૂટર પ્રોગ્રામ જેવું જ લાગે છે, સિવાય કે તે કોઈ ચોક્કસ પ્રોગ્રામિંગ ભાષાના ખાસ વાક્યરચનાને બદલે સ્યુડો કોડ્સ જેવી સામાન્ય વાક્યરચનાનો ઉપયોગ કરે છે. અલ્ગોરિધમને કમ્પ્યૂટર પ્રોગ્રામમાં રૂપાંતરિત કરવું ખૂબ જ સરળ છે, કારણ કે સંપૂર્ણ ઉકેલ પહેલેથી જ તાર્કિક ક્રમમાં પગલાંનો ઉપયોગ કરીને વિકસાવવામાં આવેલો હોય છે. પ્રોગ્રામ વિકસાવતા પહેલાં અલ્ગોરિધમ લખવાથી પ્રોગ્રામની સ્પષ્ટતા અને ચોકસાઈ સુનિશ્ચિત થાય છે.

લંબચોરસનું ક્ષેત્રફળ શોધવા માટેના અલ્ગોરિધમનું ઉદાહરણ નીચે મુજબ છે.

1. START
2. Input Length, Breadth
3. Compute Area = Length * Breadth
4. Print Area
5. STOP

નોંધી લો કે અલ્ગોરિધમમાં દરેક પગલાને 1 થી શરૂ કરીને ક્રમ આપવામાં આવે છે. અલ્ગોરિધમની શરૂઆત 'START' થી થાય છે. સ્યુડો કોડ્સ 'Input', 'Compute', 'Print' અનુક્રમે ઇનપુટ, ગણતરી અને આઉટપુટ ક્રિયાઓ દર્શાવે છે. અલ્ગોરિધમનું છેલ્લું પગલું હંમેશા 'STOP' હોય છે, જે અલ્ગોરિધમના અંતને સૂચવે છે. ઉપરોક્ત અલ્ગોરિધમ સૌ પ્રથમ લંબાઈ અને પહોળાઈ મેળવે છે, ત્યારબાદ લંબાઈ અને પહોળાઈનો ગુણાકાર કરીને ક્ષેત્રફળની ગણતરી કરે છે અને અંતે ક્ષેત્રફળને આઉટપુટ તરીકે પ્રિન્ટ કરે છે.

અલ્ગોરીધમના ઉદાહરણો (Examples of Algorithm)

ચાલો આપણે ક્રમિક, નિર્ણય લેવો અને પુનરાવર્તન સહિતના વિવિધ અલ્ગોરિધમ લખીએ, જેથી આપણે તેને ઊંડાણપૂર્વક સમજી શકીએ.

પ્રશ્ન 9 : આપેલ સંખ્યાનો વર્ગ શોધવાનું અલ્ગોરીધમ લખો.

ઉકેલ : અલગોરિધમ નીચે પ્રમાણે છે.

1. START
2. Input N
3. Compute Square = N * N
4. Print Square
5. STOP

અલગોરિધમ ખૂબ જ સરળ છે અને 1 થી 5 સુધીના પગલાંને ક્રમશઃ અનુસરે છે. તે N ની કિંમત સ્વીકારે છે, વર્ગની ગણતરી કરે છે અને પછી વર્ગને પ્રિન્ટ કરે છે.

ચાલો આપણે એવા ઉદાહરણો લઈએ જેમાં નિર્ણય અને પુનરાવર્તન/લૂપિંગનો સમાવેશ થતો હોય.

પ્રશ્ન 10 : બે સંખ્યામાંથી મોટી સંખ્યા શોધવાનું અલગોરિધમ લખો.

ઉકેલ : અલગોરિધમ નીચે પ્રમાણે છે.

1. START
2. Input N1, N2
3. If N1 > N2, goto 6
4. Print N2
5. goto 7
6. Print N1
7. STOP

પ્રથમ અલ્ગોરિધમ, N1 અને N2 ચલનો ઉપયોગ કરીને બે સંખ્યાઓ વાંચે છે. પગલું-3 થી પગલું-6 નિર્ણય લેવાની પ્રક્રિયાનો અમલ કરે છે, જ્યાં પગલું 3 N1 અને N2 ની સરખામણી કરે છે (N1 > N2) અને, જો તે સાચું હોય તો તે N1 ને પ્રિન્ટ કરવા અને ત્યારબાદ 'STOP' કરવા માટે પગલું-6 પર જાય છે. જો તે ખોટું હોય, તો તે N2 ને પ્રિન્ટ કરવા માટે પગલું-4 ને અનુસરે છે અને પગલું-5 નિયંત્રણને 'STOP' કરવા માટે પગલું-7 પર મોકલે છે. જો આપણે પગલું-5 માં 'goto 7' નો ઉપયોગ ન કરીએ, તો N2 પ્રિન્ટ કર્યા પછી તે N1 પણ પ્રિન્ટ કરશે જે ખોટું છે. તેથી, આને ટાળવા માટે કાળજી લેવામાં આવી છે. તેના માટેનો ફ્લોચાર્ટ આકૃતિ 5.12 માં આપેલ છે. અલ્ગોરિધમ અને ફ્લોચાર્ટના પગલાંની તુલના કરો જેથી તે સ્પષ્ટ થશે કે આપણે અલ્ગોરિધમને ફ્લોચાર્ટમાં અને ફ્લોચાર્ટને અલ્ગોરિધમમાં કેવી રીતે રૂપાંતરિત કરી શકીએ છીએ.

પ્રશ્ન 11 : પ્રથમ 100 બેકી સંખ્યાઓનો સરવાળો કરવાનું અલગોરિધમ લખો.

ઉકેલ : અલગોરિધમ નીચે પ્રમાણે છે.

1. START
2. Initialize Sum = 0
3. Initialize Count = 0
4. Initialize Even = 2
5. Compute Sum = Sum + Even
6. Compute Even = Even + 2
7. Increment Count by 1
8. If Count < 100, goto 5
9. Print Sum
10. STOP

પગલાં 2, 3 અને 4 અનુક્રમે Sum, Count અને Even ચલને પ્રારંભિક મૂલ્યો આપે છે. પગલું-5 થી પગલું-8 પુનરાવર્તન (લૂપ) નો અમલ કરે છે. પગલું-5 વર્તમાન Even મૂલ્યને Sum માં ઉમેરે છે. પગલું-6 આગામી બેકી સંખ્યા મેળવવા માટે Even માં 2 ઉમેરે છે. પગલું-7 Count માં 1 નો વધારો કરે છે, કારણ કે એક બેકી સંખ્યા પહેલાથી જ Sum માં ઉમેરવામાં આવી છે. પગલું-8 તપાસે છે કે Count 100 કરતાં ઓછો છે કે કેમ. જો તે સાચું હોય, તો તે પરત પગલું-5 પર જાય છે અને લૂપનું પુનરાવર્તન ફરી થાય છે. જ્યારે Count = 100 થાય છે, ત્યારે શરત ખોટી પડે છે અને તે Sum ને પ્રિન્ટ કરવા માટે પગલું-9 ને અનુસરે છે. આ રીતે, પગલું-5 થી પગલું-8, 100 વખત પુનરાવર્તિત થાય છે અને દરેક વખતે આગામી બેકી સંખ્યા (2, 4, 6, ..., 200) Sum માં ઉમેરવામાં આવે છે અને Count 0 થી 99 સુધી જાય છે અને જ્યારે તે 100 થાય છે, ત્યારે લૂપ સમાપ્ત થાય છે. તેના માટેનો ફ્લોચાર્ટ આકૃતિ 5.17 માં આપેલ છે. લૂપનો અમલ કેવી રીતે થાય છે તે સમજવા માટે તેની તુલના ઉપરોક્ત અલ્ગોરિધમ સાથે કરો.

ફ્લોચાર્ટ વિરુદ્ધ અલ્ગોરિધમ (Flowchart Vs. Algorithm)

આપણે ઉદાહરણો સાથે ફ્લોચાર્ટ અને અલ્ગોરિધમનો અભ્યાસ કર્યો અને સમજ્યા કે ક્રમ, નિર્ણય અને લૂપિંગનો અમલ કેવી રીતે કરી શકાય, જે સમસ્યાના ઉકેલ માટેના સૌથી મહત્ત્વપૂર્ણ ઘટકો છે. વાસ્તવિક જીવનની કોઈપણ સમસ્યા આ ત્રણેય પ્રકારનું અલગ-અલગ સંયોજન હોય છે. જરૂરિયાત અથવા પસંદગીના આધારે, આપણે પ્રક્રિયાને - એટલે કે સમસ્યાના ઉકેલને - સંરચિત અને તાર્કિક ક્રમમાં પગલાં સ્વરૂપે રજૂ કરવા માટે ફ્લોચાર્ટ અથવા અલ્ગોરિધમ અથવા બંનેનો ઉપયોગ કરી શકીએ છીએ. જોકે ફ્લોચાર્ટ અને અલ્ગોરિધમ બંનેનો ઉપયોગ સમસ્યાના ઉકેલની પદ્ધતિઓ તરીકે થાય છે, તેમ છતાં તેમની પોતાની યોગ્યતાઓ અને ખામીઓ છે. ચાલો આપણે ફ્લોચાર્ટ અને અલ્ગોરિધમની તુલના કરીએ. તે નીચેના ટેબલમાં આપેલ છે.

ફ્લોચાર્ટ	અલ્ગોરિધમ
પ્રતીકોનો ઉપયોગ કરીને પ્રક્રિયાનું દ્રશ્ય નિરૂપણ (visual representation) પૂરું પાડે છે.	પ્રક્રિયાનું નિરૂપણ કરવા માટે સ્યુડો કોડ્સ (pseudo codes) અને સૂચનાઓનો ઉપયોગ કરે છે.
બધા પાથ દૃશ્યમાન હોય છે અને અનુસરવા માટે સરળ હોય છે.	માર્ગો છુપાયેલા હોય છે અને સમજવા માટે પગલાં વાંચવાની જરૂર પડે છે.
પ્રક્રિયાનો પ્રવાહ દર્શાવે છે, જે દ્રશ્યથી શીખનારાઓ માટે સારું છે.	પગલાંનો ઉપયોગ કરીને પ્રક્રિયાનો તર્ક પૂરો પાડે છે, જે તાર્કિક વિચારકો માટે સારું છે.
પ્રક્રિયાનું આયોજન અને સમજાવવા માટે સારું છે.	કમ્પ્યૂટર પ્રોગ્રામ લખવા માટે સારું છે, કારણ કે અલ્ગોરિધમ કમ્પ્યૂટર પ્રોગ્રામ જેવું જ હોય છે.
નવા લોકો માટે સારું છે.	કમ્પ્યૂટર પ્રોગ્રામર માટે સારું છે.

સારાંશ

આ પ્રકરણની શરૂઆત આપણે સમસ્યાના ઉકેલ અને તેના મહત્વ વિષે રજૂઆત સાથે કરી. સમસ્યાનો ઉકેલ એ કોઈપણ સમસ્યાનો ઉકેલ શોધવા માટેની એક તાર્કિક અને ક્રમબદ્ધ પ્રક્રિયા છે. સમસ્યાનું નિરાકરણ કમ્પ્યૂટર સાયન્સના કેન્દ્રમાં છે અને તે કમ્પ્યૂટર સાયન્સ તેમજ દૈનિક જીવન માટે એક મૂળભૂત કૌશલ્ય છે. આપણે સમસ્યાઓ ઉકેલવા માટેના અભિગમનો ઉપયોગ કરતી બે બહોળા પ્રમાણમાં વપરાતી પદ્ધતિઓ ફ્લોચાર્ટ અને અલ્ગોરિધમનો અભ્યાસ કર્યો છે. ફ્લોચાર્ટએ ઉકેલનો પ્રવાહ દર્શાવવા માટે પ્રતીકોનો ઉપયોગ કરીને ઉકેલનું ચિત્રાત્મક નિરૂપણ છે. વધુ સારી સમજણ માટે, આપણે ઘણી નાની અને સરળ સમસ્યાઓ માટે ફ્લોચાર્ટ્સ વિકસાવ્યા છે. ઉદાહરણો દ્વારા આપણને નિર્ણયના આધારે ફ્લોચાર્ટના વિવિધ માર્ગો અનુસરી સમજવાની તકો પણ મળી. અલ્ગોરિધમ સમસ્યાનો પગલાં સ્વરૂપે ઉકેલ પૂરો પાડે છે. વધુ સારી સમજણ અને સ્પષ્ટતા માટે, આ પ્રકરણમાં જે સમસ્યા માટે ફ્લોચાર્ટ દોર્યા, તે જ સમસ્યાઓ માટે અલ્ગોરિધમ લખવાના ઉદાહરણો આપણે લીધાં. આ પ્રકરણનો અંત આ બંને ઉકેલ પદ્ધતિઓ, એટલે કે ફ્લોચાર્ટ અને અલ્ગોરિધમ, વચ્ચેની તુલના સાથે થાય છે. આ પદ્ધતિઓ તાર્કિક વિચારસરણી અને ઉકેલો વ્યક્ત કરવામાં સ્પષ્ટતા લાવે છે. તે વિદ્યાર્થીઓને વિવિધ ઇનપુટ્સ આપી, નિર્ણયોના આધારે વિવિધ માર્ગોને અનુસરીને ઉકેલનું પરીક્ષણ કરવાની પણ મંજૂરી આપે છે. આ પ્રકરણે આવનારા પ્રકરણોમાં C પ્રોગ્રામિંગ શીખવા માટે એક મજબૂત પાયો નાખ્યો છે.

સ્વાધ્યાય

1. સમસ્યા નિરાકરણ એટલે શું? એક ઉદાહરણ આપો.
2. સમસ્યા નિરાકરણની પદ્ધતિઓ અને તેના ફાયદાઓ જણાવો.
3. ફ્લોચાર્ટ શું છે? ફ્લોચાર્ટમાં વપરાતા પ્રતીકો અને તેમનું કાર્ય દર્શાવો.
4. ફ્લોચાર્ટના પ્રોસેસ બોક્સમાં સામાન્ય રીતે કયા પ્રકારના કાર્યોનો સમાવેશ થાય છે?
5. નિર્ણય લેવા માટે કયા પ્રતીકનો ઉપયોગ થાય છે? ઉદાહરણ સાથે તેનો ઉપયોગ સમજાવો.
6. ફ્લોચાર્ટ પ્રતીકોનો ઉપયોગ કરીને તમે પુનરાવર્તન કેવી રીતે કરશો?
7. અલ્ગોરિધમ શું છે? એક સરળ ઉદાહરણ આપો.
8. અલ્ગોરિધમનો ઉપયોગ કરીને પુનરાવર્તનનું એક ઉદાહરણ આપો.
9. અલ્ગોરિધમના ગેરફાયદાઓ જણાવો.
10. ફ્લોચાર્ટ અને અલ્ગોરિધમની સરખામણી કરો.

11. સાચું કે ખોટું જણાવો.

(1) ફ્લોચાર્ટમાં બે પગલાંને જોડવા માટે એરોનો ઉપયોગ થાય છે.

(2) અલ્ગોરિધમમાં તમામ સંભવિત માર્ગો દશ્યમાન હોય છે.

(3) સમસ્યા નિરાકરણ એ સમસ્યાને ઉકેલવા માટેની તાર્કિક અને ક્રમબદ્ધ પ્રક્રિયા છે.

(4) બહુ-પસંદગી ડાયમંડ દ્વારા અમલમાં મૂકી શકાય છે.

(5) ફ્લોચાર્ટમાં હંમેશા Start અને End (શરૂઆત અને અંત) પગલાં હોય છે.

12. ખાલી જગ્યાઓ પૂરો.

(1) ____________ ઉકેલનું ચિત્રાત્મક નિરૂપણ છે.

(2) અલ્ગોરિધમમાં ____________ કોડનો ઉપયોગ થાય છે.

(3) ફ્લોચાર્ટમાં કનેક્ટર્સ (Connectors) ____________ પ્રતીકનો ઉપયોગ કરે છે.

(4) સામાન્ય રીતે અંકગણિત અને તાર્કિક કામગીરીઓ ____________ બોક્સમાં મૂકવામાં આવે છે.

(5) સમસ્યા નિરાકરણ માટેની ____________ અને ____________ પદ્ધતિઓ છે.

13. બહુવિકલ્પી પ્રશ્નો. સૌથી યોગ્ય જવાબ પસંદ કરો.

(1) ફ્લોચાર્ટની શરૂઆત અને અંત માટે નીચેનામાંથી કયું પ્રતીક વપરાય છે?

(a) લંબચોરસ (b) ડાયમંડ (c) અંડાકાર (d) વર્તુળ

(2) નીચેનામાંથી કયું સમસ્યાના ઉકેલનું દ્રશ્ય નિરૂપણ પૂરું પાડે છે?

(a) અલ્ગોરિધમ (b) ફ્લોચાર્ટ (c) પ્રોગ્રામ (d) પક્રિયા

(3) નીચેનામાંથી કયું પ્રતીક ફ્લોચાર્ટમાં પ્રક્રિયા દર્શાવવા માટે વપરાય છે?

(a) લંબચોરસ (b) ડાયમંડ (c) અંડાકાર (d) વર્તુળ

(4) નીચેનામાંથી કયું કનેક્ટર જોડીઓને એકબીજાથી અલગ પાડવા માટે વપરાય છે?

(a) એરોની દિશા (b) વર્તુળમાં લખેલ અક્ષર

(c) વર્તુળનું કદ (d) વર્તુળનો રંગ

(5) સરખામણી નીચેમાંથી શેમાં વપરાય છે?

(a) પ્રક્રિયા (b) નિર્ણય-કરતાં (c) ઈનપુટ (d) આઉટપુટ

(6) નીચેનામાંથી કયું ઉકેલની અંદરના જુદા જુદા પાથ ઓળખવા માટે શ્રેષ્ઠ છે?

(a) ફ્લોચાર્ટ (b) અલ્ગોરિધમ

(c) ફ્લોચાર્ટ અને અલગોરિધમ બંને (d) આપણે પાથ જોઈ ન શકીએ

(7) નીચેનામાંથી કયું ઉકેલના પગલાં દર્શાવવા માટે સ્યુડો કોડ્સનો ઉપયોગ કરે છે?

(a) પ્રોગ્રામ (b) ફ્લોચાર્ટ (c) અલ્ગોરિધમ (d) પ્રક્રિયા

(8) નીચેનામાંથી કયું સામાન્ય રીતે ગણતરીઓ દર્શાવવા માટે વપરાય છે?

(a) પ્રોસેસ બોક્ષ (b) નિર્ણય બોક્ષ (c) ઈનપુટ બોક્ષ (d) આઉટપુટ બોક્ષ

(9) નીચેનામાંથી કયું ફ્લોચાર્ટને એક કરતાં વધુ પૃષ્ઠોમાં વિભાજિત કરવા માટે વપરાય છે?

(a) એરો સાથે અક્ષર (b) કનેક્ટર્સ (c) એરો (d) ખાસ પ્રતીકો

(10) જયારે આપણે બે કિંમતો ગણવાની હોય ત્યારે તેને

(a) એક જ બોક્ષમાં એક પછી એક મૂકો

(b) તેઓને બે જુદા બોક્ષમાં એક પછી એક મૂકો

(c) a અને b બંને

(d) બે જુદા બોક્ષમાં એક પછી એક એમ જ હોવા જોઈએ

પ્રાયોગિક સ્વાધ્યાય

1. C = (5.0/9.0) × (F–32) સૂત્રનો ઉપયોગ કરીને ફેરનહીટને સેલ્સિયસમાં રૂપાંતરિત કરવા માટેનો ફ્લોચાર્ટ દોરો.
2. આપેલી ધન સંખ્યા એકી છે કે બેકી તે ચકાસવા માટેનો ફ્લોચાર્ટ દોરો.
3. 1 થી 100 સુધીની માત્ર એકી સંખ્યાઓનો સરવાળો કરવા માટેનો ફ્લોચાર્ટ દોરો.
4. વિદ્યાર્થીના 100 માંથી ગુણ સ્વીકારવા અને જો ગુણ ≤ 35 હોય તો "Fail", જો ગુણ > 35 અને ≤ 60 હોય તો "Second class", જો ગુણ > 60 અને ≤ 70 હોય તો "First class", અન્યથા "Distinction" પ્રિન્ટ કરવા માટેનો ફ્લોચાર્ટ દોરો.
5. એક સંખ્યાનું અનુમાન કરવા માટેનો ફ્લોચાર્ટ દોરો. તે વપરાશકર્તા પાસેથી સંખ્યા 7 ન હોય ત્યાં સુધી વારંવાર સંખ્યા સ્વીકારે છે. જયારે સંખ્યા 7 હોય, ત્યારે તે "Congraulations!" પ્રિન્ટ કરે છે અને સમાપ્ત થાય છે.
6. આપેલી 10 સંખ્યાઓમાંથી લઘુત્તમ સંખ્યા શોધવા માટેનો ફ્લોચાર્ટ દોરો.
7. આપેલી બે સંખ્યાઓના સરવાળા અને તફાવત શોધવા માટેનું અલ્ગોરિધમ લખો.
8. ત્રણ સંખ્યાઓમાંથી મહત્તમ સંખ્યા શોધવા માટેનું અલ્ગોરિધમ લખો.
9. ઈનપુટ તરીકે આપેલી N સંખ્યાઓમાંથી માત્ર બેકી સંખ્યાઓનો સરવાળો કરવા માટેનું અલ્ગોરિધમ લખો.
10. આપેલી શ્રેણીનો સરવાળો કરવા માટેનો અલ્ગોરિધમ લખો : 1 – 2 + 3 – 4 + 5 – 6 + ... + N.

C પ્રોગ્રામિંગનો પરિચય

પરિચય

અગાઉના પ્રકરણમાં, આપણે અલ્ગોરિધમ કેવી રીતે ડિઝાઈન કરવા અને ફ્લોચાર્ટ કેવી રીતે દોરવા તે શીખ્યા. કોડિંગ શરૂ કરીએ તે પહેલાં, આ વસ્તુઓ આપણને સ્ટેપ-બાય-સ્ટેપ ઉકેલનું આયોજન કરવામાં અને તેને સમજવામાં મદદ કરે છે. પરંતુ આયોજન એ માત્ર પ્રથમ ભાગ છે. આપણા ઉકેલને કમ્પ્યૂટર પર કાર્યરત કરવા માટે, આપણે આ તાર્કિક પગલાંઓને Python, Java કે C જેવી ચોક્કસ પ્રોગ્રામિંગ ભાષા (Programming Language) માં વ્યક્ત કરવા આવશ્યક છે. આ ભાષાઓમાં, C ખૂબ જ મહત્વપૂર્ણ છે. તે ઝડપી અને શક્તિશાળી હોવા ઉપરાંત ઘણી આધુનિક પ્રોગ્રામિંગ ભાષાઓનો આધાર છે. આ પ્રકરણમાં, આપણે C પ્રોગ્રામિંગ શીખવાની શરૂઆત કરીશું. આપણે સરળ C પ્રોગ્રામ કેવી રીતે લખવા, C પ્રોગ્રામના મુખ્ય ઘટકોને સમજવા અને C પ્રોગ્રામમાં વપરાતા મૂળભૂત અક્ષરો, સંખ્યાઓ અને પ્રતીકો વિશે જાણીશું. કમ્પ્યૂટર પ્રોગ્રામિંગની દુનિયામાં આ આપણું પ્રારંભિક પગલું છે.

પ્રોગ્રામિંગ ભાષાની આપણને શી જરુરીયાત છે?

પ્રાકૃતિક ભાષાઓમાં ઘણીવાર અસ્પષ્ટતા (ambiguity) જોવા મળે છે, જ્યાં એક જ વાક્યના અનેક અર્થઘટન (interpritations) થઈ શકે છે. કમ્પ્યૂટર સાથે વાતચીત કરતી વખતે આ એક મોટો પડકાર છે, કારણ કે કમ્પ્યૂટરને ચોક્કસ અને સ્પષ્ટ હોય તેવી સૂચનાઓની જરૂર હોય છે. મનુષ્યોથી વિપરીત, કમ્પ્યૂટર અસ્પષ્ટ સૂચનાઓમાંથી અર્થ તારવી શકતા નથી. પ્રોગ્રામિંગ ભાષાઓ કડક નિયમો લાગુ કરીને આ સમસ્યાનું નિરાકરણ લાવે છે. આ નિયમો સુનિશ્ચિત કરે છે કે દરેક સૂચનાનો માત્ર એક જ ઉદ્દેશિત અર્થ હોય. પ્રોગ્રામિંગ ભાષા શીખવી એ કોમ્યુનિકેશનનું એક નવું સ્વરૂપ છે, જેના દ્વારા આપણે કમ્પ્યૂટરને ચોક્કસ કાર્યો કરવા માટે અસરકારક રીતે આદેશ આપી શકીએ છીએ.

પ્રોગ્રામ અને પ્રોગ્રામિંગની સમજણ

C ભાષા શીખતા પહેલાં, ચાલો આપણે પ્રોગ્રામિંગના મુખ્ય ખ્યાલોથી શરૂઆત કરીએ. પ્રોગ્રામ એ સૂચનાઓનો એક ચોક્કસ ક્રમ છે જે કમ્પ્યૂટરને ગણતરીઓ કરવા અથવા ડેટા પ્રદર્શિત કરવા જેવા કાર્યો કરવા માટે નિર્દેશિત કરે છે. કમ્પ્યૂટર ફક્ત બાઈનરી કોડ્સ (0 અને 1) પર પ્રક્રિયા કરતા હોવાથી, આપણે C જેવી માળખાગત પ્રોગ્રામિંગ ભાષાઓનો ઉપયોગ કરીએ છીએ જે સ્પષ્ટ હોય તેવી વાક્યરચના (statement) લાગુ કરે છે. ત્યારબાદ, કમ્પાઈલર (compiler) નામનું વિશિષ્ટ સોફ્ટવેર આ માનવ-વાંચનીય કોડને મશીન ભાષા (બાઈનરી કોડ્સ) માં અનુવાદિત કરે છે, જેનો કમ્પ્યૂટર અમલ કરી શકે છે. પ્રોગ્રામિંગ ભાષાઓ જુદી જુદી શ્રેણીઓમાં અસ્તિત્વ ધરાવે છે: ઉચ્ચ-સ્તરીય ભાષાઓ (High-level languages) માનવ-વાંચનીયતા અને અમૂર્તતા (abstraction) ને પ્રાથમિકતા આપે છે. જ્યારે નિમ્ન-સ્તરીય ભાષાઓ (low-level languages) જટિલતાના ભોગે હાર્ડવેર નિયંત્રણ પ્રદાન કરે છે. C ભાષા અનન્ય રીતે આ બંને શ્રેણીઓ વચ્ચે સેતુ બાંધે છે તેમજ કાર્યક્ષમતા અને અભિવ્યક્ત શક્તિ બંને પૂરી પાડે છે.

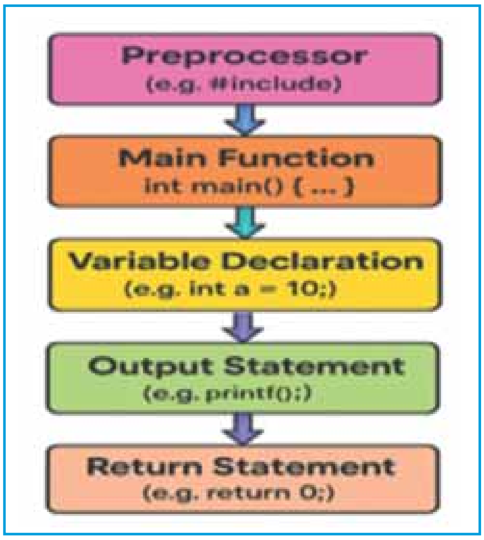
C ભાષાનો ઈતિહાસ

C ભાષાનો વિકાસ કમ્પ્યૂટર વૈજ્ઞાનિક ડેનિસ રિચી (Dennis Ritchie) એ 1972 માં બેલ લેબ્સ (Bell Labs) ખાતે કર્યો હતો. B અને BCPL જેવી અગાઉની પ્રોગ્રામિંગ ભાષાઓમાં જોવા મળતી મર્યાદાઓને દૂર કરવા માટે C પ્રોગ્રામિંગ ભાષા બનાવવામાં આવી હતી. રિચીનો ઉદ્દેશ એક એવી પ્રોગ્રામિંગ ભાષા બનાવવાનો હતો જે સરળ,

ઝડપી હોય અને કમ્પ્યૂટર હાર્ડવેર સાથે સીધો સંવાદ કરી શકે. આજે વર્ષો વીતી ગયા હોવા છતાં, C વિશ્વભરમાં વ્યાપકપણે ઉપયોગમાં લેવાય છે કારણ કે તે પ્રમાણમાં શીખવામાં સરળ છે, ઘણી જુદી જુદી કમ્પ્યૂટર સિસ્ટમ પર કાર્યક્ષમ રીતે ચાલી શકે છે અને પ્રોગ્રામરને શક્તિશાળી અને કાર્યક્ષમ કોડ લખવાની સવલત આપે છે. તેના મુખ્ય ખ્યાલોએ તેના પછી ઉદ્ભવેલી ઘણી મુખ્ય કમ્પ્યૂટર ભાષાઓને ભારે પ્રભાવિત કરી. આજે પણ, C શાળાઓમાં શીખવવામાં આવતી અને ઈન્ડસ્ટ્રીમાં સોફ્ટવેર સિસ્ટમ બનાવવા માટે વપરાતી એક આધારભૂત ભાષા બની રહી છે.

C પ્રોગ્રામનો બ્લોક ડાયાગ્રામ

આકૃતિ 6.1, C પ્રોગ્રામનો મૂળભૂત પ્રવાહ દર્શાવે છે. તે *#include* જેવી પ્રીપ્રોસેસર ડાયરેક્ટિવથી શરૂ થાય છે, ત્યારબાદ main ફંક્શન (જેમ કે *int main()*) આવે છે. પછી, ચલો (variables) ઘોષિત કરવામાં આવે છે. *printf()* નો ઉપયોગ કરીને આઉટપુટ (પરિણામ) પ્રિન્ટ કરવામાં આવે છે, અને પ્રોગ્રામ સામાન્ય રીતે *return 0;* વિધાન (statement) સાથે સમાપ્ત થાય છે, જે સફળતાપૂર્વક અમલ (execution) પૂર્ણ થયાનું સૂચવે છે. C પ્રોગ્રામ લખવાનું પ્રથમ પગલું એ છે કે તેને એડિટર (editor)નો ઉપયોગ કરીને ટાઈપ કરવો. આપણે આપણા કમ્પ્યૂટર સિસ્ટમ પર ઉપલબ્ધ કોઈપણ એડિટરનો ઉપયોગ કરી શકીએ છીએ. GCC જેવા કમ્પાઈલરનો ઉપયોગ C પ્રોગ્રામને કમ્પાઈલ (compile – મશીન કોડમાં રૂપાંતર) કરવા માટે કરી શકાય છે.



આકૃતિ 6.1 : C પ્રોગ્રામનો મૂળભૂત પ્રવાહ

પગલાંઓ : પ્રોગ્રામ બનાવી, સંગ્રહી, ખોલી અને ચલાવવાની પ્રક્રિયા – hello.c

1. એડીટર ઓપન કરો
2. તમારો કોડ ટાઈપ કરો
3. **File → Save As** ક્લિક કરો.
4. ફાઈલને નામ આપો : hello.c
5. ખાત્રી કરો કે ફાઈલ ".c" એક્સ્ટેંશન (.txt નહીં) સાથે સંગ્રહ કરી છે

નીચેનો C પ્રોગ્રામ સ્ક્રીન પર "Hello, World!" કેવી રીતે પ્રિન્ટ કરવું તે દર્શાવે છે.

```
/* C Program to print Hello World! */
#include <stdio.h>
int main()
{
        printf("Hello, World!\n");
        return 0;
}

Result:
Hello, World!
```

આ એક સાદો પ્રોગ્રામ છે, જે નવા શીખનારાઓ માટે C પ્રોગ્રામિંગ કેવી રીતે કાર્ય કરે છે તે સમજવા માટે છે. C પ્રોગ્રામ *#include <stdio.h>* લાઈનથી શરૂ થાય છે, જે કમ્પ્યૂટરને સ્ટાન્ડર્ડ ઈનપુટ/આઉટપુટ (Standard

Input/Output) લાઈબ્રેરીનો સમાવેશ કરવા જણાવે છે. આનાથી આપણે સ્ક્રીન પર આઉટપુટ દર્શાવવા માટે *printf()* જેવા ફંક્શનનો ઉપયોગ કરી શકીએ છીએ. બીજી લાઈન, *int main()*, પ્રોગ્રામનો પ્રારંભ દર્શાવે છે. આ પછી, ખૂલતો કર્લી બ્રેસ { ફંક્શનના મુખ્ય ભાગની શરૂઆત સૂચવે છે, જ્યાં પ્રોગ્રામની સૂચનાઓ લખવામાં આવશે. આ ભાગની અંદર, *printf("Hello, World!\n");* લાઈનનો ઉપયોગ સ્ક્રીન પર "Hello, World!" સંદેશ પ્રિન્ટ કરવા માટે થાય છે અને સ્ટ્રિંગમાં રહેલ \n (ન્યૂ લાઈન કેરેક્ટર) કર્સરને આગલી લાઈન પર ખસેડે છે. *return 0;* લાઈન કમ્પ્યૂટરને જણાવે છે કે પ્રોગ્રામ સફળતાપૂર્વક તેનો અમલ (execution) પૂર્ણ કરી ચૂક્યો છે. અંતે, બંધ થતો કર્લી બ્રેસ } *main()* ફંક્શનનો અંત દર્શાવે છે. આ સરળ પ્રોગ્રામ એક સામાન્ય C પ્રોગ્રામની મૂળભૂત રચના અને કાર્ય દર્શાવે છે. C પ્રોગ્રામ લખવા, સેવ કરવા, કમ્પાઈલ કરવા અને અમલ કરવાના પગલાં પ્રકરણના અંતમાં આપેલા છે.

C ભાષાનો કેરેક્ટર સેટ (Character Set in C Language)

C પ્રોગ્રામિંગમાં, કેરેક્ટર સેટ (Character Set) એ અક્ષરોના સમૂહને દર્શાવે છે જેનો ઉપયોગ પ્રોગ્રામ લખવા માટે થાય છે. જેમાં અક્ષરો (letters), અંકો (digits), ખાસ પ્રતીકો (special symbols), ખાલી જગ્યાઓ (white spaces) અને અન્ય નિયંત્રણ અક્ષરો (control characters)નો સમાવેશ થાય છે. કેરેક્ટર સેટને સમજવું આવશ્યક છે કારણ કે દરેક પ્રોગ્રામ માન્ય અક્ષરોથી બનેલો હોય છે, જે ભાષાના ચલ, ફંક્શન, ઓપરેટર અને અન્ય ઘટકોની રચના કરે છે. ટેબલ 6.1 C ભાષાનો કેરેક્ટર સેટ દર્શાવે છે.

કેરેક્ટર સેટ કેટેગરી	વર્ણન	ઉદાહરણ
1. અક્ષરો (Letters)	C ભાષામાં અક્ષરો (Alphabats)નો ઉપયોગ આઈડેન્ટિફાયર (identifiers) (જેમ કે ચલ), કીવર્ડ્સ અને અન્ય નામોની રચના કરવા માટે થાય છે. આઈડેન્ટિફાયર જાહેર કરતી વખતે અપરકેસ (મોટા અક્ષરો) અને લોઅરકેસ (નાના અક્ષરો) બંને માન્ય છે.	• અપરકેસ-મોટા અક્ષરો (A–Z) • લોઅરકેસ-નાના અક્ષરો (a–z) **ઉદાહરણ :** ચલોના નામ ઘોષિત કરવા • int Age; char name;
2. અંકો (Digits)	અંકો (Digits) નો ઉપયોગ અચલ (constants), એરે ઈન્ડેક્સ (array index), વગેરેમાં આંકડાકીય મૂલ્યો (numeric values) રજૂ કરવા માટે થાય છે.	• 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 **ઉદાહરણ :** અચલ, એરે ઈન્ડેક્સ માટે • int num = 45; • int num[5];
3. ખાસ પ્રતીકો (Special symbol / characters)	ખાસ પ્રતીકો (Special Symbols) એવા પ્રતીકો છે જેનો C ભાષામાં વિશિષ્ટ અર્થ હોય છે. તેનો ઉપયોગ ઓપરેટરો (operators), ડિલિમિટર્સ (delimiters), કોડની રચના (structuring code) અને અર્થને વ્યાખ્યાયિત કરવા માટે થાય છે.	• ગણિતીય ઓપરેટર જેવા કે +, –, *, / • Address-of ઓપરેટર (&) • તાર્કિક AND (&&), તાર્કિક OR (||) વગેરે • કોડ બ્લોક માટે કર્લી બ્રેસીસ { } • એસાઈન્મેન્ટ ઓપરેટર (=) અને • બીજા ઘણાં જેવાકે = =, (), [], #, %, !, ^

ટેબલ 6.1 : C કેરેક્ટર સેટ

વ્હાઈટ સ્પેસ (White Spaces)

C પ્રોગ્રામિંગમાં, વ્હાઈટ સ્પેસ (ખાલી જગ્યા) નો ઉપયોગ પ્રોગ્રામિંગ કોડમાં શબ્દો, પ્રતીકો અને ઘટકોને અલગ કરવા માટે થાય છે. તેઓ પ્રોગ્રામના અમલ પર અસર કરતા નથી, પરંતુ કોડને વાંચવા યોગ્ય (readable) અને વ્યવસ્થિત (organized) બનાવવા માટે મહત્વપૂર્ણ છે. સામાન્ય વ્હાઈટ સ્પેસ અક્ષરોમાં સ્પેસ (જગ્યા), ટેબ (tab) અને નવી લાઈન (newlines) નો સમાવેશ થાય છે. ટેબલ 6.2 C માં ઉપયોગમાં લેવાતા વિવિધ પ્રકારના વ્હાઈટ સ્પેસ પ્રતિકો દર્શાવે છે.

વ્હાઈટ સ્પેસ કેરેક્ટર	ઉપયોગ/વર્ણન
સ્પેસ (Space)	કીવર્ડ, આઈડેન્ટિફાયર અને ઑપરેટરને અલગ પાડવા માટે ઉપયોગમાં લેવાય છે. (e.g., 'int a = 10;')
ટેબ (Tab - \t)	કોડને વધુ માળખાકીય અને વાંચવામાં સરળ બનાવવા માટે ઉપયોગમાં લેવાય છે.
નવી લાઈન (Newline - \n)	કર્સરને આગળની લાઈન પર ખસેડે છે. સામાન્ય રીતે આઉટપુટમાં ઉપયોગમાં લેવાય છે.
કરેજ રીટર્ન (Carriage Return - \r)	કર્સરને લાઈનની શરૂઆતમાં પરત લાવે છે.
ફોર્મ ફીડ (Form Feed - \f)	પ્રિન્ટિંગ દરમિયાન પછીના પાનાં પર જવા માટે ઉપયોગમાં લેવાય છે (ભાગ્યે જ વપરાય છે).

ટેબલ 6.2 : વ્હાઈટ સ્પેસ

કીવર્ડ્સ (Keywords)

C પ્રોગ્રામિંગમાં કીવર્ડ્સ (મુખ્ય શબ્દો) એ પૂર્વનિર્ધારિત અને સંગ્રહિત શબ્દો (reserved words) છે, જે કમ્પાઈલર માટે વિશેષ અર્થ ધરાવે છે. આ શબ્દોનો ઉપયોગ નિર્ધારિત કાર્ય કરવા અને પ્રોગ્રામના પ્રવાહને નિયંત્રિત કરવા માટે થાય છે. કીવર્ડ્સનો ઉપયોગ ચલ, ફંક્શન અથવા અન્ય આઈડેન્ટિફાયરના નામ તરીકે કરી શકાતો નથી, કારણ કે તે C ભાષાનો એવો ભાગ છે, જે ખાસ અર્થ ધરાવે છે. C ભાષામાં કુલ 32 કીવર્ડ્સ છે. ટેબલ 6.3 માં સામાન્ય રીતે ઉપયોગમાં લેવાતા કેટલાક કીવર્ડ્સ દર્શાવવામાં આવ્યા છે.

કીવર્ડ (Keyword)	અર્થ/ઉપયોગ
int	પૂર્ણાંક ચલ ઘોષિત કરવા માટે. જેમકે, 1, 500, 78
float	અપૂર્ણાંક ચલ ઘોષિત કરવા માટે. જેમકે, 784.5, 6.78
char	કેરેક્ટર ચલ ઘોષિત કરવા માટે
return	ફંક્શનમાંથી મુલ્ય પાછું આપવા
if	શરતી બ્રાન્ચિંગ માટે
else	'if' સાથે વૈકલ્પિક અમલ માટે
while	લૂપ બનાવવા માટે જેનો અમલ શરત સાચી હોય ત્યાં સુધી થયા કરે
for	એવી લૂપ બનાવવા કે જેમાં પ્રારંભ (initialization), શરત (condition) અને વધારો/ઘટાડો (increment/decrement) એક જ લાઈનમાં હોય
void	ફંક્શન કોઈ મુલ્ય પરત આપતું નથી તે દર્શાવવા
break	લૂપ કે switch વિધાનને પૂરું કરવા

ટેબલ 6.3 : કીવર્ડ્સ

આઈડેન્ટિફાયર્સ (Identifiers)

C પ્રોગ્રામિંગમાં, આઈડેન્ટિફાયર્સ (Identifiers) એવા નામો છે જેનો ઉપયોગ ચલ, ફંક્શન, એરે (arrays), સ્ટ્રક્ચર (structure) અને અન્ય યુઝર-નિર્મિત ઘટક (user-defined elements) ને ઓળખવા માટે થાય છે. આઈડેન્ટિફાયર પ્રોગ્રામરને કોડમાં ચોક્કસ ડેટા અથવા પ્રક્રિયાનો સંદર્ભ લેવાની મંજૂરી આપે છે. C માં આઈડેન્ટિફાયર્સને ચોક્કસ નિયમોનું પાલન કરવું પડે છેઃ

1. આઈડેન્ટિફાયરની શરૂઆત અક્ષર (A-Z અથવા a-z) અથવા અન્ડરસ્કોર (_) થી થવી જોઈએ.
2. પ્રથમ કેરેક્ટર પછી, અંકો (0–9), અક્ષરો અથવા અન્ડરસ્કોરનો ઉપયોગ કરી શકાય છે.
3. કીવર્ડ (Keyword) નો ઉપયોગ આઈડેન્ટિફાયર તરીકે કરી શકાતો નથી.

જેમ કે આપણે અગાઉ ચર્ચા કરી કે, C એ કેસ-સેન્સિટિવ (Case-Sensitive) ભાષા છે, તેથી 'Value' અને 'value' અલગ-અલગ આઈડેન્ટિફાયર્સ ગણાય છે. ટેબલ 6.4 માન્ય અને અમાન્ય આઈડેન્ટિફાયર્સના ઉદાહરણો પૂરા પાડે છે.

આઈડેન્ટિફાયર (Identifier)	માન્ય/અમાન્ય અને કારણ
totalMarks	માન્ય – અક્ષરથી શરું થઈ બધા જ અક્ષરો ધરાવે છે
_value	માન્ય – અન્ડરસ્કોરથી શરું થાય છે
count1	માન્ય – પહેલા અક્ષર પછી અક્ષરો અને અંકો માન્ય છે
1value	અમાન્ય – અંકથી શરુ થાય છે
float	અમાન્ય - 'float' કીવર્ડ છે
user-name	અમાન્ય – હાઈફન (hyphen) માન્ય નથી
Value	માન્ય - 'value' કેસ-સેન્સિટિવ હોવાથી તેના કરતા અલગ છે

ટેબલ 6.4 : આઈડેન્ટિફાયર્સ

ચલો અને અચલો (Variables and Constants)

ચલ (Variable) અને અચલ (Constant) પ્રોગ્રામમાં ડેટા સ્ટોર કરવા માટે વપરાતા મૂળભૂત ઘટકો છે. ચલ એ સ્ટોરેજ લોકેશનને (સંગ્રહ સ્થાનને) આપેલું નામ છે, જ્યાં પ્રોગ્રામના અમલ દરમિયાન ડેટા સંગ્રહિત થાય છે. પ્રોગ્રામના અમલ દરમિયાન ચલની કિંમત બદલાઈ શકે છે. આપણે ચલનો ઉપયોગ કરતા પહેલા તેના ડેટા પ્રકાર (દા.ત., *int, float, char*) સ્પષ્ટ કરીને અને તેને એક નામ આપીને તેને ઘોષિત કરવો જોઈએ. ઉદાહરણ તરીકેઃ

```
int age = 25;
```

જે age નામનો પૂર્ણાંક (integer) ચલ જાહેર કરે છે અને તેને 25 ની કિંમત સાથે આરંભ (initialize) કરે છે. આ કિંમતનો ઉપયોગ આપણા પ્રોગ્રામમાં પછીથી કરી શકાય છે. અચલ એક નિશ્ચિત કિંમત છે, જે એકવાર વ્યાખ્યાયિત થઈ ગયા પછી, પ્રોગ્રામ દ્વારા બદલી શકાતી નથી. અચલ એ સુનિશ્ચિત કરે છે કે મહત્વપૂર્ણ કિંમતો આકસ્મિક રીતે સંશોધિત (modified) ન થાય. અચલને વ્યાખ્યાયિત કરવાની એક સામાન્ય રીત *#define* પ્રી-પ્રોસેસર ડાયરેક્ટીવ અથવા *const* કીવર્ડનો ઉપયોગ કરવાની છે. ઉદાહરણ તરીકે :

```
#define PI 3.14159
```

આ ઘોષણા (declaration) ખાતરી કરે છે કે PI ની કિંમત પ્રોગ્રામ દરમિયાન 3.14159 (અચલ) જ રહે છે. ચલ અને અચલ પ્રોગ્રામમાં ડેટાનું અસરકારક રીતે સંચાલન અને ફેરફાર કરવાની મંજૂરી આપે છે.

C પ્રોગ્રામ ઉબુન્ટુમાં ચલાવવાની પ્રક્રિયા (Executing C Program in Ubuntu (Linux))

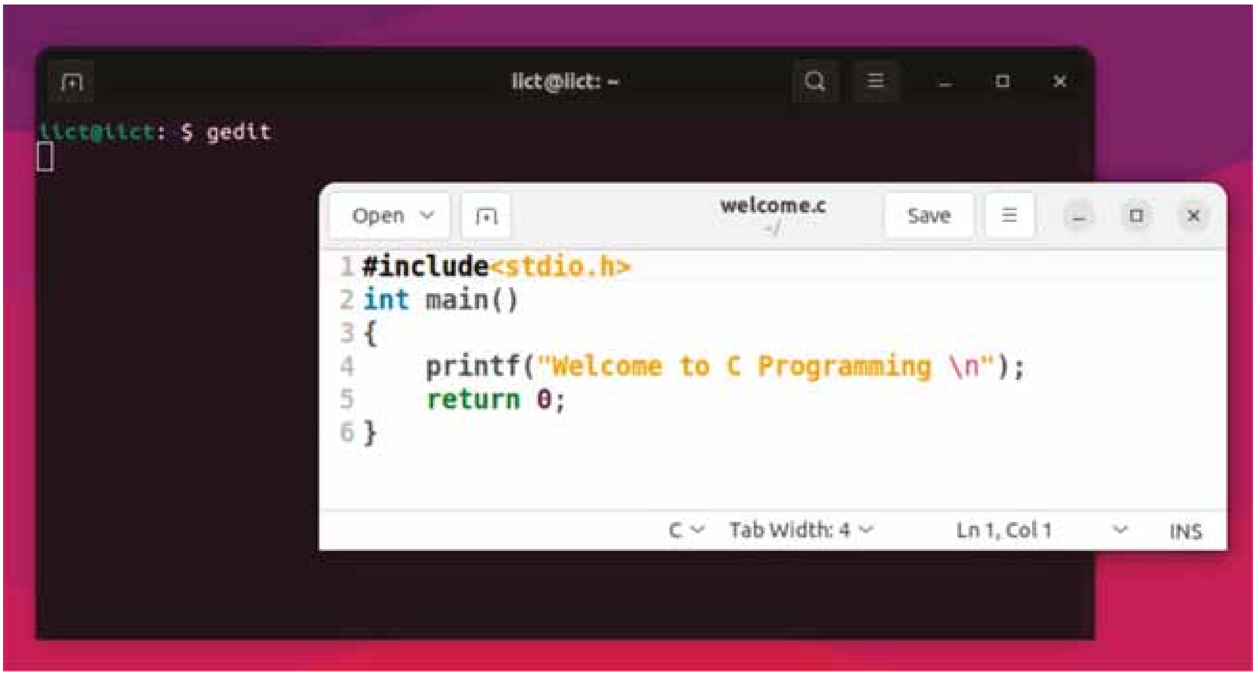
ઉબુન્ટુ ઑપરેટિંગ સિસ્ટમ પર C પ્રોગ્રામ લખવા, કમ્પાઇલ કરવા અને ચલાવવાની પ્રક્રિયાને સમજીએ. આપણે ઉબુન્ટુ OS ટર્મિનલનો ઉપયોગ કરીને C પ્રોગ્રામ લખી અને ચલાવી શકીએ છીએ.

C પ્રોગ્રામ ઉબુન્ટુમાં અમલ કરવા માટેના પગલાં:

1. પ્રોગ્રામ લખો (Write the program)

આકૃતિ 6.2માં દર્શાવ્યા મુજબ gedit જેવા કોઈપણ ટેક્સ્ટ એડિટરને ખોલો અને તેમાં નીચે આપેલ C કોડ લખો:

```
/* C program execution in Ubuntu OS */
#include <stdio.h>
int main()
{
    printf("Welcome to C Programming \n");
    return 0;
}
```



આકૃતિ 6.2 : gedit એડીટરમાં C પ્રોગ્રામ લખવો

2. પ્રોગ્રામને સંગ્રહો (Save the Program)

ફાઈલનું નામ "welcome.c" રાખો. એ સુનિશ્ચિત કરો કે ફાઈલનું ".c" એક્સ્ટેંશન હોય જેથી તે C કોડ ફાઈલ તરીકે ઓળખાય.

3. **કમ્પાઈલ કરો (Compile the Code)**

C કોડને કમ્પાઇલ કરવા માટે GNU કમ્પાઇલર કલેક્શન (GCC) જરૂરી છે. મોટા ભાગના ઉબુન્ટુ (Ubuntu) વર્ઝન GCC સાથે આવે છે, જેમાં C કમ્પાઇલરનો સમાવેશ થાય છે. તમારું ઉબુન્ટુ ટર્મિનલ ખોલો અને નીચે આપેલા (આકૃતિ 6.3ની લાઈન 2) GCC કમાન્ડનો ઉપયોગ કરીને પ્રોગ્રામને કમ્પાઇલ કરો:

gcc welcome.c -o welcome

આ કમાન્ડ "welcome.c" ને કમ્પાઈલ કરે છે અને "welcome" નામની એક એક્ઝિક્યુટેબલ ફાઇલ (executable file) જનરેટ કરે છે.

આકૃતિ 6.3 : ઉબુન્ટુ ટર્મિનલ પર C પ્રોગ્રામને કમ્પાઈલ અને અમલ કરવો

4. **પ્રોગ્રામનો અમલ કરો (Execute the Program)**

કમ્પાઈલ કરેલા પ્રોગ્રામને રન કરવા માટે, નીચેનો કમાન્ડ (આકૃતિ 6.3ની લાઈન 3) દાખલ કરો:

./welcome

અમલ થયા બાદ નીચેનો આઉટપુટ ટર્મીનલ પર જોવા મળશે:

Welcome to C Programming

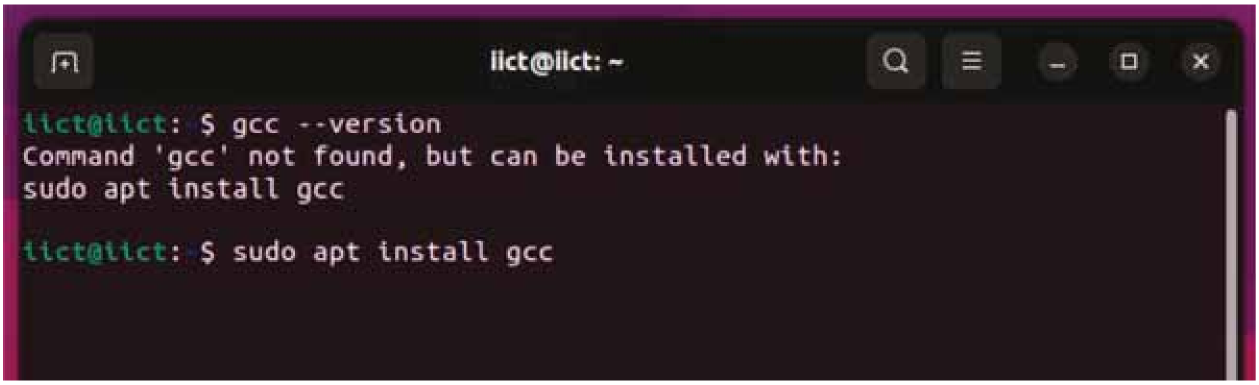
આ પગલાંઓને અનુસરીને, તમે ઉબુન્ટુ ઑપરેટિંગ સિસ્ટમ પર C પ્રોગ્રામને સફળતાપૂર્વક લખી, કમ્પાઈલ અને અમલ કરી શકો છો.

C પ્રોગ્રામને ઉબુન્ટુ (અથવા કોઈપણ Linux - આધારિત) OS પર અમલ કરતી વખતે નીચેના મહત્વપૂર્ણ મુદ્દાઓ ધ્યાનમાં લો:

1. **GCC ઈન્સ્ટોલેશન (GCC Installation):**

સામાન્ય રીતે ઉબુન્ટુ ઓપરેટિંગ સિસ્ટમમાં GCC ઈન્સ્ટોલ થયેલું જ હોય છે.

- **ઈન્સ્ટોલેશન તપાસવા માટે** : તમારા ઉબુન્ટુ ટર્મિનલ પર **gcc --version** કમાન્ડ ચલાવો, જે આકૃતિ 6.4 માં દર્શાવેલ છે.
- **જો ઈન્સ્ટોલ ન હોય, તો** : GCC ને તમારા કમ્પ્યૂટરમાં ઈન્સ્ટોલ કરવા માટે **sudo apt install gcc** કમાન્ડ ચલાવો, જે આકૃતિ 6.4 માં દર્શાવેલ છે.



આકૃતિ 6.4 : ઉબુન્ટુમાં GCC ઈન્સ્ટોલેશન

2. **ફાઈલનું એક્સ્ટેંશન (File extension) :** તમારા પ્રોગ્રામને .c એક્સ્ટેંશન સાથે સેવ કરો (દા.ત., welcome.c) જેથી કમ્પાઈલર તેને C કોડ તરીકે ઓળખે.
3. **કમ્પાઈલેશનની ભૂલો (Compilation errors) :** જો તમારા કોડમાં સિન્ટેક્સની ભૂલો (syntax errors) હશે, તો GCC ભૂલના સંદેશાઓ (error messages) લાઈન નંબર સાથે દર્શાવશે. તેમને ધ્યાનથી વાંચો અને ભૂલો સુધારો.
4. **કેસ-સેન્સિટિવિટી (Case Sensitivity) :** Linux કેસ-સેન્સિટિવ છે (એટલે કે તે નાના અને મોટા અક્ષરો વચ્ચે તફાવત કરે છે). Welcome.c અને welcome.c જેવા ફાઈલના નામોને અલગ-અલગ ફાઈલો તરીકે ગણવામાં આવે છે.
5. **સારી પ્રથા (Good Practice) :** મૂળ ફાઈલો (જેમ કે welcome.c) અને આઉટપુટ ફાઈલો માટે હંમેશા અર્થપૂર્ણ ફાઈલના નામ (meaningful file names) નો ઉપયોગ કરવો જોઈએ.

સારાંશ

આ પ્રકરણ, C પ્રોગ્રામિંગના મુખ્ય ખ્યાલોનો પરિચય આપે છે, જે તેની ઝડપ, કાર્યક્ષમતા અને પોર્ટેબિલિટી (એક પ્લેટફોર્મ પરથી બીજા પ્લેટફોર્મ પર સરળતાથી લઈ જવાની ક્ષમતા) માટે જાણીતી ભાષા છે. આપણે પ્રોગ્રામિંગમાં સૂચનાઓના મહત્વનો અભ્યાસ કર્યો અને કેવી રીતે C ભાષા માનવ તર્ક (human logic) અને મશીન અમલ (machine execution) વચ્ચે સેતુ તરીકે કાર્ય કરે છે તે પણ જોયું. C પ્રોગ્રામની મૂળભૂત રચનાની ચર્ચા કરવામાં આવી, જેમાં કેરેક્ટર સેટ્સ (character sets), કીવર્ડ્સ (keywords), અને આઈડેન્ટિફાયર્સ (identifiers) જેવા માન્ય C કોડ લખવા માટેના આવશ્યક ઘટકોનો સમાવેશ થાય છે. વધુમાં, આપણે C ના ઈતિહાસ વિશે પણ જાણકારી મેળવી, જે ડેનિસ રિચી દ્વારા વિકસાવવામાં આવી હતી, અને આધુનિક સોફ્ટવેર ડેવલપમેન્ટમાં તેની સતત પ્રાસંગિકતા (relevance) વિશે ચર્ચા કરી. GCC કમ્પાઈલર અને ટેક્સ્ટ એડિટર્સ જેવા મુખ્ય સોફ્ટવેર C પ્રોગ્રામ લખવા અને ચલાવવા કેવી રીતે વપરાય તે પણ જોયું. લીનક્ષ સિસ્ટમ પર C કોડને કેવી રીતે કમ્પાઈલ અને એક્ઝિક્યુટ કરવો તે દર્શાવવા માટે વ્યવહારુ ઉદાહરણો પૂરા પાડવામાં આવ્યા.

સ્વાધ્યાય

1. C પ્રોગ્રામમાં #include ડાયરેક્ટીવનો ઉપયોગ શું છે?
2. C માં main() ફંક્શન શા માટે મહત્વનું છે?
3. C પ્રોગ્રામિંગમાં return 0; સ્ટેટમેન્ટ શું દર્શાવે છે?

4. C પ્રોગ્રામમાં કર્લી બ્રેસિસ {} ની ભૂમિકા શું છે?

5. C પ્રોગ્રામિંગમાં તમે આઉટપુટ કેવી રીતે દર્શાવો છો?

6. C પ્રોગ્રામિંગના ઈતિહાસ વિશે જણાવો.

7. C પ્રોગ્રામિંગમાં કેરેક્ટર સેટ (character set) ની યાદી બનાવો.

8. C પ્રોગ્રામિંગમાં કેસ સેન્સિટિવિટી (Case Sensitivity) વિશે ઉદાહરણ સાથે વિગતવાર સમજાવો.

9. લીનક્ષમાં C પ્રોગ્રામ ચલાવવા માટેના પગલાંની યાદી બનાવો.

10. C પ્રોગ્રામ ફાઈલને સેવ કરવા માટેના પગલાંની યાદી બનાવો.

11. સાચું કે ખોટું જણાવો.

(1) C પ્રોગ્રામમાં main() ફંક્શન વૈકલ્પિક છે.

(2) int અને float જેવા કીવર્ડ્સનો ઉપયોગ ચલનામ (variable names) તરીકે કરી શકાય છે.

(3) #include ડાયરેક્ટીવનો ઉપયોગ પ્રોગ્રામમાં લાઈબ્રેરીઓનો સમાવેશ કરવા માટે થાય છે.

(4) C એ કેસ-સેન્સિટિવ (case-sensitive) પ્રોગ્રામિંગ ભાષા નથી.

(5) C માં આઉટપુટ દર્શાવવા માટે printf() નો ઉપયોગ થાય છે.

12. ખાલી જગ્યાઓ પૂરો.

(1) C પ્રોગ્રામની શરૂઆત ____________ ફંક્શનથી થાય છે.

(2) C પ્રોગ્રામમાં ____________ ડાયરેક્ટીવ હેડર ફાઈલનો સમાવેશ કરે છે.

(3) C માં વિધાનનો અંત લાવવા માટે ____________ પ્રતીકનો ઉપયોગ થાય છે.

(4) સ્ક્રીન પર લખાણ પ્રિન્ટ કરવા માટે ____________ નો ઉપયોગ થાય છે.

(5) C પ્રોગ્રામિંગમાં કીવર્ડ્સનો ઉપયોગ ____________ તરીકે કરી શકાતો નથી.

13. બહુવિકલ્પી પ્રશ્નો. સૌથી યોગ્ય જવાબ પસંદ કરો.

(1) નીચનામાંથી કયું પ્રતિક C માં હેડર ફાઈલ સામેલ કરવા ઉપયોગી છે?

(a) $ (b) @ (c) # (d) %

(2) printf("Hello \t World!"); નું આઉટપુટ શું હશે?

(a) Hello World? (b) Hello World

(c) Hello World! (d) Error

(3) નીચેનામાંથી ક્યો કીવર્ડ પૂર્ણાંક ચલ ઘોષિત કરવા વપરાય છે?

(a) int (b) float (c) char (d) include

(4) નીચાનામાંથી ક્યો માન્ય આઈડેન્ટિફાયર નથી?

(a) totalMarks (b) 1value (c) _value (d) value1

(5) C માં return 0; વિધાનનો હેતુ શું છે?

(a) પ્રોગ્રામ શરુ કરવો (b) main() ફંક્શનનો અંત કરવો

(c) હેડર સામેલ કરવા (d) આઉટપુટ પ્રિન્ટ કરવા

(6) નીચેનામાંથી કયું ચલના નામ તરીકે માન્ય છે?

(a) 123abc (b) _count (c) float (d) -value

(7) નીચેનામાંથી ક્યો કીવર્ડ અપૂર્ણાંક નંબર ઘોષિત કરવા વપરાય છે?

(a) int (b) float (c) char (d) None

(8) નીચેનામાંથી કઈ એસ્કેપ સીક્વન્સ નવી લાઈન માટે વપરાય છે?

(a) \t (b) \n (c) \ (d) \r

(9) નીચેનામાંથો ક્યો કોડ બ્લોક બનાવવા વપરાય છે?

(a) () (b) [] (c) {} (d) <>

(10) નીચેનામાંથી ક્યો C માં કીવર્ડ છે?

(a) loop (b) int (c) print (d) name

પ્રાયોગિક સ્વાધ્યાય

1. printf() ફંક્શનનો ઉપયોગ કરીને તમારું પૂરું નામ પ્રિન્ટ કરતો એક C પ્રોગ્રામ લખો.

2. "Hello, India!" પ્રિન્ટ કરતો એક C પ્રોગ્રામ લખો.

3. નીચે પ્રમાણે મેસેજ પ્રિન્ટ કરતો એક C પ્રોગ્રામ લખો.

Hello,
World!

C ભાષામાં ડેટા પ્રકારો

પરિચય

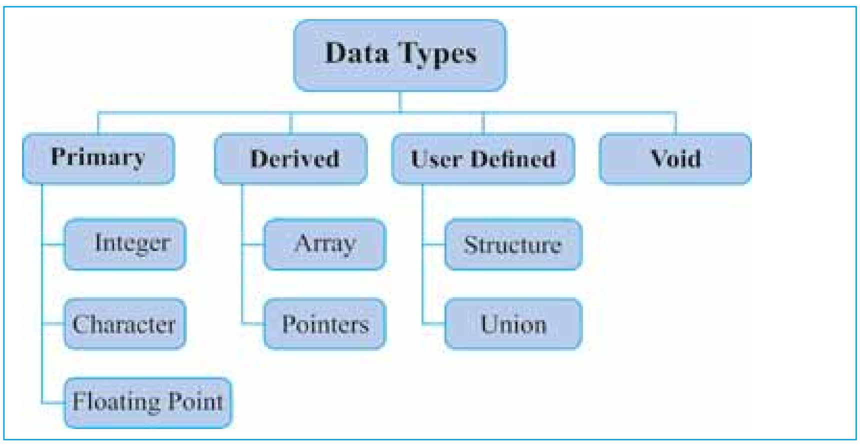
અગાઉના પ્રકરણે C પ્રોગ્રામિંગનો પરિચય આપ્યો હતો. હવે આપણે ડેટા પ્રકારોને સમજીને આપણી C પ્રોગ્રામિંગ યાત્રાને આગળ વધારીએ. C પ્રોગ્રામિંગમાં, ડેટા પ્રકારો ખૂબ જ મહત્વપૂર્ણ છે કારણ કે તે કમ્પ્યૂટરને જણાવે છે કે આપણે કયા પ્રકારનો ડેટા વાપરી રહ્યા છીએ. તે યોગ્ય વસ્તુનો સંગ્રહ કરવા માટે યોગ્ય કન્ટેનર (પાત્ર) વાપરવા જેવું છે - ઉદાહરણ તરીકે, પાણી માટે બોટલ અને ખાંડ માટે બરણી. C પ્રોગ્રામિંગમાં પૂર્ણાંક સંખ્યાઓ, દશાંશ (અપૂર્ણાંક) સંખ્યાઓ અને અક્ષરો જેવા વિવિધ પ્રકારના મૂલ્યોનો સંગ્રહ કરવા માટે જુદા જુદા ડેટા પ્રકારોનો ઉપયોગ થાય છે. દરેક ડેટા પ્રકાર કમ્પ્યૂટરને જણાવે છે કે તેને કેટલી મેમરીની જરૂર છે અને તે ડેટાનો ઉપયોગ કેવી રીતે કરવો. યોગ્ય ડેટા પ્રકારનો ઉપયોગ આપણને સચોટ અને કાર્યક્ષમ પ્રોગ્રામ લખવામાં મદદ કરે છે. આ પ્રકરણમાં C પ્રોગ્રામિંગમાં વિવિધ ડેટા પ્રકારોના ઉપયોગનો પરિચય આપવામાં આવ્યો છે, જેમાં મૂળભૂત (basic), ડિરાઇવ્ડ (derived), યુઝર-નિર્મિત (user-defined) અને ખાલી (empty) (void) ડેટા પ્રકારોનો સમાવેશ થાય છે. આપણે સ્ટોરેજ ક્લાસ (storage class) વિશે પણ ચર્ચા કરીશું, જે મેમરીમાં ચલ કેવી રીતે વર્તે તે નિયંત્રિત કરે છે. આપણે જાણીશું કે ડેટા પ્રકારો શું છે, C માં તેમનો ઉપયોગ કેવી રીતે થાય છે અને આપણા પ્રોગ્રામ માટે યોગ્ય ડેટા પ્રકાર કેવી રીતે પસંદ કરવો.

ડેટા પ્રકારો (Data Types)

C પ્રોગ્રામિંગમાં ડેટા ટાઇપને ચાર મુખ્ય જૂથોમાં વહેંચવામાં આવ્યા છેઃ

1. **મૂળભૂત ડેટા ટાઇપ (Primary Data Types) :** આ બેઝિક ડેટા ટાઇપ છે, જે સરળ મૂલ્યો રાખવા માટે વપરાય છે જેમ કે પૂર્ણાંક *(int)*, અક્ષર *(char)* અને અપૂર્ણાંક સંખ્યા *(float)*.
2. **ડિરાઇવ્ડ ડેટા ટાઇપ (Derived Data Types) :** આ મૂળભૂત ડેટા ટાઇપ પરથી બનેલા હોય છે અને તેમાં એરે (array) અને પોઈન્ટર (pointer) નો સમાવેશ થાય છે, જે આપણને વધુ જટિલ ડેટાને સંભાળવામાં મદદ કરે છે.
3. **યુઝર-નિર્મિત ડેટા ટાઇપ (User-Defined Data Types):** આ પ્રોગ્રામરને પોતાની જરૂરિયાત મુજબ નવા ડેટા માળખાં (data structures) બનાવવા દે છે, જેમ કે *struct, union* વગેરે.
4. **ખાલી ડેટા ટાઇપ (Void Data Type) :** *void* પ્રકાર જે "કોઈ મૂલ્ય નથી" એવો અર્થ દર્શાવે છે અને સામાન્ય રીતે પરત મૂલ્ય ન આપતા ફંક્શન માટે વપરાય છે.

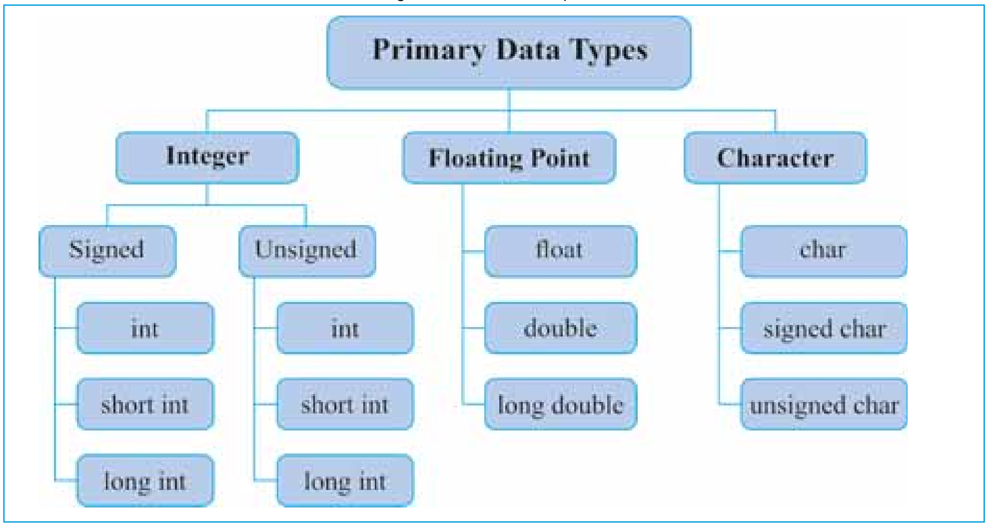
તદુપરાંત, સ્ટોરેજ ક્લાસ (storage class) જેમ કે ઓટો (*auto*), સ્ટેટીક (*static*), ગ્લોબલ (*global*) અને રજીસ્ટર (*register*) નક્કી કરે છે કે કોઈ ચલ મેમરીમાં કેટલો સમય રહેશે અને તે પ્રોગ્રામના ક્યાં ભાગમાં ઉપલબ્ધ હશે. આકૃતિ 7.1 વિવિધ ડેટા ટાઇપ દર્શાવે છે.



આકૃતિ 7.1 : ડેટા પ્રકારો

મૂળભૂત ડેટા ટાઈપ (Primary Data Types)

ચાલો, આકૃતિ 7.2 મુજબ મૂળભૂત ડેટા ટાઈપ (Primary Data Types) સમજીએ. મૂળભૂત ડેટા ટાઈપ એ એવા મૂળ પ્રકારના ડેટા છે, જે C પ્રોગ્રામિંગમાં પાયારૂપ છે. જેમ, આપણે અગાઉ ચર્ચા કરી હતી કે C ભાષામાં વિવિધ પ્રકારના ડેટા અલગ-અલગ પ્રકારના મૂલ્યો (સંખ્યા, અક્ષર વગેરે) સંગ્રહવા માટે વપરાય છે. C માં મુખ્ય ચાર પ્રકારના મૂળભૂત ડેટા ટાઈપ છે: *int* (integer) - પૂર્ણ સંખ્યાઓ સંગ્રહવા માટે, *float* (floating-point) - અપૂર્ણાંક (દશાંશ) સંખ્યાઓ સંગ્રહવા માટે, *char* (character) - એક અક્ષર અથવા પ્રતીક (single letter or symbol) સંગ્રહવા માટે, *double* - વધુ ચોકસાઈવાળી અપૂર્ણાંક સંખ્યાઓ (extra-precise decimal values) માટે. પ્રત્યેક પ્રકારનો પોતાનો અલગ ઉપયોગ અને મેમરીની જરૂરિયાત હોય છે, જે ટેબલ 7.1માં દર્શાવવામાં આવી છે.



આકૃતિ 7.2 : મૂળભૂત ડેટા પ્રકારો

ડેટા ટાઈપ (Data Type)	વર્ણન	ઉદાહરણ
int	પૂર્ણાંક સંખ્યાઓ (ધન અથવા ઋણ) સંગ્રહે છે.	int rollNumber = 25;
float	મર્યાદિત ચોકસાઈ સાથે અપૂર્ણાંક (દશાંશ) સંખ્યાઓ સંગ્રહે છે.	float height = 5.6;
double	float કરતાં વધુ ચોકસાઈ સાથે અપૂર્ણાંક સંખ્યાઓ સંગ્રહે છે.	double distance = 123.456789;
char	' ' માં મૂકેલ એક અક્ષર (single character) ને સંગ્રહે છે.	char grade = 'A';

ટેબલ 7.1 : મૂળભૂત ડેટા પ્રકારો

int ડેટા પ્રકારો (int Data Type)

int ડેટા પ્રકાર C પ્રોગ્રામિંગમાં સૌથી મહત્વપૂર્ણ અને વારંવાર વપરાતા ડેટા પ્રકારોમાંનો એક છે. તે પૂર્ણાંક સંખ્યાઓ (whole numbers) - ધન, ઋણ કે શૂન્ય - સંગ્રહવા માટે વપરાય છે. ઉદાહરણ તરીકે, જો આપણે લખીએ: **int age = 15;** તો તેનો અર્થ થાય છે કે કમ્પ્યૂટરને age નામના ચલમાં પૂર્ણાંક 15 સંગ્રહવા કહેવામાં આવ્યું છે. C પ્રોગ્રામિંગમાં *int* ના વિવિધ પ્રકારો ઉપલબ્ધ છે (જે ટેબલ 7.2માં દર્શાવેલ છે), જે નાની કે મોટી સંખ્યાઓને સંભાળી શકે છે. આથી, પ્રોગ્રામરને પોતાની જરૂરિયાત મુજબ યોગ્ય પ્રકાર પસંદ કરવાની લવચીકતા (flexibility) મળે છે. વસ્તુઓની ગણતરી કરવી હોય, સ્કોર ટ્રેક કરવો હોય કે માત્રાઓ (quantities) સંગ્રહવી હોય — તમામ કિસ્સાઓમાં પૂર્ણ સંખ્યાઓ સાથે કામ કરવા માટે *int* ડેટા પ્રકાર C માં સૌથી પ્રાથમિક પસંદગી છે.

વાક્યરચના (Syntax):

```
data-type identifier;
Example : int number = 3;
```

પ્રકાર (Type)	વર્ણન	ઉદાહરણ
int	મૂળભૂત પૂર્ણાંક પ્રકાર (પૂર્ણ સંખ્યાઓ માટે)	int x = 100;
short int	નાની સાઇઝનો પૂર્ણાંક પ્રકાર (સામાન્ય રીતે 2 બાઇટ)	short int a = 10;
long int	મોટી સાઇઝનો પૂર્ણાંક પ્રકાર (સામાન્ય રીતે 4 બાઇટ)	long int b = 100000;
unsigned int	ફક્ત ધન સંખ્યાઓ માટે (ઋણ મૂલ્યો નહીં)	unsigned int c = 50;
signed int	ધન અને ઋણ બંને પ્રકારની સંખ્યાઓ માટે (મૂળભૂત રીતે)	signed int d = –25;

ટેબલ 7.2 : int ડેટા પ્રકારો

```
/* C program to illustrate sum of two integer numbers.*/
#include <stdio.h>
int main()
{
     int number1 = 25;
     int number2 = 35;
     int sum;
     sum = number1 + number2;
     printf("The sum of %d and %d is %d", number1, number2, sum);
     return 0;
}
Result:
The sum of 25 and 35 is 60
```

ચાલો, આ C પ્રોગ્રામને સમજીએ. C પ્રોગ્રામમાં *main()* ફંક્શન આવશ્યક છે — પ્રોગ્રામનું અમલીકરણ (execution) અહીંથી શરૂ થાય છે. *main()* ના { } કર્લી બ્રેકેટની અંદર, આપણે બે પૂર્ણાંક ચલ number1 અને number2 ઘોષિત કરીએ છીએ અને તેમને અનુક્રમે 25 અને 35 મૂલ્યો આપીએ છીએ. ત્યારબાદ, આપણે એક અન્ય ચલ sum લઈએ છીએ, જેમાં આ બંને સંખ્યાઓનો સરવાળો સંગ્રહ કરાય છે. વાસ્તવિક સરવાળો sum = number1 + number2 લાઇનમાં થાય છે. છેલ્લે, *printf()* ફંક્શનનો ઉપયોગ કરીને પરિણામ બતાવીએ છીએ. અહીં %d ફોર્મેટ સ્પેસિફાયરનો અર્થ છે "પૂર્ણ સંખ્યા દર્શાવો". આ સરળ ઉદાહરણ બતાવે છે કે કેવી રીતે આપણે સંખ્યાઓ સંગ્રહવી, ગણતરી કરવી અને પરિણામ દર્શાવવું - આ બધું C પ્રોગ્રામિંગમાં કરી શકીએ છીએ. હવે ચાલો, unsigned integer સંખ્યાઓ વાપરતો પ્રોગ્રામ લખીએ.

```
/* C program to illustrate use of unsigned int.*/

#include <stdio.h>
int main()
{
     unsigned int a = 300;
     printf("The value of unsigned int a is: %u\n", a);

     return 0;
}
Result:
The value of unsigned int a is: 300
```

unsigned int ડેટા પ્રકાર ફક્ત ધન પૂર્ણ સંખ્યાઓ (positive whole numbers) જ સંગ્રહ કરે છે તે ઋણ મૂલ્યો (negative values) સંગ્રહ કરી શકતું નથી. આ પ્રોગ્રામમાં, આપણે ચલ a ને મૂલ્ય 300 આપીએ છીએ અને તેને %u ફોર્મેટ સ્પેસિફાયરથી પ્રિન્ટ કરીએ છીએ, જે ખાસ કરીને unsigned integers માટે વપરાય છે. નીચેનો પ્રોગ્રામ signed integers નો ઉપયોગ દર્શાવે છે.

```
/* C program to illustrate use of signed int.*/

#include <stdio.h>
int main()
{
    signed int b = -200;
    printf("The value of signed int b is: %d\n", b);

    return 0;
}
Result:
The value of signed int b is: -200
```

signed int એ C ભાષામાં મૂળભૂત (default) પૂર્ણાંક પ્રકાર છે. તે ધન તેમજ ઋણ બંને પ્રકારની સંખ્યાઓ સંગ્રહી શકે છે. અહીં, આપણે વેરિયેબલ b ને મૂલ્ય -200 આપીએ છીએ અને તેને %d ફોર્મેટ સ્પેસિફાયર વડે પ્રિન્ટ કરીએ છીએ, જે signed integers માટે વપરાય છે.

float અને double ડેટા પ્રકારો (float and double Data Types)

C માં *float* ડેટા પ્રકારનો ઉપયોગ દશાંશ ચિહ્નવાળી સંખ્યાઓ સંગ્રહ કરવા માટે થાય છે, જેમ કે 3.14 અથવા 89.5. તે C ના મૂળભૂત ડેટા પ્રકારમાંનો એક છે અને જ્યારે આપણને માપ, ટકાવારી અથવા દશાંશની ચોકસાઈની જરૂર હોય તેવી કોઈપણ ગણતરીઓ સાથે કામ કરવાની જરૂર હોય ત્યારે તે યોગ્ય છે. ઉદાહરણ તરીકે, જો આપણે કોઈનો ટેસ્ટ એવરેજ 87.65 ને સંગ્રહ કરવા માંગતા હોઈએ, તો આપણે આનો ઉપયોગ કરીશું: **float average = 87.65;.** રોજિંદી મોટાભાગની દશાંશ ગણતરીઓ માટે *float* પ્રકાર યોગ્ય છે, પરંતુ જ્યારે આપણને તેનાથી પણ વધુ ચોકસાઈની જરૂર હોય, ત્યારે C *double* નામનો બીજો પ્રકાર ઓફર કરે છે, જે ટેબલ 7.3 માં દર્શાવેલ છે. આ દશાંશ સંખ્યાના પ્રકારો કોઈપણ પ્રોગ્રામ માટે આવશ્યક છે, જે વાસ્તવિક દુનિયાના મૂલ્યો સાથે કામ કરે છે, જ્યાં ફક્ત પૂર્ણાંક સંખ્યાઓ પૂરતી નથી!

પ્રકાર (Type)	વર્ણન	ઉદાહરણ
float	દશાંશ મૂલ્યોને ~6 અંકોની ચોકસાઈ સાથે સંગ્રહે છે (4 બાઈટ)	float price = 10.75;
double	દશાંશ મૂલ્યોને ~15 અંકોની ચોકસાઈ સાથે સંગ્રહે છે (8 બાઈટ)	double pi = 3.14159265359;
long double	દશાંશ મૂલ્યોને તેનાથી પણ વધુ ચોકસાઈ સાથે સંગ્રહે છે (સામાન્ય રીતે 10 બાઈટ)	long double value = 3.141592653589793238;

ટેબલ 7.3 : float અને double ડેટા પ્રકારો

ચાલો આપણે તેમને ઉદાહરણ સાથે સમજીએ.

```
/* C program to illustrate multiplication of two float or double
numbers. */
#include <stdio.h>
int main()
{
  float num1 = 5.5;
  double num2 = 4.25;
  double result;
  result = num1 * num2;
  printf("The multiplication of %f and %f is %f", num1, num2,result);
  return 0;
}
Result:
The multiplication of 5.500000 and 4.250000 is 23.375000
```

ચાલો ઉપરના પ્રોગ્રામને સમજીએ, જ્યાં *float* ચલ num1 ઘોષિત કરવામાં આવ્યો છે અને તેને 5.5 ની કિંમત આપવામાં આવી છે. તેવી જ રીતે, એક *double* ચલ num2 ઘોષિત કરવામાં આવ્યો છે અને તેને 4.25 ની કિંમત સાથે આરંભિત કરવામાં આવ્યો છે. ગુણાકારના પરિણામને સંગ્રહ કરવા માટે result નામનો બીજો *double* ચલ ઘોષિત કરવામાં આવ્યો છે. વિધાન (statement) result = num1 * num2; એ *float* અને *double* મૂલ્યોનો ગુણાકાર કરે છે અને પરિણામને result ચલમાં સંગ્રહ કરે છે. છેલ્લે, પ્રોગ્રામ return 0; સાથે સમાપ્ત થાય છે, જે સફળતાપૂર્વક અમલ થયાનો સંકેત આપે છે. ચાલો એક વધુ ઉદાહરણ લઈએ.

```
/* C program to find radius of a circle. */
#include <stdio.h>
#include <math.h>
int main()
{
    double area = 78.5;  double pi = 3.14;
    double radius;
    radius = sqrt(area / pi);
    printf("Radius using double: %f\n", radius);

    return 0;
}
Result:
Radius using double: 5.000000
```

ચાલો, આપેલા ક્ષેત્રફળ (area) અને પાઈ π (pi) ના મૂલ્યનો ઉપયોગ કરીને વર્તુળની ત્રિજ્યા શોધવા માટેના ઉપરના C પ્રોગ્રામને સમજીએ. પ્રોગ્રામ *sqrt()* ફંક્શનને ઉપયોગ કરવા માટે <*math.h*> ને સામેલ કરીને શરૂ થાય છે. યાદ રાખો કે <*math.h*> એ ગાણિતિક હેડર ફાઈલ છે, જે વર્ગમૂળ (square roots) ની ગણતરી માટે જરૂરી ફંક્શન ધરાવે છે. *double* પ્રકારના ત્રણ ચલ ઘોષિત કરવામાં આવ્યા છેઃ area, pi અને radius. area નું મૂલ્ય 78.5 તરીકે આપવામાં આવ્યું છે, અને pi ને 3.14 મૂલ્ય આપેલ છે. ત્રિજ્યા શોધવા માટે, radius = sqrt(area/pi); સૂત્રનો ઉપયોગ કરવામાં આવ્યો છે, જે ક્ષેત્રફળને pi વડે ભાગે છે અને પછી

પરિણામનું વર્ગમૂળ લે છે. *sqrt()* ફંક્શન આ વર્ગમૂળની પ્રક્રિયા કરે છે. છેલ્લે, પરિણામને સ્ક્રીન પર %f ફોર્મેટ સ્પેસિફાયરનો ઉપયોગ કરીને પ્રિન્ટ કરવામાં આવેલ છે, જે ખાસ કરીને *double* ટાઇપના ચલનું મૂલ્ય દર્શાવવા માટે વપરાય છે.

char ડેટા પ્રકારો (char Data Types)

char ડેટા પ્રકારનો ઉપયોગ અક્ષરો, અંકો અથવા પ્રતીકો જેવા એક જ અક્ષર (single characters) સંગ્રહવા માટે થાય છે, જેને હંમેશા સિંગલ અવતરણ ચિહ્નો (‘’) માં લખવામાં આવે છે. ઉદાહરણ તરીકે, જ્યારે આપણે char grade = ‘A’; લખીએ છીએ, ત્યારે આપણે grade નામનો એક ચલ બનાવી રહ્યા છીએ જે ‘A’ અક્ષર ધરાવે છે. આ સરળ છતાં મહત્ત્વપૂર્ણ ડેટા ટાઇપ આપણને વ્યક્તિગત કીબોર્ડ અક્ષરો સાથે કામ કરવાની મંજૂરી આપે છે, પછી ભલે આપણે કોઈના પ્રથમ અક્ષર (initial) અથવા વિરામચિહ્ન (punctuation mark) સંગ્રહી રહ્યા હોઈએ. *char* પ્રકાર ખાસ કરીને ટેક્સ્ટ ઇનપુટ, પાસવર્ડ અથવા અક્ષરો અને પ્રતીકોને મેનેજ કરવાની જરૂર હોય તેવા કોઈપણ પ્રોગ્રામ સાથે કામ કરવા માટે ઉપયોગી છે. પ્રોગ્રામિંગમાં, રેન્જ (Range) એટલે કે ચલનું લઘુત્તમ (minimum) થી મહત્તમ (maximum) મૂલ્ય, *signed char* ની રેન્જ -128 થી 127 સુધીની હોય છે, જ્યારે *unsigned char* ની રેન્જ 0 થી 255 સુધીની હોય છે. ટેબલ 7.4 *char* ડેટા પ્રકાર દર્શાવેલા છે. યાદ રાખો કે *char* ડેટા પ્રકારનો ઉપયોગ 8-બિટ *signed* અથવા *unsigned* પૂર્ણાંક (integers) તરીકે પણ થઈ શકે છે.

પ્રકાર (Type)	વર્ણન	ઉદાહરણ
char	એક જ અક્ષર સંગ્રહ કરે છે (કમ્પાઇલર પર આધાર રાખીને મૂળભૂત રીતે signed અથવા unsigned)	char letter=‘B’;
signed char	અક્ષરોને signed મૂલ્યો તરીકે સંગ્રહ કરે છે (-128 થી 127)	signed char sign = 'z';
unsigned char	અક્ષરોને unsigned મૂલ્યો તરીકે સંગ્રહ કરે છે (0 થી 255)	unsigned char symbol = '$';

ટેબલ 7.4 : char ડેટા પ્રકારો

ચાલો, આપણે એક ઉદાહરણ લઈએ.

```
/* C program to illustrate use of character.*/
#include <stdio.h>
int main()
{
    char grade = 'A';
    printf("Your grade is: %c", grade);

    return 0;
}

Result:
Your grade is: A
```

ચાલો ઉપરના પ્રોગ્રામને સમજીએ. અહીં grade નામનો એક *char* પ્રકારનો ચલ ઘોષિત કરવામાં આવ્યો છે અને તેમાં ‘A’ સંગ્રહવામાં આવ્યો છે. કેરેક્ટર હંમેશા single quotes (‘ ’)માં મુકવામાં આવે છે. પછી *printf()* ફંક્શનનો ઉપયોગ કરીને grade ચલમાં રહેલ અક્ષર પ્રિન્ટ કરવામાં આવે છે, જેમાં %c ફોર્મેટ સ્પેસિફાયર ઉપયોગ કરેલ છે, જે ખાસ કરીને અક્ષરો પ્રિન્ટ કરવા માટે વપરાય છે. અંતે, પ્રોગ્રામ return 0; વડે પૂર્ણ થાય છે, જે દર્શાવે છે કે પ્રોગ્રામ સફળતાપૂર્વક ચલાવવામાં આવ્યો છે.

ડિરાઈવ્ડ ડેટા પ્રકારો (Derived Data Types)

ડિરાઈવ્ડ (Derived) ડેટા પ્રકાર તે છે જે મૂળભૂત અથવા પ્રાથમિક ડેટા પ્રકાર પરથી બનાવેલ ડેટા પ્રકાર છે. તે અનેક મૂલ્યો અથવા જટિલ ડેટા સ્ટ્રક્ચર કાર્યક્ષમ રીતે સંગ્રહવા માટે વપરાય છે. ડિરાઈવ્ડ ડેટા ટાઈપ લવચીકતા (flexibility) પ્રદાન કરે છે અને વાસ્તવિક પ્રોગ્રામિંગમાં વ્યાપકપણે વપરાય છે, ખાસ કરીને અરે, મેમરી એડ્રેસ અને ગ્રુપ ડેટાને સંભાળવા માટે. ટેબલ 7.5માં ડિરાઈવ્ડ ડેટા ટાઈપની યાદી આપવામાં આવી છે.

પ્રકાર (Type)	વર્ણન	ઉદાહરણ
Array	એકસરખા ડેટા ટાઈપના મૂલ્યોનો સંગ્રહ, જે સતત (contiguous) મેમરીમાં સંગ્રહ થયેલ હોય છે.	int numbers[5] = {1, 2, 3, 4, 5};
Pointer	બીજા ચલના મેમરી એડ્રેસને સંગ્રહ કરે છે.	int *ptr; ptr = #

ટેબલ 7.5 : ડિરાઈવ્ડ ડેટા પ્રકારો

યુઝર-નિર્મિત ડેટા પ્રકારો (User-Defined Data Types)

યુઝર-નિર્મિત (User-defined) ડેટા પ્રકાર પ્રોગ્રામરને મૂળભૂત અથવા ડિરાઈવ્ડ (derived) પ્રકારના આધાર પર પોતાના ડેટા પ્રકારો બનાવવા દે છે. આ કસ્ટમ (ખાસ બનાવેલ) પ્રકાર જટિલ ડેટાને સુવ્યવસ્થિત કરવામાં અને કોડની વાંચનક્ષમતા તથા જાળવણી ક્ષમતા સુધારવામાં મદદરૂપ બને છે. C ભાષામાં મુખ્ય યુઝર-નિર્મિત ડેટા ટાઈપ છે: *struct* અને *union*. તેમની યાદી ટેબલ 7.6માં આપવામાં આવી છે.

પ્રકાર (Type)	વર્ણન	ઉદાહરણ

struct	વિવિધ પ્રકારના ચલોને એક જ નામ હેઠળ જૂથબદ્ધ કરે છે.	struct Student {
		int id;
		char name [20];
		};

union	એક જ મેમરી સ્થાનમાં (એક સમયે એક પ્રકારના) વિવિધ ડેટા ટાઈપ સંગ્રહવાની મંજૂરી આપે છે.	union Data {
		int i;
		float f;
		};

ટેબલ 7.6 : યુઝર-નિર્મિત ડેટા પ્રકારો

ચાલો એક ઉદાહરણ સાથે એરેને સમજીએ. એરે એકસરખા ડેટા પ્રકારની કિંમતોનો સંગ્રહ છે, જે સતત મેમરી સ્થાનોમાં સંગ્રહ થયેલ છે અને તેમને ઈન્ડેક્સ (index-સૂચક) નો ઉપયોગ કરીને મેળવી શકાય છે.

વાક્યરચના (Syntax):

```
data_type array_name[size];

Example: int marks[5] = {90, 80, 70, 60, 50};
```

```
/*C program to illustrate use of an array*/
#include <stdio.h>
int main()
{
    int numbers[5] = {10, 20, 30, 40, 50};
    printf("The elements of the array are:\n");
    printf("%d ", numbers[0]);
    printf("%d ", numbers[1]);
    printf("%d ", numbers[2]);
    printf("%d ", numbers[3]);
    printf("%d\n", numbers[4]);
    return 0;
}
Result:
The elements of the array are:
10 20 30 40 50
```

પ્રોગ્રામની શરૂઆતમાં numbers નામની એક પૂર્ણાંક સંખ્યાઓની એરે ઘોષિત કરવામાં આવી છે જેમાં 5 કિંમતો છે: 10, 20, 30, 40 અને 50. C ભાષામાં એરે આપણને એકસરખા પ્રકારના અનેક મૂલ્યોને એક જ ચલમાં સંગ્રહવાની સુવિધા આપે છે, જેમાં કિંમતોને ઇન્ડેક્સ (index) દ્વારા મેળવવામાં આવે છે, અને ઇન્ડેક્સની ગણતરી 0 થી શરૂ થાય છે. આ પ્રોગ્રામમાં એરેની દરેક કિંમતને તેના ઇન્ડેક્સ દ્વારા સીધી જ ઉપયોગમાં લીધેલ છે. ઉદાહરણ તરીકે, numbers[0] પ્રથમ કિંમત મેળવે છે, numbers[1] બીજી કિંમતને, અને આમ આગળ વધે છે. બધી કિંમતો પ્રિન્ટ કરવા માટે લૂપનો ઉપયોગ કરવાની જગ્યાએ, અહીં અલગ-અલગ *printf()* નો ઉપયોગ કરીને દરેક કિંમત પ્રિન્ટ કરવામાં આવી છે. આ રીત સરળ છે અને નાની એરે માટે યોગ્ય છે, પરંતુ મોટી એરે માટે અકાર્યક્ષમ બને છે. આ ઉદાહરણ શરૂઆતના શીખનારાઓને C પ્રોગ્રામિંગમાં એરેની ઘોષણા (declaration), પ્રારંભિકકરણ (initialization) અને અલગ-અલગ દરેક કિંમતોને કેવી રીતે જુદી-જુદી ઉપયોગમાં લેવી તે સમજવામાં મદદ કરે છે.

સ્ટોરેજ ક્લાસીસ અને તેની અગત્યતા (Storage Classes and Their Significance)

સ્ટોરેજ ક્લાસિસ ચલનો વ્યાપ (scope), જીવનકાળ (lifetime) અને લિંકેજ (linkage) નક્કી કરે છે. તે કમ્પાઇલરને પ્રોગ્રામની અંદર ચલની મેમરી અને ઉપલબ્ધતા કેવી રીતે સંભાળવી તે સમજવામાં મદદ કરે છે. C ભાષામાં સ્ટોરેજ ક્લાસિસના ચાર મુખ્ય પ્રકાર છે: auto, register, static, અને global. દરેક સ્ટોરેજ ક્લાસના પોતાના નિયમો હોય છે, જે ચલને ક્યાં ઉપલબ્ધ કરી શકાય અને તે મેમરીમાં કેટલો સમય ટકી રહે છે તે નક્કી કરે છે. ટેબલ 7.7 માં આ સ્ટોરેજ ક્લાસની યાદી આપવામાં આવી છે.

સ્ટોરેજ ક્લાસ	કીવર્ડ	વ્યાપ (scope) અને જીવનકાળ (life)	ઉદાહરણ
Automatic (local)	auto	બ્લોક/ફંક્શન માટે સ્થાનિક (local); બ્લોક/ફંક્શન સમાપ્ત થાય ત્યાં સુધી અસ્તિત્વ ધરાવે છે.	int a = 5;
Register	register	ફંક્શન માટે સ્થાનિક પરંતુ ઝડપ મેળવવા માટે તેને CPU રજિસ્ટરમાં સંગ્રહ કરવામાં આવે છે.	register int speed = 10;
Static	static	ફંક્શન માટે સ્થાનિક પરંતુ કોલ વચ્ચે તેનું અગાઉનું મૂલ્ય જાળવી રાખે છે.	static int count = 0;
Global	-	ગ્લોબલ (આખા પ્રોગ્રામમાં) વ્યાપ; બધા ફંક્શનની બહાર વ્યાખ્યાયિત	int total;

ટેબલ 7.7 : સ્ટોરેજ ક્લાસીસ અને તેની અગત્યતા

void ડેટા પ્રકાર (void Data Type)

C ભાષામાં *void* ડેટા પ્રકાર મૂલ્યના અભાવ (absence of value) ને દર્શાવે છે. તેને ખાલી ડેટા ટાઈપ (empty data type) તરીકે પણ ઓળખવામાં આવે છે. અન્ય ડેટા ટાઈપ જેમ કે *int, float* અથવા *char* ડેટા સંગ્રહે છે, જ્યારે *void* સૂચવે છે કે કોઈ ડેટા પરત અપાતો નથી, સ્વીકારાતો નથી, અથવા સંગ્રહ થતો નથી. તે મુખ્યત્વે એવા ફંક્શનમાં વપરાય છે જે કોઈ મૂલ્ય પરત (return) આપતા નથી અથવા પેરામીટર્સ સ્વીકારતા નથી, તેમજ *void* પોઈન્ટરમાં પણ તેનો ઉપયોગ થાય છે. આકૃતિ 7.8માં *void* ડેટા પ્રકારો આપેલા છે.

વાક્યરચના (Syntax):

```
void main()
{
// code
}
```

ઉપયોગ પરિસ્થિતિ (Use Case)	વર્ણન	ઉદાહરણ
void Function (no return)	ફંક્શન કોઈ કાર્ય કરે છે પરંતુ કંઈ પણ મૂલ્ય પરત આપતું નથી.	void greet(){ ... }
void Parameter (no input)	ફંક્શન કોઈ ઈનપુટ પેરામીટર (argument) લેતું નથી.	void show(void){...}
void Pointer	એવો પોઈન્ટર જે કોઈપણ પ્રકારના એડ્રેસને સંગ્રહ કરી શકે છે.	void *ptr;

ટેબલ 7.8 : void ડેટા પ્રકારો

સારાંશ

આ પ્રકરણમાં, આપણે C પ્રોગ્રામિંગમાં વપરાતા વિવિધ પ્રકારના ડેટા અને તે પ્રોગ્રામમાં માહિતીનું સંચાલન કરવામાં કેવી રીતે મદદ કરે છે, તે વિશે વિગતવાર સમજ્યા. આપણે મૂળભૂત ડેટા પ્રકારો (Primary Data Types) જેમ કે int, float, double અને char થી શરૂઆત કરી, જે અનુક્રમે પૂર્ણ સંખ્યાઓ, અપૂર્ણાંક (દશાંશ) સંખ્યાઓ અને અક્ષરોનો સંગ્રહ કરે છે. આપણે મૂલ્યોની શ્રેણી (range) ને નિયંત્રિત કરવા માટે આ પ્રકારોના signed અને unsigned પ્રકારનો ઉપયોગ કેવી રીતે કરવો તે શીખ્યા. આપણે ડિરાઈવ્ડ ડેટા પ્રકારો (Derived Data Types) વિશે પણ જાણ્યું, જેમાં એરે અને પોઈન્ટરનો સમાવેશ થાય છે, જે બહુવિધ મૂલ્યો અથવા મેમરી એડ્રેસને સંભાળવા માટે હાલના પ્રકારોમાંથી બનાવવામાં આવે છે. ત્યારબાદ, આપણે યુઝર-નિર્મિત ડેટા પ્રકારો (User-Defined Data Types) જેમ કે struct અને union નો અભ્યાસ કર્યો, જે પ્રોગ્રામરોને ડેટાને વ્યવસ્થિત કરવા માટે યુઝર-નિર્મિત ફોર્મેટ બનાવવાની મંજૂરી આપે છે. આ પ્રકરણમાં સ્ટોરેજ ક્લાસિસ (Storage Classes) જેમ કે auto, static, register, અને global પણ આવરી લેવામાં આવ્યા છે, જે ચલનો વ્યાપ (scope) અને જીવનકાળ (lifetime) નક્કી કરે છે. છેલ્લે, આપણે void ડેટા પ્રકાર વિશે શીખ્યા, જેનો ઉપયોગ ફંક્શન અને પોઈન્ટરમાં ''કોઈ મૂલ્ય નહીં'' (no value) દર્શાવવા માટે થાય છે.

સ્વાધ્યાય

1. C પ્રોગ્રામિંગમાં ડેટા પ્રકારોનો હેતુ શું છે?
2. C માં ડેટા પ્રકારોની ચાર મુખ્ય શ્રેણીઓનાં નામ આપો.
3. પૂર્ણાંક (int) ડેટા પ્રકારની વ્યાખ્યા આપો અને એક ઉદાહરણ આપો.

4. float અને double ડેટા પ્રકારોને ઉદાહરણ સાથે સમજાવો.

5. C પ્રોગ્રામિંગમાં char (અક્ષર) ના ઉપયોગની સમજૂતી આપો.

6. યુઝર-નિર્મિત ડેટા પ્રકાર (User-Defined Data Type) શું છે? કોઈપણ બે પ્રકારના નામ આપો.

7. C માં ડિરાઈવ્ડ ડેટા પ્રકારો (Derived Data Types) શું છે? એક ઉદાહરણ આપો.

8. C માં void ડેટા પ્રકાર શું છે?

9. C માં ચાર સ્ટોરેજ ક્લાસ (Storage Classes) ની યાદી બનાવો અને તેમાંથી એકનું વર્ણન કરો.

10. signed int અને unsigned int વચ્ચે શું તફાવત છે?

11. સાચું કે ખોટું જણાવો.

(1) char ડેટા પ્રકાર એક સમયે એક કરતાં વધારે અક્ષરો સંગ્રહ કરી શકે છે.

(2) C પ્રોગ્રામમાં main() ફંક્શન વૈકલ્પિક છે.

(3) unsigned int ધન અને ઋણ બંને સંખ્યાઓ સંગ્રહ કરી શકે છે.

(4) C માં એરે (Arrays) એ ડિરાઈવ્ડ ડેટા પ્રકારો (Derived Data Types) નો ભાગ છે.

(5) સ્ટોરેજ ક્લાસ (Storage Class) મેમરીમાં ચલના જીવનકાળ (life) ને અસર કરે છે.

12. ખાલી જગ્યાઓ પૂરો.

(1) ____________ ડેટા પ્રકારનો ઉપયોગ પૂર્ણ સંખ્યાઓ સંગ્રહ કરવા માટે થાય છે.

(2) ____________ સ્ટોરેજ ક્લાસ ફંક્શન કોલ્સ વચ્ચે તેનું મૂલ્ય જાળવી રાખે છે.

(3) જે ફંક્શન કોઈ મૂલ્ય પાછું નથી આપતું તે ____________ ડેટા પ્રકાર સાથે ઘોષિત કરવામાં આવે છે.

(4) એરે ____________ ડેટા પ્રકારના મૂલ્યોનો સંગ્રહ કરે છે.

(5) ____________ ડેટા પ્રકારનો ઉપયોગ એક જ અક્ષર સંગ્રહ કરવા માટે થાય છે.

13. બહુવિકલ્પી પ્રશ્નો. સૌથી યોગ્ય જવાબ પસંદ કરો.

(1) નીચેનામાંથી ક્યો ડેટા પ્રકાર દશાંશ સંખ્યાઓ સંગ્રહ કરવા ઉપયોગી છે?

(a) int (b) char (c) float (d) void

(2) નીચેનામાંથી ક્યો યુઝર-નિર્મિત ડેટા પ્રકાર છે?

(a) int (b) float (c) union (d) char

(3) નીચેનામાંથી ક્યો કીવર્ડ ચલને સ્થાનિક વ્યાપ (local scope) સાથે વ્યાખ્યાયિત કરે છે?

(a) auto (b) register (c) static (d) global

(4) નીચેનામાંથી ક્યું ફોર્મેટ સ્પેસિફાયર signed int માટે વપરાય છે?

(a) %u (b) %f (c) %d (d) %c

(5) નીચેનામાંથી ક્યો ડેટા પ્રકાર "કોઈ મૂલ્ય નહિ" દર્શાવે છે?

(a) void (b) int (c) char (d) float

(6) નીચેનામાંથી ક્યો ડેટા પ્રકાર એક અક્ષર સંગ્રહ કરવા ઉપયોગી છે?

(a) float (b) char (c) int (d) double

(7) નીચેનામાંથી ક્યો ડિરાઈવ્ડ ડેટા પ્રકાર મેમરી એડ્રેસ સંગ્રહ કરવા વપરાય છે?

(a) Array (b) Pointer (c) Struct (d) Union

(8) C માં float ડેટા પ્રકારની સાઈઝ કેટલી હોય છે?

(a) 2 bytes (b) 4 bytes (c) 8 bytes (d) 10 bytes

(9) CPU માં ઝડપી ઉપયોગ માટે ક્યો સ્ટોરેજ ક્લાસ ઉપયોગી છે?

(a) static (b) register (c) global (d) auto

(10) નીચેનામાંથી ક્યો યુઝર-નિર્મિત ડેટા પ્રકાર એકથી વધારે અલગ-અલગ પ્રકારના ચલો ધરાવી શકે?

(a) enum (b) struct (c) int (d) float

પ્રાયોગિક સ્વાધ્યાય

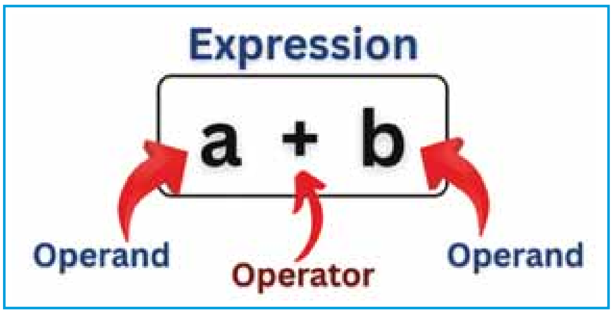
1. int અને unsigned int ના ઉપયોગનું નિદર્શન કરતો C પ્રોગ્રામ લખો.
2. બે float સંખ્યાઓનો ગુણાકાર કરતો C પ્રોગ્રામ લખો.
3. *printf()* ફંક્શનનો ઉપયોગ કરીને વિદ્યાર્થીની વિગતો દર્શાવતો C પ્રોગ્રામ લખો.
4. અરેનો ઉપયોગ કરીને 5 સંખ્યાઓનો સંગ્રહ અને પ્રિન્ટ કરતો પ્રોગ્રામ લખો.
5. તમારી શાળાનું સંપૂર્ણ સરનામું પ્રિન્ટ કરતો C પ્રોગ્રામ લખો.

ઓપરેટર અને પદાવલીઓ

પરિચય

પાછલા પ્રકરણમાં, આપણે C પ્રોગ્રામિંગના વિવિધ ડેટા પ્રકારો અને સ્ટોરેજ ક્લાસ તેમજ તેમના મહત્ત્વનો અભ્યાસ કર્યો. આ પ્રકરણમાં, C પ્રોગ્રામિંગમાં વપરાતા વિવિધ પ્રકારના ઓપરેટરો અંગે ચર્ચા કરવામાં આવી છે. પ્રોગ્રામર દ્વારા પદાવલીઓ (Expressions) બનાવવા અને પ્રોગ્રામના તાર્કિક પ્રવાહને નિયંત્રિત કરવા માટે ઓપરેટરોનો ઉપયોગ થાય છે. ઓપરેટર C પ્રોગ્રામિંગ ભાષાના મુખ્ય ઘટકો છે, જે ચલ અને મૂલ્યો પર વિવિધ પ્રકારની ક્રિયાઓ કરવાની ક્ષમતા આપે છે. આ પ્રકરણમાં ઓપરેટરોને વિવિધ પ્રકારોમાં વર્ગીકૃત કરવામાં આવ્યા છે જેમ કે - ગણિતીય (Arithmatic), સંબંધસૂચક (Relational), તાર્કિક (Logical), અસાઈનમેન્ટ (Assignment), બિટવાઈઝ (Bitwise) વગેરે. પ્રકરણમાં તેમની વાક્યરચના, પ્રાથમિકતા (precedence) અને સંબંધતા (associativity) ના નિયમોનું વિગતવાર વર્ણન છે. વ્યવહારિક ઉદાહરણો દ્વારા બતાવવામાં આવ્યું છે કે પદાવલીઓ કેવી રીતે રચાય છે, મૂલ્યાંકન થાય છે અને પ્રોગ્રામના તર્ક તથા ડેટા સંચાલન માટે કેવી રીતે ઉપયોગમાં લેવાય છે. ઓપરેટર અને પદાવલીઓની સમજ કાર્યક્ષમ અને તર્કસંગત C પ્રોગ્રામ લખવા માટે અત્યંત જરૂરી છે. તે ગણતરીઓ, નિર્ણય લેવાની પ્રક્રિયા (decision-making) અને લૂપમાં વ્યાપકપણે ઉપયોગમાં લેવાય છે. આ પ્રકરણ ભવિષ્યના વધુ અદ્યતન પ્રોગ્રામિંગ સિદ્ધાંતો સમજવા માટે મજબૂત પાયો પૂરો પાડે છે.

પ્રોગ્રામિંગમાં ઓપરેટરો (Operators) અને પદાવલીઓ (Expressions) શું છે?



આકૃતિ 8.1 : ઓપરેટર અને પદાવલી

પ્રોગ્રામિંગમાં ઓપરેટર (Operators) અને પદાવલીઓ એવા સાધનો છે જે ડેટા પર વિવિધ પ્રકારની ક્રિયાઓ કરવા માટે વપરાય છે. ઓપરેટર એ એક પ્રતીક છે જે પ્રોગ્રામને એક અથવા વધુ મૂલ્યો પર ચોક્કસ ક્રિયા કરવા સૂચવે છે. પદાવલી (Expression) એ મૂલ્યો, ચલો (variables) અને ઓપરેટરોનું સંયોજન છે જે કોઈ પરિણામ આપે છે. એક પદાવલીમાં ઓછામાં ઓછો એક ઓપરેન્ડ (operand) અને શૂન્ય અથવા વધુ ઓપરેટર હોય છે.

આકૃતિ 8.1 પદાવલિનું ઉદાહરણ દર્શાવે છે.

નીચે પદાવલી (expression) નું એક સરળ ઉદાહરણ આપેલ છે: "5 + 3", જ્યાં -

- 5 અને 3 મૂલ્યો છે. તેમને ઓપરેન્ડ્સ કહેવામાં આવે છે.
- + એ ઓપરેટર છે, જેના કારણે પદાવલીનું મૂલ્યાંકન થઈને પરિણામ 8 મળે છે.

આપણે પદાવલીઓમાં ચલોનો પણ ઉપયોગ કરી શકીએ છીએ. નીચે ચલો વાપરતી પદાવલીનું ઉદાહરણ છે:

A = 10
B = 20
Result = A + B

અહીં, A અને B ચલના નામો છે; A + B એક પદાવલી છે, જે પરિણામ 30 આપે છે.

હવે, આપણે C પ્રોગ્રામિંગમાં મળતા વિવિધ પ્રકારના ઓપરેટરો વિશે જાણકારી મેળવીશું. મુખ્ય પ્રકારો આ પ્રમાણે છે: ગણિતીય (Arthmatic), સંબંધસૂચક (Relational), તાર્કિક (Logical), બિટવાઈઝ (Bitwise),

અસાઈનમેન્ટ (Assignment), વધારો અને ઘટાડો (Increment & Decrement), અને વિશેષ (Special) ઓપરેટરો. ચાલો ઉદાહરણો સાથે મહત્વના ઓપરેટરોના ઉપયોગને સમજીએ.

ગણિતીય ઓપરેટરો (Arithmatic Operators)

ગણિતીય ઓપરેટરો C પ્રોગ્રામિંગ ભાષામાં ગણિતીય ગણતરીઓ કરવા માટે વપરાય છે. તેઓ આપણને સંખ્યાત્મક મૂલ્યો પર ક્રિયાઓ કરવાની મંજૂરી આપે છે જેથી વિવિધ પ્રકારના પરિણામો પ્રાપ્ત થઈ શકે. ગણિતીય ઓપરેટરોનો ઉપયોગ કરીને આપણે સરવાળા, બાદબાકી, ગુણાકાર, ભાગાકાર, ભાગશેષ જેવી ક્રિયાઓ કરી શકીએ છીએ. આ ઓપરેટરો સરળ ગણનાઓથી લઈને જટિલ અલ્ગોરિધમ સુધીના કાર્ય માટે અત્યંત મહત્વપૂર્ણ છે. આકૃતિ 8.2 ગણિતીય ઓપરેટરની યાદી દર્શાવે છે.

આ ઓપરેટરો બાઈનરી પ્રકારના હોય છે, એટલે કે તેઓ બે ઓપરેન્ડ પર કાર્ય કરે છે.

આકૃતિ 8.2 : ગણિતીય ઓપરેટરો

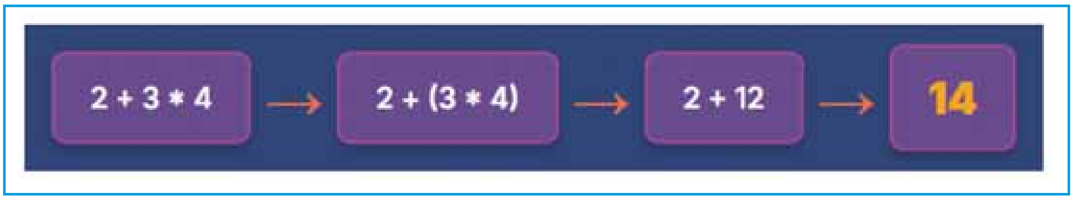
ગણિતીય ઓપરેટરના ઉદાહરણો

નીચેનો C પ્રોગ્રામ બે પૂર્ણાંક પ્રકારના ચલ a અને b પર મૂળભૂત ગણિતીય ઓપરેશન (+, -, , /, %) દર્શાવે છે.

```
/* Program to illustrate use of Arithmatic Operators */
#include <stdio.h>
int main() {
    int a=10, b=3;

    printf("Result of Arithmatic Operations on a and b:\n");

    printf("Addition       : a + b = %d \n", a + b);
    printf("Subtraction    : a - b = %d \n", a - b);
    printf("Multiplication : a * b = %d \n", a * b);
    printf("Division       : a / b = %d \n", a / b);
    printf("Modulus        : a %% b = %d\n", a % b);

    return 0;
}
Result:
Result of Arithmatic Operations on a and b:
Addition       : a + b = 13
Subtraction    : a - b = 7
Multiplication : a * b = 30
Division       : a / b = 3
Modulus        : a % b = 1
```

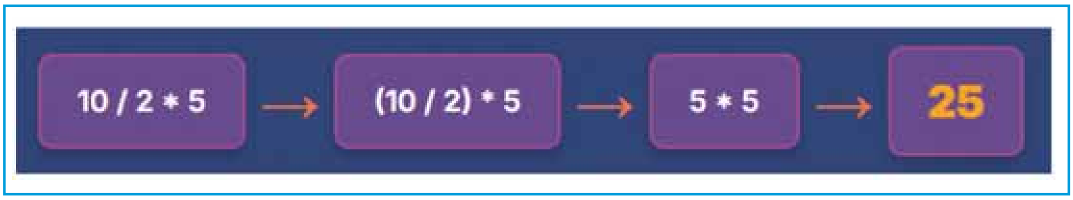
ગણિતીય ઓપરેટરોના મહત્વપૂર્ણ લક્ષણો

- **પૂર્ણાંક ભાગફળમાં છટણી (Integer Division Truncation) :** જ્યારે ભાગાકાર ઓપરેટર (/) ના બંને ઓપરેન્ડ્સ પૂર્ણાંક હોય, ત્યારે પરિણામ પણ પૂર્ણાંક આવે છે અને કોઈ પણ ભાગાકાર ભાગ (fractional part) દૂર કરવામાં આવે છે. ઉદાહરણ તરીકે, 10 / 3 નું પરિણામ 3 આવે છે, 3.33 નહીં. અપૂર્ણાંક પરિણામ મેળવવા માટે ઓપરેન્ડ્સમાં ઓછામાં ઓછો એક અપૂર્ણાંક પ્રકારનો હોવો જોઈએ, (જેમ કે 10.0 / 3 અથવા 10 / 3.0).
- **મોડ્યુલસ ઓપરેટર (મોડ્યુલસ - %) :** મોડ્યુલસ (ભાગશેષ) ઓપરેટરનો ઉપયોગ માત્ર પૂર્ણાંક ઓપરેન્ડ સાથે જ થઈ શકે છે. તેનો ઉપયોગ અપૂર્ણાંક પ્રકારો (*float* અથવા *double*) સાથે થતો નથી.
- **ઓપરેટરની પ્રાથમિકતા અને સંબંધતા (Operator Precedence and Associativity) :**
  - o પ્રાથમિકતા (Precedence) : ગુણાકાર (*), ભાગાકાર (/), અને ભાગશેષ (%) ઓપરેટરો સરવાળા (+) અને બાદબાકી (–) કરતાં વધુ પ્રાથમિકતા ધરાવે છે. એનો મતલબ છે કે પદાવલીમાં તેઓનું મૂલ્યાંકન પહેલા કરવામાં આવે છે. ઉદાહરણ તરીકે, 2 + 3 * 4 નું મૂલ્યાંકન 14 આવે છે, કારણ કે 3 * 4 પહેલા ગણવામાં આવે છે. (જુઓ આકૃતિ 8.3)



આકૃતિ 8.3 : ઓપરેટર પ્રાથમિકતા

  - o સંબંધતા (Associativity) : જ્યારે પદાવલીમાં સમાન પ્રાથમિકતા ધરાવતા ઓપરેટરો એક સાથે આવે છે, જેમ કે 10 / 2 * 5, ત્યારે તેમનું મૂલ્યાંકન ડાબેથી જમણે (left to right) કરવામાં આવે છે. ઉદાહરણ તરીકે, 10 / 2 * 5 નું મૂલ્યાંકન થાય છે: (10 / 2) * 5 = 5 * 5 = 25. તેને આકૃતિ 8.4 માં દર્શાવવામાં આવ્યું છે.



આકૃતિ 8.4 : ઓપરેટર સંબંધતા

- **પ્રકાર રૂપાંતર (Type Conversion) :** જ્યારે અલગ-અલગ ડેટા પ્રકારના ઓપરેન્ડો ગણિતીય ઓપરેટરો સાથે વપરાય છે, ત્યારે C સ્વયંસંચાલિત પ્રકાર રૂપાંતર (implicit type conversion / type promotion) કરે છે. સામાન્ય રીતે, "નાનો" ડેટા ટાઈપ "મોટા" ડેટા ટાઈપમાં પ્રમોટ કરવામાં આવે છે જેથી ડેટા નુકસાન ટાળવું શક્ય બને. ઉદાહરણ તરીકે, જો *int* અને *float* એક જ પદાવલીમાં સાથે વપરાય છે, તો *int* ને પહેલા *float* માં રૂપાંતરિત કરવામાં આવે છે. જેમ કે,

  float result = 2 + 5.5;

  આ સ્થિતિમાં, 2 ને પ્રથમ 2.0 માં ફેરવવામાં આવે છે, પછી 2.0 + 5.5 = 7.5 ની ગણના થાય છે. આ રીતે, **int result = 2 + 5.5;** માટે, પહેલા 2 ને 2.0 માં રૂપાંતરિત કરવામાં આવે છે, પછી 2.0 + 5.5 = 7.5 થાય છે, અને અંતે 7.5 ને 7 કરવામાં આવે છે કારણ કે result ચલનો પ્રકાર *int* છે.

- **શૂન્યથી ભાગાકાર (Division by Zero) :** C માં શૂન્યથી ભાગાકાર અવ્યખાયાયિત ક્રિયા છે. જો તમે 10 / 0 અથવા 10.0 / 0.0 કરવા પ્રયત્ન કરો તો તે રનટાઈમ ત્રુટી (error) અથવા અનિર્ધારિત પરિણામ આપે છે, જેને લીધે ઘણીવાર પ્રોગ્રામ તૂટી (crash) પડે છે. તેથી આપણા પ્રોગ્રામમાં શૂન્ય વડે ભાગાકાર (division by zero) જેવી કોઈપણ શક્યતાઓને નિવારવી અત્યંત આવશ્યક છે.

સંબંધસૂચક ઓપરેટરો (Relational Operators)

C પ્રોગ્રામિંગમાં સંબંધસૂચક ઓપરેટરોને તુલનાત્મક ઓપરેટરો (Comparison Operators) તરીકે પણ ઓળખવામાં આવે છે. તે પ્રોગ્રામના પ્રવાહને નિયંત્રિત કરવા માટે અગત્યના છે, કારણ કે તે ચલ, અચલ અને પદાવલીઓ વચ્ચે તુલના કરીને નિર્ણય લેવા સક્ષમ બનાવે છે. સંબંધસૂચક ઓપરેટરોનો ઉપયોગ બે મૂલ્યોની તુલના કરવા માટે થાય છે. સંબંધસૂચક પદાવલીનું પરિણામ 1 (સાચું) અથવા 0 (ખોટું) સ્વરૂપે મળે છે. આ ઓપરેટરો સામાન્ય રીતે નિર્ણય વિધાનો (*if, else-if*) અને લૂપ (*for, while, do-while*) માં પ્રોગ્રામના પ્રવાહને નિયંત્રિત કરવા માટે સૌથી વધુ ઉપયોગમાં લેવાય છે. ટેબલ 8.1માં C પ્રોગ્રામિંગમાં વપરાતા સંબંધસુચક ઓપરેટરો અને તેમનો ઉપયોગ દર્શાવવામાં આવ્યો છે.

સંબંધસૂચક ઓપરેટરનો ઉપયોગ ઉદાહરણ સાથે સમજવા માટે, ચાલો ધારીએ કે આપણા પાસે બે પૂર્ણાંક ચલો a અને b છે. આ બે ચલો પર વિવિધ સંબંધસૂચક ઓપરેટરોના પરિણામો નીચેની કિંમતો પ્રમાણે ટેબલ 8.1માં દર્શાવવામાં આવ્યા છે.

```
int a = 10;
int b = 5;
```

ઓપરેટર	નામ	વર્ણન	C માં ઉદાહરણ (a = 10, b = 5)	પરિણામ
==	Equal to	બે ઓપરેન્ડનું મૂલ્ય સરખું છે તે ચકાસવા	a == b	0 (false)
!=	Not equal to	બે ઓપરેન્ડનું મૂલ્ય સરખું નથી તે ચકાસવા	a != b	1 (true)
>	Greater than	ડાબી બાજુના ઓપરેન્ડનું મૂલ્ય જમણી બાજુના ઓપરેન્ડ કરતા વધારે છે તે ચકાસવા	a > b	1 (true)
<	Less than	ડાબી બાજુના ઓપરેન્ડનું મૂલ્ય જમણી બાજુના ઓપરેન્ડ કરતા ઓછું છે તે ચકાસવા	a < b	0 (false)
>=	Greater than or equal to	ડાબી બાજુના ઓપરેન્ડનું મૂલ્ય જમણી બાજુના ઓપરેન્ડ કરતા વધારે કે સરખું છે તે ચકાસવા	a >= 10	1 (true)
<=	Less than or equal to	ડાબી બાજુના ઓપરેન્ડનું મૂલ્ય જમણી બાજુના ઓપરેન્ડ કરતા ઓછું કે સરખું છે તે ચકાસવા	b <= 5	1 (true)

ટેબલ 8.1 : સંબંધસૂચક ઓપરેટરો

સંબંધસૂચક ઓપરેટરનું ઉદાહરણ

નીચે આપેલો C પ્રોગ્રામ બધા સંબંધસૂચક ઓપરેટરનો ઉપયોગ દર્શાવે છે અને તેમના પરિણામો દર્શાવે છે.

```
/* Program to illustrate use of Relational Operators */

#include <stdio.h>
int main() {
  int a = 10;
  int b = 5;

  printf("Using a = %d and b = %d \n", a, b);
  printf("---------------------------------\n");

  printf("Is a == b? Result: %d \n", a == b);  /* 10 == 5 is false */
  printf("Is a != b? Result: %d \n", a != b);  /* 10 != 5 is true */
  printf("Is a > b?  Result: %d \n", a > b);   /* 10 > 5 is true */
  printf("Is a < b?  Result: %d \n", a < b);   /* 10 < 5 is false */
  printf("Is a >= b? Result: %d\n",  a >= b);  /* 10 >= 5 is true */
  printf("Is b <= a? Result: %d\n",  b <= a);  /* 5 <= 10 is true */

   return 0;
}
Result:
Using a = 10 and b = 5
---------------------------------
Is a == b?  Result: 0
Is a != b?  Result: 1
Is a > b?   Result: 1
Is a < b?   Result: 0
Is a >= b?  Result: 1
Is b <= a?  Result: 1
```

યાદ રાખવાના મુખ્ય મુદ્દાઓ

- **== વિરુદ્ધ (verses) = :** નવા શીખનારાઓ પ્રોગ્રામમાં એક ખૂબ જ સામાન્ય ભૂલ એ કરે છે કે શરતમાં સમાનતા ઓપરેટર (==) ને બદલે અસાઇનમેન્ટ ઓપરેટર (=) નો ઉપયોગ કરવો. if (x = 5) એ x ને 5 કિંમત આપે છે ને તે પદાવલી પોતે 5 તરીકે મૂલ્યાંકિત થાય છે (જે શૂન્ય નથી, તેથી સાચું (true) ગણાય છે), જ્યારે *if* (x == 5) એ x ની કિંમત પહેલેથી 5 છે કે કેમ તે તપાસે છે.
- **પરત મૂલ્ય (Return Value) :** સંબંધસૂચક ઓપરેશનનું પરિણામ હંમેશા *int* પ્રકારનું હોય છે: 1 એટલે સાચું (true) અને 0 એટલે ખોટું (false).
- **ડેટા પ્રકારો (Data Types) :** આ ઓપરેટરોનો ઉપયોગ ફક્ત પૂર્ણાંકો જ નહીં, પણ અપૂર્ણાંક સંખ્યાઓ અને અક્ષરો *(char)* ની પણ સરખામણી કરવા માટે થઈ શકે છે. ધ્યાન રાખો કે જ્યારે અપૂર્ણાંક કિંમતોની સમાનતા (==) માટે સરખામણી કરવામાં આવે છે, ત્યારે અપૂર્ણાંક નંબરોની રાઉન્ડિંગ ભૂલો (rounding errors) ને કારણે અણધાર્યા પરિણામો આવી શકે છે. તેથી, ફ્લોટિંગ પોઇન્ટ કિંમતો સાથે (==) નો ઉપયોગ કરતી વખતે સાવચેત રહેવું.

તાર્કિક ઓપરેટરો (Logical Operators)

વિવિધ સમસ્યા-ઉકેલ પરિસ્થિતિઓમાં, આપણને ઘણીવાર આપણા પ્રોગ્રામમાં બહુવિધ શરતોને જોડવાની જરૂર પડે છે. C પ્રોગ્રામિંગમાં તાર્કિક ઓપરેટરોનો ઉપયોગ કરીને, આપણે ઇચ્છિત પરિણામો પ્રાપ્ત કરવા માટે આ શરતોને અસરકારક રીતે ભેગી કરી શકીએ છીએ.

દાખલા તરીકે, ચાલો એક એવી પરિસ્થિતિ જોઈએ જ્યાં આપણે વિદ્યાર્થીના ગુણ અને હાજરીના આધારે તેણે પરીક્ષા પાસ કરી છે કે નહીં તે નક્કી કરવાની જરૂર છે. પાસ થવા માટેના માપદંડો આ પ્રમાણે છે:

- માર્ક્સ 40 કે તેથી વધુ હોવા જોઈએ.
- હાજરી 75% કે તેથી વધુ હોવી જોઈએ.

આ કિસ્સામાં, આપણે આ બે શરતોને જોડવા અને આપણી સમસ્યાનું નિરાકરણ લાવવા માટે તાર્કિક AND (&&) ઓપરેટરનો ઉપયોગ કરી શકીએ છીએ.

તાર્કિક ઓપરેટરો પ્રોગ્રામમાં નિર્ણય લેવા માટે ખૂબ જ ઉપયોગી છે, જે આપણને વધુ જટિલ શરતોને નિયંત્રિત કરવામાં સક્ષમ બનાવે છે. તે બુલિયન પદાવલી (boolean expressions) માં પણ મદદ કરે છે, જેનું મૂલ્યાંકન સાચા (true) અથવા ખોટા (false) માં થાય છે. ધ્યાન રાખો કે C પ્રોગ્રામિંગમાં, કોઈપણ બિન-શૂન્ય (non-zero) અને બિન-નલ (non-null) મૂલ્યોને સાચા તરીકે ગણવામાં આવે છે, જ્યારે શૂન્ય (zero) ને ખોટા તરીકે ગણવામાં આવે છે. ટેબલ 8.2 તાર્કિક ઓપરેટરો અને C પ્રોગ્રામિંગમાં તેના ઉપયોગનો સારાંશ આપે છે.

ઓપરેટર	નામ	ઉપયોગ	ઉદાહરણ (ધારોકે A=1, B=0)	પરિણામ
&&	Logical AND	બંને ઓપરેન્ડ સાચાં હશે તો સાચું (true) આપશે.	A && B (1 && 0)	0
||	Logical OR	કોઈ એક ઓપરેન્ડ સાચો હશે તો સાચું (true) આપશે.	A || B (1 || 0)	1
!	Logical NOT	ઓપરેન્ડની સ્થિતિ બદલશે. (true to false and vice versa)	!A (!1)	0

ટેબલ 8.2 : તાર્કિક ઓપરેટરો

તાર્કિક AND અને તાર્કિક OR ઓપરેટરોનું મૂલ્યાંકન નીચે આપેલ ટ્રૂથ ટેબલ (Truth Table) મુજબ થાય છે, જે ટેબલ 8.3 માં દર્શાવ્યું છે. ટેબલ 8.4 માં તાર્કિક NOT ઓપરેટરનું મૂલ્યાંકન આપેલ છે.

A (true/false)	B (true/false)	A && B (Logical AND)	A || B (Logical OR)
0 (false)	0 (false)	0	0
0 (false)	1 (true)	0	1
1 (true)	0 (false)	0	1
1 (true)	1 (true)	1	1

ટેબલ 8.3 : તાર્કિક AND અને OR નું ટ્રૂથ ટેબલ

A (true/false)	!A
0	1
1	0

ટેબલ 8.4 : તાર્કિક NOT (!) નું ટ્રૂથ ટેબલ

તાર્કિક ઓપરેટરનું ઉદાહરણ

નીચેનો C પ્રોગ્રામ તાર્કિક AND ઓપરેટરનું ઉદાહરણ પ્રસ્તુત કરે છે.

```
/* C program to illustrate use of AND operator. */

#include <stdio.h>
int main() {
    int marks = 50, attendance = 80;

    if (marks >= 40 && attendance >= 75)
        printf("Student has passed the exam.");
    else
        printf("Student has not passed the exam.");
    return 0;
}
Result:
Student has passed the exam.
```

આ પ્રોગ્રામમાં બંને પદાવલીઓ (marks >= 40) અને (attendance >= 75) નું મૂલ્યાંકન સાચું (true) થાય છે, તેથી તે આઉટપુટ તરીકે "Student has passed the exam." પ્રિન્ટ કરશે.

નીચેનો C પ્રોગ્રામ તાર્કિક OR ઓપરેટરનું ઉદાહરણ પ્રસ્તુત કરે છે.

```
/* C program to illustrate use of OR operator. */
#include <stdio.h>
int main() {
    int a = -5, b = 10;
    if (a > 0 || b > 0)
        printf("At least one is positive.");
    return 0;
}
Result:
At least one is positive.
```

નીચેનો C પ્રોગ્રામ તાર્કિક NOT ઓપરેટરનું ઉદાહરણ પ્રસ્તુત કરે છે.

```
/* C program to illustrate use of NOT operator. */
#include <stdio.h>
int main() {
    int isAdmin = 0;
    if (!isAdmin)
        printf("User is not an administrator.");
    else
       printf("User is an administrator.");
    return 0;
}
Result:
User is not an administrator.
```

અસાઈનમેન્ટ ઓપરેટરો (Assignment Operators)

C પ્રોગ્રામિંગમાં અસાઈનમેન્ટ ઓપરેટરોનો ઉપયોગ ચલને મૂલ્ય આપવા માટે થાય છે. સૌથી સામાન્ય અસાઈનમેન્ટ ઓપરેટર છે = (સમાન ચિહ્ન).

મૂળ (Basic) અસાઈનમેન્ટ ઓપરેટર (=)

અસાઈનમેન્ટ ઓપરેટર = નો ઉપયોગ ડાબી બાજુના ચલને જમણી બાજુનું મૂલ્ય આપવા માટે થાય છે. ઉદાહરણ:

```
int a;
a = 10;        /* This statement assigns 10 to variable a */
```

શૉર્ટ-હેન્ડ (Short-hand) અસાઈનમેન્ટ ઓપરેટરો

શૉર્ટ-હેન્ડ અસાઈનમેન્ટ ઑપરેટર એ ગણિતીય ઓપરેટરને અસાઈનમેન્ટ ઓપરેટર સાથે જોડે છે. આનો ઉપયોગ કોડને વધુ સંક્ષિપ્ત બનાવવા માટે થાય છે. આ પ્રકારના ઓપરેટર સામાન્ય રીતે લૂપ અને ગણિતીય ગણતરીઓમાં વપરાય છે.

નીચે આપેલ વિધાન જુઓ, જેમાં += શૉર્ટ-હેન્ડ ઓપરેટરનો ઉપયોગ થયો છે, જે + ઓપરેટરને = ઓપરેટર સાથે જોડે છે:

```
a += 5;
```

તે નીચેના બરાબર છે:

```
a = a + 5;
```

+= ઓપરેટર જમણી બાજુના ઓપરેન્ડનું મૂલ્ય ડાબી બાજુના ઓપરેન્ડમાં ઉમેરે છે અને પરિણામ ફરીથી ડાબી બાજુના ઓપરેન્ડને આપે છે. ટેબલ 8.5માં વિવિધ શૉર્ટ-હેન્ડ અસાઈનમેન્ટ ઓપરેટરો, તેમનો ઉપયોગ અને ઉદાહરણો દર્શાવવામાં આવ્યા છે.

ઓપરેટર	અર્થ	ઉદાહરણ	ના બરાબર
+=	Add and assign	a += 5;	a = a + 5;
-=	Subtract and assign	a -= 3;	a = a - 3;
*=	Multiply and assign	a *= 2;	a = a * 2;
/=	Divide and assign	a /= 2;	a = a / 2;
%=	Modulus and assign	a %= 3;	a = a % 3;

ટેબલ 8.5 : શૉર્ટ-હેન્ડ અસાઈનમેન્ટ ઓપરેટરો

શૉર્ટ-હેન્ડ અસાઈનમેન્ટ ઓપરેટરનું ઉદાહરણ

```
/*Program to illustrate use of Short-hand assignment operators */
#include <stdio.h>
int main() {
    int a = 10;
    a += 5;     /* a = 10 + 5 = 15  */
    a *= 2;     /* a = 15 * 2 = 30  */
    printf("Final value of a: %d", a);
    return 0;
}
Result:
Final value of a: 30
```

બિટવાઈઝ ઓપરેટરો (Bitwise Operators)

C પ્રોગ્રામિંગમાં બિટવાઈઝ ઓપરેટરોનો ઉપયોગ પૂર્ણાંક પર બિટ સ્તરે (bit level) કામગીરી કરવા માટે થાય છે. આ ઓપરેટરો પૂર્ણાંકને બિટ્સની શ્રેણી (0 અને 1) તરીકે ગણે છે અને દરેક બિટ પર અલગથી ક્રિયા કરે છે. આ રીત કેટલીક કામગીરી માટે ખૂબ જ કાર્યક્ષમ સાબિત થાય છે, ખાસ કરીને જ્યારે ફ્લેગ્સ (flags) સાથે કામ કરવું હોય, હાર્ડવેર રજિસ્ટર (hardware registers) સંચાલિત કરવા હોય અથવા અલ્ગોરિધમ્સને કાર્યક્ષમ કરવા હોય. ટેબલ 8.6માં બિટવાઈઝ ઓપરેટરો અને તેમનો ઉપયોગ દર્શાવવામાં આવ્યો છે.

ઓપરેટર	ઉપયોગ
&	બિટવાઈઝ AND
|	બિટવાઈઝ OR
^	બિટવાઈઝ Exclusive OR
~	બિટવાઈઝ NOT (આ એક યુનરી ઓપરેટર છે)
<<	ડાબે ખસેડવું (Left shift - as per bits specified)
>>	જમણે ખસેડવું (Right shift - as per bits specified)

ટેબલ 8.6 : બિટવાઈઝ ઓપરેટરો

બિટવાઈઝ AND, બિટવાઈઝ OR, બિટવાઈઝ Exclusive OR, ડાબે ખસેડવું (Left Shift) અને જમણે ખસેડવું (Right Shift) બાઈનરી ઓપરેટરો છે, કારણ કે તેઓની ક્રિયા માટે બે ઓપરેન્ડની જરૂર હોય છે. બિટવાઈઝ NOT ઓપરેટર એક યુનરી (unary) ઓપરેટર છે, કારણ કે તેની ક્રિયા માટે એક જ ઓપરેન્ડ પૂરતો છે. આ કોર્સમાં બિટવાઈઝ ઓપરેટરો વિશે વધુ વિગતો આવરી લેવાઈ નથી. તમે તેમને તમારા ઉચ્ચસ્તરીય અભ્યાસમાં વધુ વિગતવાર શીખશો.

વધારા (Increment) અને ઘટાડા (Decrement) સૂચક ઓપરેટરો

C પ્રોગ્રામિંગમાં વધારા-સૂચક (Increment) ઓપરેટરનો ઉપયોગ ચલનું મૂલ્ય એકથી વધારવા માટે થાય છે અને ઘટાડા-સૂચક (Decrement) ઓપરેટરનો ઉપયોગ ચલનું મૂલ્ય એકથી ઘટાડવા માટે થાય છે. આ ઓપરેટરો લૂપોને નિયંત્રિત કરવા, કાઉન્ટર ચલોને મેનેજ કરવા, અને પુનરાવર્તન કામ સંબંધી કોડને સરળ બનાવવા માટે ઉપયોગી છે.

વધારા-સૂચક (++) અને ઘટાડા-સૂચક (--) ઓપરેટરોને યુનરી (unary) ઓપરેટર કહેવામાં આવે છે, કારણ કે તેઓ એક જ ઓપરેન્ડનું મૂલ્ય બદલતા હોય છે.

- **વધારા-સૂચક ઓપરેટર (++) :** આ ઓપરેટર ઓપરેન્ડનું મૂલ્ય એકથી વધારવાનું કામ કરે છે. તેને ચલની આગળ (prefix જેમ કે, ++x) અને ચલની પાછળ (postfix જેમ કે, x++) બંને રીતે વાપરી શકાય છે.
- **ઘટાડા-સૂચક ઓપરેટર (--) :** આ ઓપરેટર ઓપરેન્ડનું મૂલ્ય એકથી ઘટાડે છે. વધારા-સૂચક ઓપરેટરની જેમ જ, તેને ચલની આગળ (prefix જેમ કે, --x) અને ચલની પાછળ (postfix જેમ કે, x--) બંને રીતે વાપરી શકાય છે.

નોંધ : prefix વધારા-સૂચક ઓપરેટર પહેલા ચલની કિંમતમાં 1નો વધારો કરે છે અને પછી ઉપયોગમાં લે છે જ્યારે postfix વધારા-સૂચક ઓપરેટર પહેલા ઉપયોગમાં લે છે અને પછી ચલની કિંમતમાં 1નો વધારો કરે છે. આજ નિયમ ઘટાડા-સૂચક ઓપરેટરને લાગુ પડે છે.

વધારા અને ઘટાડા સૂચક ઓપરેટરોનું ઉદાહરણ

```
/* Program to demonstrate use of the Increment and Decrement
operators.*/
#include <stdio.h>
int main() {
    int a = 5;
    int b = 10;
    /* Using ++ operator */

    a++;    /* same as: a = a + 1  */
    printf("Value of a after increment: %d\n", a);

    /* Using -- operator */

    b--;    /* same as: b = b - 1  */
    printf("Value of b after decrement: %d", b);

    return 0;
}

Result:

Value of a after increment: 6
Value of b after decrement: 9
```

ટર્નરી ઓપરેટર (Ternary Operator)

ટર્નરી ઓપરેટર C પ્રોગ્રામિંગમાં એક અનોખો ઓપરેટર છે, જે ત્રણ ઓપરેન્ડ ધરાવે છે. તે શરતી (conditional) ક્રિયાઓ કરવા માટેની એક સંક્ષિપ્ત રીત છે, જેનો ઉપયોગ ઘણીવાર *if-else* વિધાનના ટુંકારૂપ તરીકે થાય છે.

ટર્નરી ઓપરેટરની વાક્યરચના

```
condition ? expression1 : expression2;
```

ટર્નરી ઓપરેટર સૌપ્રથમ આપેલી શરતનું (condition) મૂલ્યાંકન કરે છે; જો શરત સાચી (true) હોય, તો પછી expression1 નો અમલ કરે છે, નહીં તો expression2 નો અમલ કરે છે. આ ઓપરેટર શરતના મૂલ્યાંકનના આધારે expression1 અથવા expression2 માંથી કોઈ એકનું પરિણામ પાછું આપે છે. તેનો ઉપયોગ સામાન્ય રીતે એક જ લીટીમાં સરળ શરતી અસાઈનમેન્ટ (conditional assignments) અથવા નિર્ણયો માટે થાય છે, જે કોડને વધુ સંક્ષિપ્ત અને વાંચન માટે સરળ બનાવે છે.

ટર્નરી ઓપરેટરનું ઉદાહરણ

```
/* Program to illustrate use of ternary operator */
#include <stdio.h>
int main() {
    int a = 10;
    int b = 5;
    int max;
```

```
        max = (a > b) ? a : b;
        printf("The maximum of %d and %d is: %d.", a, b, max);
        return 0;
    }
    Result:
    The maximum of 10 and 5 is: 10.
```

આ ઉદાહરણમાં, ટર્નરી ઓપરેટર તપાસે છે કે શું a એ b કરતાં મોટો છે. આ શરત સાચી હોવાથી, max ને a ની કિંમત અસાઈન કરવામાં આવે છે.

યુનરી, બાઈનરી અને ટર્નરી ઓપરેટરો (Unary, Binary and Ternary Operators)

ઓપરેટરને તે જેટલા ઓપરેન્ડ પર પ્રક્રિયા કરે છે તેના આધારે યુનરી, બાઈનરી અને ટર્નરી તરીકે વર્ગીકૃત કરી શકાય છે.

- **યુનરી ઓપરેટર (Unary Operator) :** C ભાષામાં, યુનરી ઓપરેટર એવા છે જે એક જ ઓપરેન્ડ પર કાર્ય કરે છે. તેનો ઉપયોગ સામાન્ય રીતે મૂલ્ય વધારવા (incrementing), ઘટાડવા (decrementing) અથવા નકારાત્મક બનાવવા (negating) જેવી મૂળભૂત કામગીરી માટે થાય છે. વધારા-સૂચક (++), ઘટાડા-સૂચક (--) અને NOT (!) એ યુનરી ઓપરેટરો છે.
- **બાઈનરી ઓપરેટર (Binary Operator) :** બાઈનરી ઓપરેટરને બે ઓપરેન્ડની જરૂર હોય છે. બાઈનરી ઓપરેટરોના ઉદાહરણોમાં ગણિતીય, સંબધસૂચક અને તાર્કિક ઓપરેટરોનો સમાવેશ થાય છે.
- **ટર્નરી ઓપરેટર (Ternary Operator) :** ટર્નરી ઓપરેટર એ C માં એક વિશિષ્ટ ઓપરેટર છે જે ત્રણ ઓપરેન્ડ ધરાવે છે. તે શરતી કામગીરીને અમલમાં મૂકવાની એક સંક્ષિપ્ત રીત પૂરી પાડે છે, જે ઘણીવાર *if-else* સ્ટેટમેન્ટ માટેના ટુંકારૂપ તરીકે કામ કરે છે.

વિશેષ ઓપરેટરો (Special Operators)

ગણિતીય, સંબંધસૂચક, તાર્કિક અને બિટવાઇઝ જેવા સામાન્ય ઓપરેટરો ઉપરાંત, C પ્રોગ્રામિંગમાં કેટલાક વિશેષ ઓપરેટરો (special operators) નો સમાવેશ થાય છે જે અનન્ય કાર્યો કરે છે. જે ઘણીવાર મેમરી મેનેજમેન્ટ, પોઇન્ટર અથવા પદાવલી મૂલ્યાંકન સાથે સંબંધિત હોય છે. ચાલો આપણે આમાંના કેટલાક વિશેષ ઓપરેટરો વિશે જાણીએ. ટેબલ 8.7 વિશેષ ઓપરેટરો અને તેના ઉપયોગોની યાદી આપે છે.

ઓપરેટર	નામ	વર્ણન	સાદું ઉદાહરણ
sizeof	Sizeof operator	ડેટા ટાઈપ અથવા ચલની સાઈઝ આપે છે (બાઇટમાં)	sizeof(int) 4 આપે છે
&	Address-of operator	ચલનું મેમરી એડ્રેસ પાછું આપે છે	&a (ચલ a નું મેમરી એડ્રેસ આપે છે)
*	Value-at operator (Pointer dereference)	પોઇન્ટર દ્વારા નિર્દેશિત મેમરી સ્થાન પર સંગ્રહિત મૂલ્યને ઍક્સેસ કરે છે	*ptr (પોઇન્ટર ptr દ્વારા નિર્દેશિત એડ્રેસ પરની કિંમત આપે છે)
. (dot)	Structure member access	સ્ટ્રક્ચર પ્રકારના ચલનો ઉપયોગ કરીને સ્ટ્રક્ચરના સભ્યને ઍક્સેસ કરે છે.	s.age (સ્ટ્રક્ચર s ના age સભ્યની કિંમત)
->	Structure pointer member access	સ્ટ્રક્ચર તરફ નિર્દેશ કરતા પોઇન્ટરનો ઉપયોગ કરીને સ્ટ્રક્ચરના સભ્યને ઍક્સેસ કરે છે.	ptr->age (જો ptr સ્ટ્રક્ચરનું પોઇન્ટર હોય)

ટેબલ 8.7 : વિશેષ ઓપરેટરો

sizeof ઓપરેટર (sizeof Operator)

sizeof ઓપરેટર એ લોકપ્રિય યુનરી ઓપરેટરમાંનો એક છે જે આપેલા ચલની સાઈઝ બાઇટમાં આપે છે. તેનો મુખ્યત્વે ઉપયોગ ડાયનેમિક મેમરી એલોકેશન (dynamic memory allocation) દરમિયાન અને વિવિધ સિસ્ટમ પર પોર્ટેબિલિટી (portability) સુનિશ્ચિત કરવા માટે થાય છે, જ્યાં ડેટા ટાઇપની સાઈઝ અલગ-અલગ હોઈ શકે.

sizeof ઓપરેટરનું ઉદાહરણ

```
/* Program to illustrate use of sizeof operator */

#include <stdio.h>
int main() {
    int num;
    char ch;

    /* To print size of num variable */
    printf("Size of num (int): %d bytes\n", sizeof(num));

    /* To print size of ch variable */
    printf("Size of ch (char): %d byte\n", sizeof(ch));

    return 0;
}

Result:
Size of num (int): 4 bytes
Size of ch (char): 1 byte
```

આ પ્રોગ્રામ num અને ch ચલો દ્વારા મેમરીમાં રોકાયેલ સાઈઝ પ્રિન્ટ કરશે. તમે તમારા આગામી ધોરણોમાં વિશેષ ઓપરેટરો વિશે વધુ ઊંડાણપૂર્વક અભ્યાસ કરશો.

નોંધ : પૂર્ણાંક ચલની ચોક્કસ સાઈઝ સિસ્ટમ આર્કિટેક્ચર અને કમ્પાઇલરના આધારે બદલાઈ શકે છે. ઉદાહરણ તરીકે, 64-બીટ સિસ્ટમ પર *int* ની સાઈઝ 4 બાઇટ અને *char* ની સાઈઝ 1 બાઇટ હોય છે.

પદાવલી (expression) શું છે?

પ્રોગ્રામિંગમાં, પદાવલી યુઝરને ગણતરીઓ કરવા, નિર્ણયો લેવા અને પ્રોગ્રામના પ્રવાહને નિયંત્રિત કરવા માટે જરૂરી સવલત પૂરી પાડે છે.

પદાવલી એ ચલ, અચલ અને ઓપરેટરોનું એક સંયોજન છે જે કોઈ મૂલ્ય ઉત્પન્ન કરે છે.

પદાવલી એક જ ચલ અથવા અચલની બનેલી હોઈ શકે છે, અથવા બહુવિધ ચલો, અચલો અને ઓપરેટરોના સંયોજનથી પણ બનેલી હોઈ શકે છે.

ઉદાહરણ

```
int a = 10, b = 5;
int result = a + b;     /* a + b is an expression */
```

આ ઉદાહરણમાં, a+b એ એક પદાવલી, a અને b ચલો અને + એ ઓપરેટર છે. આ પદાવલી a અને b માં સંગ્રહિત મૂલ્યોનો સરવાળો કરી તેને result ચલમાં સંગ્રહ કરે છે.

તે જ રીતે, પદાવલીઓ નિર્ણયો લેવા માટે તાર્કિક ઓપરેટરોનો સમાવેશ કરી શકે છે, જેમ કે a > b માં જો a એ b કરતા મોટો હોય તો 'સાચું' (true) તરીકે મૂલ્યાંકન કરે છે.

પદાવલી માત્ર ગણિતીય ક્રિયાઓ પૂરતી મર્યાદિત નથી, તેમાં સંબંધસૂચક, તાર્કિક અને બિટવાઈઝ ક્રિયાઓ પણ સામેલ હોઈ શકે છે. ઉદાહરણ તરીકે, (a > b) && (b < c) જેવી પદાવલી મૂલ્યોની તુલના કરવા માટે સંબંધસૂચક ઓપરેટરનો અને પરિણામોને જોડવા માટે તાર્કિક ઓપરેટરનો ઉપયોગ કરે છે, જે આખરે બુલિયન (boolean) મૂલ્ય આપે છે.

પદાવલી પ્રોગ્રામરને ગણતરીઓ કરવા, નિર્ણયો લેવા અને પ્રોગ્રામના પ્રવાહને નિયંત્રિત કરવા સક્ષમ બનાવે છે, જેનાથી સોફ્ટવેર ડેવલપમેન્ટનો પાયો નંખાય છે.

પદાવલીઓમાં ઓપરેટરનો ઉપયોગ

ટેબલ 8.8 પદાવલીઓમાં સામાન્યપણે વપરાતા ઓપરેટરનો ઉપયોગ દર્શાવે છે.

ઓપરેટરનો પ્રકાર	પદાવલીનું ઉદાહરણ	વર્ણન
Arithmatic	a + b	બે સંખ્યાનો સરવાળો
Relational	a > b	ચકાસો કે a એ b કરતા મોટો છે
Logical	a > 0 && b > 0	ચકાસો કે a એ b બંને ધન છે
Assignment	c = a + b	a + b નું પરિણામ ચલ c ને આપે છે
Increment/Decrement	a++, --b	મૂલ્યને 1 થી વધારે કે ઘટાડે
Bitwise	a & b	a અને b પર બીટવાઈઝ AND કરવું

ટેબલ 8.8 : પદાવલીઓમાં સામાન્યપણે ઉપયોગ થતાં ઓપરેટર

પદાવલીઓમાં ઓપરેટરનો ઉપયોગ C પ્રોગ્રામને નિર્ણયો લેવા, ગણતરીઓ કરવા અને પ્રોગ્રામના પ્રવાહને નિયંત્રિત કરવાની સગવડતા આપે છે. નિર્ણય સંરચનાઓ (Decision Structures) અને નિયંત્રણ સંરચનાઓ (Control Structures) સંબંધિત વધારે આગામી પ્રકરણમાં સમજીશું.

સારાંશ

આ પ્રકરણમાં, આપણે C પ્રોગ્રામિંગમાં ઓપરેટરો અને પદાવલીઓના મૂળભૂત ખ્યાલો વિશે જાણકારી મેળવી છે. તેઓ પ્રોગ્રામમાં થતી તમામ ગણતરીઓ અને તાર્કિક ક્રિયાઓ માટેના આધારસ્તંભ છે. આપણે ઓપરેટરોના વિવિધ પ્રકારો વિશે જાણકારી મેળવી છે, જેમાં ગાણિતિક ગણતરીઓ માટે ગણિતીય, તુલનાઓ માટે સંબંધસૂચક, શરતોને જોડવા માટે તાર્કિક, નીચલા-સ્તરના ડેટા હેરફેર માટે બિટવાઈઝ (bitwise) અને અનન્ય વધારો/ઘટાડો તથા sizeof અને & જેવા વિશેષ ઓપરેટરોનો સમાવેશ થાય છે. ઓપરેટર પ્રાથમિકતા અને સંબંધતાને સમજવું એ પદાવલીનું સચોટ અર્થઘટન કરવા અને તેનું નિર્માણ કરવા માટે મહત્વપૂર્ણ છે. પદાવલીમાં ઓપરેટર અને ઓપરેન્ડનું સંયોજન હોય છે જે આખરે એક જ મૂલ્યમાં મૂલ્યાંકન કરે છે. આ ખ્યાલોમાં નિપુણતા મેળવવી ખૂબ જ મહત્ત્વપૂર્ણ છે, કારણ કે તે પ્રોગ્રામરોને કાર્યક્ષમ, સચોટ અને મજબૂત C પ્રોગ્રામ બનાવવા માટે સક્ષમ બનાવે છે.

સ્વાધ્યાય

1. પ્રોગ્રામિંગમાં પદાવલીનો અર્થ શું છે?
2. C પ્રોગ્રામિંગ ભાષામાં ઓપરેટરોનો ઉપયોગ જણાવો.
3. C પ્રોગ્રામિંગના તાર્કિક ઓપરેટરોની યાદી બનાવો.
4. કયો ઓપરેટર ચલની કિંમતમાં 1 નો વધારો કરવા માટે ઉપયોગમાં લેવાય છે?

5. (=) અને (==) વચ્ચે શું ફરક છે?

6. અસાઈનમેન્ટ ઓપરેટરનો ઉપયોગ ઉદાહરણ સાથે સમજાવો.

7. એડ્રેસ-ઓફ ઓપરેટર (&) નો ઉપયોગ શું છે?

8. તાર્કિક AND ઓપરેટર (&&) નો ઉપયોગ યોગ્ય ઉદાહરણ સાથે સમજાવો.

9. કયા ઓપરેટરને વધુ પ્રાથમિકતા છે - Division (/) કે Addition (+) ?

10. C પ્રોગ્રામિંગમાં 5 + 2 * 3 + 5 પદાવલીનો આઉટપુટ શું આવશે?

11. **સાચું કે ખોટું જણાવો.**

(1) બાઈનરી ઓપરેટરને બે ઓપરેન્ડની જરૂર હોય છે.

(2) વધારા-સૂચક ઓપરેટર એ યુનરી ઓપરેટરનું ઉદાહરણ છે.

(3) ગુણાકાર (*) ઓપરેટરની પ્રાથમિકતા (precedence) સરવાળા (+) ઓપરેટર કરતાં વધારે છે.

(4) && ને C પ્રોગ્રામિંગમાં મોડ્યુલસ (modulus) ઓપરેટર તરીકે ઓળખવામાં આવે છે.

(5) || પ્રતીકનો ઉપયોગ તાર્કિક OR ઓપરેટરને રજૂ કરવા માટે થાય છે.

12. **ખાલી જગ્યાઓ પૂરો.**

(1) ટર્નરી ઓપરેટર એ C માં એક અનોખો ઓપરેટર છે જે ____________ ઓપરેન્ડ લે છે.

(2) મોડ્યુલસ (Modulus) ઓપરેટરનો ઉપયોગ માત્ર ____________ પ્રકારના ઓપરેન્ડ સાથે જ થઈ શકે છે.

(3) ____________ ઓપરેટર એક લોકપ્રિય યુનરી ઓપરેટર છે જે આપેલા ચલની બાઈટમાં સાઈઝ પરત કરે છે.

(4) ____________ પ્રતીકનો ઉપયોગ લોજિકલ NOT ઓપરેટરને રજૂ કરવા માટે થાય છે.

(5) ઈંક્રીમેન્ટ (++), ડિક્રીમેન્ટ (--) અને NOT (!) એ ____________ ઓપરેટરો (યુનરી / બાઈનરી / ટર્નરી) છે.

13. **બહુવિકલ્પી પ્રશ્નો. સૌથી યોગ્ય જવાબ પસંદ કરો.**

(1) નીચેનામાંથી કયો C માં ગણિતીય ઓપરેટર છે?

(a) && (b) || (c) + (d) !

(2) પદાવલી num1 == num2 શું ચકાસે છે?

(a) num2, num1 ને અસાઈન કરે છે (b) num1 અને num2 ની તુલના કરે છે
(c) num1 ને 1 થી વધારે છે (d) num1 અને num2 નો ગુણાકાર કરે છે

(3) નીચેના કોડનો અમલ કર્યા બાદ number નું મૂલ્ય શું હશે?

```
int number = 5;
number *= 5;
```

(a) 10 (b) 15 (c) 20 (d) 25

(4) નીચનામાંથી કયો C માં તાર્કિક AND ઓપરેટર છે?

(a) & (b) && (c) | (d) ||

(5) નીચેના C પ્રોગ્રામનો આઉટપુટ શું હશે?

```
#include <stdio.h>
int main() {
     int a = 5, b = 2;
     printf("%d", a % b);
     return 0;
}
```

(a) 1 (b) 2 (c) 5 (d) 10

(6) a + b * c પદાવલીના અમલ દરમ્યાન પહેલા કયું ઓપરેશન થશે?

(a) a + b (b) b * c (c) (a+b) અને (b *c) (d) a * c

(7) નીચેનામાંથી કયો ઓપરેટર મૂલ્ય 1 થી ઘટાડવા માટે વપરાશે?

(a) – – (b) ++ (c) – (d) *

(8) નીચેના C પ્રોગ્રામનો આઉટપુટ શું આવશે?

```
#include <stdio.h>
int main() {
    printf("%d", !(1));
    return 0;
}
```

(a) 0 (b) 1 (c) One (d) Not output

(9) C પ્રોગ્રામિંગમાં મોડ્યુલસ (%) ઓપરેટરનો પ્રાથમિક હેતુ શો છે?

(a) શેષ શોધવા (b) ગુણાકાર કરવા

(c) તાર્કિક AND મૂલ્ય શોધવા (d) ભાગફળ શોધવા

(10) NUM += 5 પદાવલી પ્રોગ્રામમાં શું કરશે?

(a) NUM ને 5 અસાઈન કરશે (b) NUM માં 5 ઉમેરશે

(c) NUM ને 5 ગુણશે (d) NUM માં કોઈ ફેરફાર નહી કરે

પ્રાયોગિક સ્વાધ્યાય

1. એવો C પ્રોગ્રામ લખો જે યુઝરને યોગ્ય ઈનપુટ ફંક્શનનો ઉપયોગ કરીને બે પૂર્ણાંક મૂલ્યો દાખલ કરવા કહે. નીચે જણાવેલ કાર્યોનું પરિણામ દર્શાવો

- બે સંખ્યાઓનો સરવાળો
- તફાવત (પ્રથમ સંખ્યા – બીજી સંખ્યા)
- બંને સંખ્યાઓનો ગુણાકાર
- પ્રથમ સંખ્યાને બીજી સંખ્યાથી ભાગાકાર કરતાં મળતું ભાગફળ - (integer division નું પરિણામ)
- પ્રથમ સંખ્યાને બીજી સંખ્યાથી ભાગાકાર કરતાં મળતી શેષ - (division પછીનો remainder અથવા modulus)

2. એવો C પ્રોગ્રામ લખો જે વધારા-સૂચક (Increment) ઓપરેટરના ઉપયોગનું પ્રદર્શન કરે. યુઝર પાસેથી NUM નામના ચલનું મૂલ્ય વાંચો. પ્રોગ્રામમાં ++NUM અને NUM++ વચ્ચેનો તફાવત તેનું મૂલ્ય પ્રિન્ટ કરીને બતાવો.

3. એવો C પ્રોગ્રામ લખો જે સંબંધસૂચક ઓપરેટરના ઉપયોગનું પ્રદર્શન કરે. તમારા પ્રોગ્રામમાં ==, !=, <, >, <=, >= ઓપરેટરો નો ઉપયોગ કરો. બે સંખ્યાઓ વાંચો, તેમની તુલના કરો અને પરિણામ મુજબ યોગ્ય સંદેશો પ્રિન્ટ કરો.

4. એવો C પ્રોગ્રામ લખો જે ટર્નરી ઓપરેટરનો ઉપયોગ કરીને બે સંખ્યાઓમાંથી મહત્તમ સંખ્યા શોધે. **સૂચન :** બે સંખ્યાઓની તુલના કરવા અને કઈ મોટી છે તે નક્કી કરવા માટે ટર્નરી ઓપરેટર (?:) નો ઉપયોગ કરો.

5. નીચેની પદાવલીનું મૂલ્યાંકન કરવા માટેનો C પ્રોગ્રામ લખો.

```
a + b * c / d - e
```

જરૂરી કિંમતો યુઝર પાસેથી વાંચો.

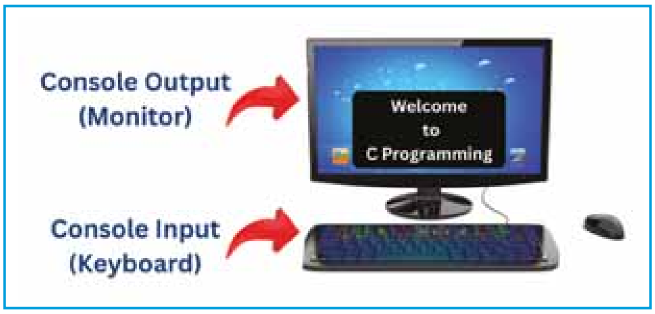
6. નીચેનો C પ્રોગ્રામ ગણિતીય ઓપરેટરોનો ઉપયોગ દર્શાવે છે. પરંતુ, તેમાં વાક્યરચનાને લગતી તેમજ તાર્કિક ભૂલો છે. આ બધી ભૂલો ઓળખી તેને દૂર કરી પ્રોગ્રામને ફરીથી લખો જેથી એ સાચો આઉટપુટ પ્રિન્ટ કરે.

```
/* Program to illustrate use of Arithmatic Operators */
#inclide <stdio.h>
int main() {
    int a=20, b=5;

    printf("Result of Arithmatic Operations on a and b:\n");

    printf("Addition       : a * b = %d \n", a + b);
    printf("Subtraction    : a - b = %d \n , a - b)
    print("Multiplication  : a + b = %d \n , a * b);
    print("Division        : a / b = %d \n , a / b)
    printf("Modulus        : a % b = %d \n", a % b);

    return 0;
}
```

ઈનપુટ આઉટપુટ ઓપરેશન

પરિચય

અગાઉના પ્રકરણમાં, આપણે C પ્રોગ્રામિંગના ઓપરેટરો અને પદાવલીઓના ઉપયોગ વિશે જાણ્યું. આ પ્રકરણ C પ્રોગ્રામિંગમાં ઈનપુટ અને આઉટપુટ (I/O) ઓપરેશનો પર ધ્યાન કેન્દ્રિત કરે છે. C પ્રોગ્રામિંગમાં, ઈનપુટ/આઉટપુટ (I/O) ઓપરેશનો યુઝર અને સિસ્ટમ સાથે સંવાદ કરવા માટે આવશ્યક છે. I/O ઓપરેશનો પ્રોગ્રામને બાહ્ય સ્રોતો (જેમ કે યુઝર ઈનપુટ અથવા ફાઈલ ઈનપુટ) માંથી ડેટા પ્રાપ્ત કરવાની અને પરિણામોને આઉટપુટ સ્ક્રીન પર પ્રદર્શિત કરવાની સવલત આપે છે. C માં પ્રમાણભૂત I/O ફંક્શન, જેમ કે આઉટપુટ માટેનું *printf()* અને ઈનપુટ માટેનું *scanf()*, stdio.h હેડર ફાઈલમાં વ્યાખ્યાયિત થયેલા છે. I/O ઓપરેશનો શીખવા એ C શીખવાનો એક મૂળભૂત તબક્કો છે, કારણ કે તે સંવાદીત પ્રોગ્રામો લખવાનો આધાર પૂરો પાડે છે. આપણે વિવિધ ઈનપુટ ફંક્શનોનો અભ્યાસ કરીશું, જે દરેક યુઝર ઈનપુટને ખાસ પ્રકારે દાખલ કરવા માટે રચાયેલા છે. તે જ રીતે, આપણે આવશ્યક આઉટપુટ ફંક્શનોની ચર્ચા કરીશું જેનો ઉપયોગ પ્રોગ્રામના આઉટપુટને પ્રદર્શિત કરવા માટે થાય છે. આ પ્રકરણ એસ્કેપ સિક્વન્સ (escape sequences) પણ રજૂ કરશે, જે આઉટપુટની વાંચનીયતા વધારે છે. વધુમાં, આપણે ફોર્મેટેડ I/O ફંક્શનો અને ડેટાને વ્યવસ્થિત પ્રદર્શિત કરવા માટે ઉપયોગમાં લેવાતા વિવિધ ફોર્મેટ સ્પૅસિફાયર્સ (format specifiers) ના ઉપયોગનો અભ્યાસ કરીશું. આ પ્રકરણના અંત સુધીમાં, તમે ઈન્ટરેક્ટિવ C પ્રોગ્રામ બનાવવા માટે વિવિધ ફંક્શનનો અસરકારક રીતે ઉપયોગ કેવી રીતે કરવો તેની વ્યાપક સમજ મેળવી લેશો.



આકૃતિ 9.1 : કન્સોલ I/O ડિવાઈસીસ

પ્રોગ્રામિંગમાં I/O ઓપરેશન શું છે?

પ્રોગ્રામિંગમાં ઈનપુટ અને આઉટપુટ (I/O) ઓપરેશન એવી પ્રક્રિયાઓ છે જે પ્રોગ્રામને યુઝર પાસેથી ડેટા મેળવવા (ઈનપુટ) અને યુઝરને પરિણામો પ્રિન્ટ કરવાની (આઉટપુટ) સગવડતા પૂરી છે. જુઓ આકૃતિ 9.1.

ઈનપુટ ઓપરેશન (Input Operation) : આમાં કીબોર્ડ, માઉસ, ફાઈલ અથવા નેટવર્ક જેવા ઉપકરણોમાંથી ડેટા મેળવવાનો સમાવેશ થાય છે. ઉદાહરણ તરીકે, કીબોર્ડ દ્વારા એક નંબર દાખલ કરવો.

આઉટપુટ ઓપરેશન (Output Operations) : આમાં સ્ક્રીન, પ્રિન્ટર અથવા ડીસ્ક (ફાઈલ) જેવા ઉપકરણોને ડેટા મોકલવાનો સમાવેશ થાય છે. ઉદાહરણ તરીકે, વિદ્યાર્થીનું પરિણામ સ્ક્રીન પર પ્રિન્ટ કરવું.

C પ્રોગ્રામિંગમાં, I/O ઓપરેશનો સામાન્ય રીતે *stdio.h* (સ્ટાન્ડર્ડ ઈનપુટ/આઉટપુટ) લાઈબ્રેરીમાં વ્યાખ્યાયિત પ્રમાણભૂત ફંક્શનો દ્વારા કરવામાં આવે છે. નીચે લોકપ્રિય I/O ફંક્શનો આપેલા છેઃ

- scanf(), gets(), getch(), અને fgets() ઈનપુટ માટે
- printf(), puts(), putc(), અને fputs() આઉટપુટ માટે

આ ઓપરેશનો આપણા પ્રોગ્રામને વધુ સંવાદાત્મક (ઈન્ટરેક્ટિવ) અને યુઝર-ફ્રેન્ડલી (યુઝર-અનુકૂળ) બનાવવા માટે આવશ્યક છે. આવા ફંક્શન પ્રોગ્રામના અમલ દરમિયાન યુઝરને સોફ્ટવેર સાથે સંવાદ કરવાની મંજૂરી આપે છે.

પ્રોગ્રામિંગમાં I/O ઉપકરણોનો ઉપયોગ

I/O (ઈનપુટ/આઉટપુટ) ડિવાઈસ એ હાર્ડવેરના એવા ઘટકો છે જે કમ્પ્યૂટર સિસ્ટમને બાહ્ય વિશ્વ સાથે સંવાદ કરવાની મંજૂરી આપે છે. પ્રોગ્રામ કન્સોલ ઈનપુટ ડિવાઈસમાંથી ઈનપુટ મેળવે છે અને કન્સોલ આઉટપુટ ડિવાઈસ પર આઉટપુટ પ્રદાન કરે છે, જે આકૃતિ 9.1માં દર્શાવેલ છે.

- ઈનપુટ ડિવાઈસનો ઉપયોગ કમ્પ્યૂટર સિસ્ટમમાં ડેટા મોકલવા માટે થાય છે. ઈનપુટ ડિવાઈસના લોકપ્રિય ઉદાહરણો કીબોર્ડ, માઉસ, સ્કેનર અને માઈક્રોફોન છે.
- આઉટપુટ ડિવાઈસનો ઉપયોગ ડેટા પ્રદર્શિત કરવા માટે થાય છે. આઉટપુટ ડિવાઈસના લોકપ્રિય ઉદાહરણો મોનિટર, પ્રિન્ટર, સ્પીકર અને LEDs છે.
- કેટલાક ડિવાઈસીસ, જેમ કે ટચસ્ક્રીન, ઈનપુટ અને આઉટપુટ બંને ડિવાઈસ તરીકે કાર્ય કરે છે.

પ્રોગ્રામિંગમાં, ફાઈલ I/O (File I/O) પણ ખૂબ જ ઉપયોગી છે, જેનો ઉપયોગ હાર્ડ ડિસ્ક જેવા સ્ટોરેજ ડિવાઈસમાંથી વાંચવા અથવા તેના પર લખવા માટે થાય છે. તમે તમારા ઉચ્ચ અભ્યાસમાં ફાઈલ I/O વિશે શીખશો.

ચાલો એક સરળ પ્રોગ્રામથી શરૂઆત કરીએ જે વિદ્યાર્થીનો રોલ નંબર વાંચશે અને પ્રિન્ટ કરશે.

```
/* Program to read and print a roll number of a student. */
#include <stdio.h>
int main() {
    int rollNumber;

    printf("Enter your roll number: ");
    scanf("%d", &rollNumber);

    printf("Your roll number is: %d\n", rollNumber);
    return 0;
}

Result:

Enter your roll number: 101
Your roll number is: 101
```

આ પ્રોગ્રામ C માં પ્રમાણભૂત ઈનપુટ/આઉટપુટ ફંક્શનનો ઉપયોગ કરીને યુઝર પાસેથી ઈનપુટ કેવી રીતે લેવો અને તેને સ્ક્રીન પર કેવી રીતે પ્રિન્ટ કરવો તે દર્શાવે છે. તે યુઝરનો ઈનપુટ સંગ્રહિત કરવા માટે rollNumber નામનો એક પૂર્ણાંક (integer) ચલ ઘોષિત કરીને શરૂ થાય છે. ત્યારબાદ, *printf()* ફંક્શન યુઝરને રોલ નંબર દાખલ કરવા માટેનો સંદેશ પ્રદર્શિત કરે છે. *scanf()* ફંક્શનનો ઉપયોગ પૂર્ણાંક ઈનપુટ વાંચવા અને તેને rollNumber ચલમાં સંગ્રહિત કરવા માટે થાય છે. છેલ્લે, પ્રોગ્રામ યુઝર દ્વારા દાખલ કરવામાં આવેલ મૂલ્ય પહેલા "Your roll number is: " (તમારો રોલ નંબર છે:) સંદેશ પ્રદર્શિત કરવા માટે *printf()* ફંક્શનનો ઉપયોગ કરે છે.

C પ્રોગ્રામિંગમાં <stdio.h> ની ભૂમિકા

<stdio.h> ફાઈલનું પૂરું નામ સ્ટાન્ડર્ડ ઈનપુટ આઉટપુટ હેડર (Standard Input Output Header) છે. તે C કમ્પાઈલરમાં એક સમાવિષ્ટ હેડર ફાઈલ છે જે ઈનપુટ અને આઉટપુટ ઓપરેશન માટે જરૂરી ફંક્શનો પુરા પાડે છે, જેમ કે યુઝર પાસેથી ડેટા વાંચવો અથવા સ્ક્રીન પર આઉટપુટ પ્રદર્શિત કરવો. આપણા પ્રોગ્રામમાં *<stdio.h>* નો સમાવેશ કર્યા વિના, *printf()* અને *scanf()* જેવા ફંક્શનો કમ્પાઈલેશન ત્રુટિ (compilation errors) આપશે. તેમાં વ્યાખ્યાયિત સામાન્ય રીતે ઉપયોગમાં લેવાતા ફંક્શનો નીચે સૂચિબદ્ધ છે :

- printf() – આઉટપુટ પ્રદર્શિત કરવા
- scanf() – ઈનપુટ લેવા
- getchar(), putchar() – અક્ષર (character) ઈનપુટ/આઉટપુટ માટે
- gets(), fgets(), puts() – સ્ટ્રીંગ (શબ્દો) ઈનપુટ/આઉટપુટ માટે
- fopen(), fclose(), fscanf(), fprintf() – ફાઈલ સંચાલન માટે

ઈનપુટ ફંક્શનો (Input Functions)

C પ્રોગ્રામિંગમાં, ઈનપુટ ફંક્શનનો ઉપયોગ પ્રોગ્રામના અમલ દરમિયાન યુઝર પાસેથી ડેટા મેળવવા માટે થાય છે. <*stdio.h*> હેડર ફાઈલમાં વિવિધ પ્રકારના ઈનપુટ જેમ કે અક્ષરો (characters), સ્ટ્રિંગ્સ (strings) અને સંખ્યાઓ (numbers) વાંચવા માટે અલગ-અલગ ફંક્શનો ઉપલબ્ધ છે. ટેબલ 9.1 સામાન્ય ઈનપુટ ફંક્શનો અને તેનો ઉપયોગ દર્શાવે છે:

ફંક્શન	ઉપયોગ	ઉદાહરણ
`scanf()`	number/char/string વાંચવા	`scanf("%d", &num);`
`getchar()`	એક character વાંચવા	`ch = getchar();`
`gets()`	string (unsafe) વાંચવા	`gets(name);`
`fgets()`	string (safe) વાંચવા	`fgets(name, 50, stdin);`

ટેબલ 9.1 : ઈનપુટ ફંક્શનો

ચાલો લોકપ્રિય ઈનપુટ ફંક્શનોનો ઉપયોગ ઉદાહરણ સાથે સમજીએ.

scanf() ફંક્શન

scanf() ફંક્શન એ સૌથી લોકપ્રિય ઈનપુટ ફંક્શનોમાંનું એક છે, જેનો ઉપયોગ કીબોર્ડમાંથી પૂર્ણાંકો (integers), અપૂર્ણાંકો (floats), અક્ષરો (characters) અને સ્ટ્રિંગ્સ (strings) જેવા ઈનપુટ વાંચવા માટે થાય છે. *scanf()* ફંક્શનની સામાન્ય વાક્યરચના નીચે મુજબ છે:

```
scanf("format_specifier", &variable);
```

જ્યાં,

- **"format_specifier"** (ફોર્મેટ સ્પેસિફાયર) : કયા પ્રકારનો ડેટા વાંચવો છે તે જણાવે છે. જેમ કે, %d પૂર્ણાંક (*int*) માટે, %f અપૂર્ણાંક (*float*) માટે વગેરે.
- **&variable :** એ ચલનું એડ્રેસ છે, જ્યાં વાંચવામાં આવેલ ઈનપુટ ડેટા સંગ્રહ કરવામાં આવશે.

scanf() ના ઉદાહરણો

યુઝર પાસેથી પૂર્ણાંક (integer) સંખ્યા વાંચવા માટે, આપણે નીચે બતાવ્યા પ્રમાણે "%d" ફોર્મેટ સ્પેસિફાયર સાથે *scanf()* ફંક્શનનો ઉપયોગ નીચે મુજબ કરી શકીએ છીએ:

```
int marks;
scanf("%d", &marks);
```

આજ રીતે, યુઝર પાસેથી અપૂર્ણાંક (*float*) સંખ્યા વાંચવા માટે, આપણે *scanf()* ફંક્શન સાથે "%f" ફોર્મેટ સ્પેસિફાયરનો ઉપયોગ નીચે પ્રમાણે કરી શકીએ છીએ:

```
float percentage;
scanf("%f", &percentage);
```

વિદ્યાર્થીનું નામ વાંચવા, આપણે નીચે પ્રમાણે વિધાનો લખી શકીએ:

```
char name[30];
scanf("%s", name);
```

જુઓ કે *scanf()* માં %s ફોર્મેટ સ્પેસિફાયર એક જ શબ્દ વાંચી શકે છે. *scanf()* ફંક્શન ઈનપુટમાં જ્યાં સુધી પહેલી ખાલી જગ્યા (space) આવે નહીં ત્યાં સુધીના અક્ષરો જ વાંચે છે. આથી, જો આપણે name ચલમાં બે શબ્દો દાખલ કરીએ (જેમ કે "Mudra Patel"), તો તે ફક્ત પહેલો શબ્દ "Mudra" જ સંગ્રહિત કરશે.

વિવિધ ફોર્મેટ સ્પેસિફાયર વિશેની વિગતવાર માહિતી આ અધ્યાયના આગળના ભાગમાં આપવામાં આવી છે.

getchar() ફંક્શન

getchar() ફંક્શનનો ઉપયોગ કીબોર્ડ જેવા સ્ટાન્ડર્ડ ઈનપુટ ડિવાઈસમાંથી એક જ અક્ષર (single character) વાંચવા માટે થાય છે. ઈનપુટ કરેલ અક્ષરને સામાન્ય રીતે આપણે *char* પ્રકારના ચલમાં સંગ્રહિત કરીએ છીએ. જ્યારે યુઝર પાસેથી ચોક્કસ અક્ષર-બાદ-અક્ષર (character-by-character) ઈનપુટ લેવાની જરૂર હોય, ત્યારે getchar() ફંક્શનનો ઉપયોગ કરવો જોઈએ.

getchar() નું ઉદાહરણ

```
char ch;
ch = getchar();     /* This will read one character from keyboard */
```

gets() ફંક્શન

gets() ફંક્શનનો ઉપયોગ ખાલી જગ્યા સહિતની આખી લાઈન (ઘણા શબ્દો ધરાવતી સ્ટ્રિંગ) વાંચવા માટે થાય છે. આ ફંક્શન સ્ટાન્ડર્ડ ઈનપુટ (stdin) માંથી નવી લાઈન (\n) ન મળે ત્યાં સુધી ઈનપુટની લાઈન વાંચે છે. આનો અર્થ એ છે કે, જ્યાં સુધી આપણે કીબોર્ડ પર Enter (નવી લાઈન) દબાવીશું નહીં, ત્યાં સુધી *gets()* ફંક્શન કીબોર્ડમાંથી વાંચવાનું ચાલુ રાખશે. પણ આપણે *gets()* માં વાંચવા માટેના અક્ષરોની સંખ્યા સ્પષ્ટ કરતા નથી, તેથી તે અસુરક્ષિત બને છે કારણ કે તે બફર ઓવરફ્લો (buffer overflow) ની સમસ્યા તરફ દોરી શકે છે. સરળ શબ્દોમાં કહીએ તો, બફર ઓવરફ્લો ત્યારે થાય છે જ્યારે કોઈ પ્રોગ્રામ ચલની મેમરી સ્પેસ (જેને બફર કહેવાય છે) માં તે ખરેખર સમાવી શકે તેના કરતાં વધુ ડેટા સંગ્રહવાનો પ્રયાસ કરે છે.

gets() નું ઉદાહરણ

```
char name[50];
gets(name);     /* This will allow us to enter full name including spaces. */
```

નોંધ : સુરક્ષાના કારણોસર, *gets*() ફંક્શન વાપરવું સલામત નથી. તેના બદલે *fgets()* નો ઉપયોગ કરવો વધુ સુરક્ષિત અને ભલામણપાત્ર છે.

fgets() ફંક્શન

આ ફંક્શનનો ઉપયોગ યુઝર પાસેથી ખાલી જગ્યા સહિતની સ્ટ્રિંગ્સ (ઈનપુટ લાઈન) વાંચવા માટે થાય છે. *fgets()* ફંક્શનમાં, આપણે યુઝર પાસેથી વાંચવા માંગતા હોઈએ તેટલા અક્ષરોની સંખ્યા સ્પષ્ટ કરી શકીએ છીએ, તેથી તેને *gets()* ફંક્શનની તુલનામાં વધુ સુરક્ષિત ફંક્શન ગણવામાં આવે છે. *fgets()* ફંક્શન કાં તો નવી લાઈન (\n) મળે ત્યારે અથવા મર્યાદા (limit) પૂરી થાય ત્યારે વાંચવાનું બંધ કરશે. ફંક્શનમાં બફરનું કદ સ્પષ્ટ કરીને, આપણે બફર ઓવરફ્લો અટકાવી શકીએ છીએ. *fgets()* ફંક્શનની સામાન્ય વાક્યરચના નીચે મુજબ છે:

```
fgets(variable, size, stdin);
```

fgets() નું ઉદાહરણ

```
char name[50];
fgets(name, 50, stdin);
```

આ કોડમાં,

- `char name[50];` વિદ્યાર્થીનું નામ સંગ્રહવા ચલ name ઘોષિત કરે છે.

- fgets(name, 50, stdin); આ વિધાન સ્ટાન્ડર્ડ ઇનપુટ ડિવાઇસ (કીબોર્ડ) માંથી ખાલી જગ્યા સહિતની આખી લાઇન વાંચશે. તે કુલ 50 અક્ષરો સંગ્રહિત કરી શકે છે. આનો અર્થ છે કે તે 49 જેટલા અક્ષરો ઇનપુટ તરીકે લઈ શકે છે, જેમાં વત્તા એક નલ કેરેક્ટર ('\0') છે, જે સ્ટ્રિંગના અંતને દર્શાવે છે. અહીં "stdin" નો અર્થ સ્ટાન્ડર્ડ ઇનપુટ (કીબોર્ડ) થાય છે.

નોંધ : સુરક્ષાના જોખમોને કારણે *gets*() ફંક્શનને C11 સ્ટાન્ડર્ડમાંથી દૂર કરવામાં આવ્યું છે. સલામત ઇનપુટ મેનેજમેન્ટ માટે *fgets()* નો ઉપયોગ કરો.

મિક્સ ડેટાનું વાંચન (Reading Mixed Data)

આપણે એક અક્ષર વાંચવા માટે *getchar()* અને એક આખી લાઇન વાંચવા માટે *gets()* નો ઉપયોગ કેવી રીતે કરવો તે શીખી લીધું છે. હવે, ચાલો જોઈએ કે એક જ *scanf()* ફંક્શનનો ઉપયોગ કરીને એક જ ઇનપુટ વિધાનમાં *int, char,* અને *string* જેવા વિવિધ ડેટા પ્રકારો કેવી રીતે વાંચી શકાય.

```
/* Program to illustrate mixed data input */
#include <stdio.h>

int main() {
    int age;
    char gender;
    char name[30];

    printf("Enter age, gender (M/F), and name:");
    scanf("%d %c %s", &age, &gender, name);

    printf("\nYou entered:\n");
    printf("Age: %d \n", age);
    printf("Gender: %c \n", gender);
    printf("Name: %s \n", name);

    return 0;
}
```

Result:

```
Enter age, gender (M/F), and name:20 F Mudra

You entered:
Age: 20
Gender: F
Name: Mudra
```

આ પ્રોગ્રામમાં, માત્ર એક જ *scanf()* વિધાનનો ઉપયોગ ત્રણ જુદા જુદા યુઝર ઇનપુટ વાંચવા માટે થાય છે. આ *scanf()* વિધાનમાં, %d એક પૂર્ણાંક (age), % c એક અક્ષર (gender), અને %s એક સ્ટ્રિંગ (name, પ્રથમ સ્પેસ પર અટકે છે) વાંચે છે.

એ નોંધવું અગત્યનું છે કે આ પ્રકારના મિક્સ-મોડ ઇનપુટનો ઉપયોગ કરતી વખતે, જો %c નો ઉપયોગ %d પછી તરત જ કરવામાં આવે, તો તે વધેલો ન્યૂલાઇન અક્ષર ('\n') વાંચી શકે છે, અને આપણે પછીના ઇનપુટને યોગ્ય રીતે વાંચી શકીશું નહીં. આ ટાળવા માટે, %c પહેલાં એક સ્પેસનો ઉપયોગ કરવો, જેમ કે નીચેના વિધાનમાં દર્શાવેલ છે:

```
scanf("%d %c", &age, &gender);
```

આઉટપુટ ફંક્શનો (Output Functions)

C પ્રોગ્રામિંગ કન્સોલ પર આઉટપુટ દર્શાવવા માટે ઘણા ફંક્શનો પૂરાં પાડે છે. આઉટપુટ ફંક્શનોનો મુખ્ય હેતુ પ્રોગ્રામમાંથી ડેટાને પ્રમાણભૂત આઉટપુટ ડિવાઈસ પર મોકલવાનો છે. મોટા ભાગના કિસ્સાઓમાં, આપણે મોનિટરનો ઉપયોગ આપણા પ્રમાણભૂત આઉટપુટ ડિવાઈસ તરીકે કરીએ છીએ. આઉટપુટ ફંક્શનો મુખ્યત્વે *<stdio.h>* હેડર ફાઈલમાં જોવા મળે છે. ચાલો આપણે સૌથી વધુ ઉપયોગમાં લેવાતા આઉટપુટ ફંક્શનોને સરળ ઉદાહરણો સાથે સમજીએ. ટેબલ 9.2 સામાન્ય આઉટપુટ ફંક્શનો અને તેનો ઉપયોગ દર્શાવેલ છે.

ફંક્શન	હેતુ/ઉપયોગ	ઉદાહરણ	નમૂનાનો આઉટપુટ
`printf()`	ફોરમેટ સાથે text, variables, numbers પ્રિન્ટ કરવા	`int age=25;` `printf("Age=%d", age);`	Age = 25
`putchar()`	એક character પ્રિન્ટ કરવો	`putchar('A');`	A
`puts()`	string ને newline સાથે પ્રિન્ટ કરવી	`puts("Hello");`	Hello (ટેક્સ્ટ પછી નવી લાઈન)
`fputs()`	string ને newline વગર પ્રિન્ટ કરવી	`fputs("Hi", stdout);`	Hi

ટેબલ 9.2 : આઉટપુટ ફંક્શનો

ચાલો સામાન્ય આઉટપુટ ફંક્શનના ઉપયોગને ઉદાહરણ સાથે સમજીએ.

printf() ફંક્શન

printf() ફંક્શન C માં ફોર્મેટેડ આઉટપુટ દર્શાવવા માટેનો સૌથી શક્તિશાળી અને લવચીક રસ્તો છે. પ્રોગ્રામમાં *printf()* નો ઉપયોગ કરવા માટે, આપણે #include <stdio.h> વિધાનનો ઉપયોગ કરીને *stdio.h* હેડર ફાઈલને ઉમેરવી પડે છે. *printf()* એ લગભગ દરેક C પ્રોગ્રામમાં સૌથી વધુ ઉપયોગમાં લેવાતું ફંક્શન છે. તે આપણને ટેક્સ્ટ, ચલની કિંમતો, પરિણામો વગેરેને નિશ્ચિત ફોર્મેટમાં પ્રિન્ટ કરવાની સગવડતા આપે છે.

printf() ની વાક્યરચના નીચે પ્રમાણે છે:

```
printf("control string", var1, var2, var3, … varN);
```

જ્યાં, var1, var2, var3 …varN એ ચલ અથવા અચલ અથવા પદાવલી હોઈ શકે છે. આઉટપુટ ફોર્મેટિંગ માટેની સૂચનાઓ કંટ્રોલ સ્ટ્રીંગ (Control String) માં લખવામાં આવે છે. કંટ્રોલ સ્ટ્રીંગમાં નીચેનામાંથી કોઈપણ અથવા બધા ઘટકો સામેલ હોઈ શકે છે:

- મોનિટર પર પ્રદર્શિત થવા માટેનો અક્ષરોનો સમૂહ (string).
- દરેક ચલ માટેનું એક ફોર્મેટ સ્પેસિફાયર (Format Specifier) (જેમ કે %d, %c, %f, વગેરે).
- એસ્કેપ સિક્વન્સ અક્ષરો (Escape Sequence characters) જેમ કે નવી લાઈન (\n), ટેબ (\t), બેકસ્પેસ (\b) વગેરે.

કંટ્રોલ સ્ટ્રિંગ (Control String) ની અંદરના ફોર્મેટ સ્પૅસિફાયર (Format Specifiers) ખાલી જગ્યા (space) દ્વારા અલગ પડે છે અને % થી શરૂ થાય છે. *printf()* ફંક્શનનો અનુગામી ભાગ ચલોની સૂચિ ધરાવે છે, જેના મૂલ્યોને આઉટપુટ તરીકે સ્કીન પર પ્રિન્ટ કરવાના હોય છે. આ ચલો કંટ્રોલ સ્ટ્રિંગમાં પ્રદાન કરેલ ફોર્મેટ સ્પૅસિફિકેશન સાથે સંખ્યા, ક્રમ અને પ્રકારમાં મેળ ખાતા હોવા જોઈએ, અન્યથા ખોટો અથવા અયોગ્ય આઉટપુટ પ્રદર્શિત થશે.

printf() ફંક્શન જુદા જુદા પ્રકારના ડેટાને છાપવા માટે %d, %f, %s વગેરે જેવા સામાન્ય ફોર્મેટ સ્પેસિફાયર (Format Specifiers) નો ઉપયોગ કરે છે. આ ફંક્શન ન્યૂલાઈન (newline) આપમેળે ઉમેરતું નથી. આઉટપુટમાં નવી લાઈન લેવા માટે, આપણે *printf()* ફંક્શનમાં \n નો સમાવેશ કરવો પડશે.

printf() ફંક્શનની એક મર્યાદા એ છે કે તેમાં આંતરિક ભૂલ ચકાસણી પદ્ધતિનો અભાવ હોય છે. આનો અર્થ એ છે કે ખોટા ફોર્મેટ સ્પેસિફાયર ઉપયોગ કરવાથી રનટાઈમ સમસ્યાઓ થઈ શકે છે. વધુમાં, *printf()* માત્ર કન્સોલ આઉટપુટ માટે જ કામ કરે છે અને GUI (ગ્રાફિકલ યુઝર ઈન્ટરફેસ) અથવા ફાઈલ આઉટપુટ માટે તેનો ઉપયોગ કરી શકાતો નથી.

printf() નું ઉદાહરણ

```
/* Program to illustrate use of printf() with different format
specifiers. */

#include <stdio.h>
int main() {
    int age = 18;
    float score = 94.50;
    char grade = 'A';
    char name[20] = "Nirva";
    printf("Name:%s, Age:%d, Grade:%c, Score:%.2f", name,
    age, grade, score);
    return 0;
}

Result:

Name:Nirva, Age:18, Grade:A, Score:94.50
```

putchar() ફંક્શન

putchar() એક જ અક્ષરને આઉટપુટ સ્ક્રીન પર પ્રિન્ટ કરે છે. તે અક્ષર-બાદ-અક્ષર (character-by-character) આઉટપુટ માટે સરળ અને કાર્યક્ષમ છે. તે સ્ટ્રિંગ્સ (strings) અથવા એરે (arrays) ને પ્રિન્ટ કરવા માટે લૂપમાં સારી રીતે કામ કરે છે. ઉપરાંત, તેના સરળ માળખાને કારણે, તેને વાંચવું અને સમજવું સરળ છે. નોંધ લો કે માત્ર એક *putchar()* ફંક્શન સ્ટ્રિંગ્સ અથવા એરેને હેન્ડલ કરી શકતું નથી. આને હેન્ડલ કરવા માટે, તમારે તેનો ઉપયોગ અનેક વખત અથવા લૂપની અંદર કરવાની જરૂર પડશે.

putchar() ફંક્શનમાં આઉટપુટ પ્રિન્ટ કરતી વખતે ફોર્મેટિંગ ક્ષમતાઓ મર્યાદિત હોય છે. ફંક્શનમાં ફોર્મેટ સ્પેસિફાયરના અભાવને કારણે, અક્ષર પ્રિન્ટ કર્યા પછી ખાલી જગ્યાઓ (spacing) અથવા ન્યૂલાઈન (newline) ઉમેરવા પર કોઈ નિયંત્રણ રહેતું નથી.

putchar() નું ઉદાહરણ

```
char ch = 'A';

putchar(ch);
```

સ્ક્રીન પર આઉટપુટ A પ્રિન્ટ કરશે.

puts() ફંક્શન

puts() C માં એક સ્ટાન્ડર્ડ આઉટપુટ ફંક્શન છે, જે <*stdio.h*> હેડર ફાઇલમાં વ્યાખ્યાયિત થયેલ છે. તેનો ઉપયોગ આખી સ્ટ્રિંગને કન્સોલ આઉટપુટ પર પ્રિન્ટ કરવા માટે થાય છે. સ્ટ્રિંગ પ્રિન્ટ કર્યા પછી તે આપમેળે અંતે એક નવી લાઈન ઉમેરે છે. સ્ટ્રિંગ આઉટપુટ માટે તે *printf()* કરતાં વાપરવામાં સરળ છે. પ્રિન્ટ કર્યા પછી તે કર્સરને આપમેળે પછીની લાઈન પર ખસેડે છે.

puts() ફંક્શનના મુખ્ય ફાયદાઓ આ પ્રમાણે છે: તે ફોર્મેટિંગ વગર સ્ટ્રિંગ્સ પ્રિન્ટ કરવા માટે સરળ અને ઝડપી છે. તે આપણને મેન્યુઅલી \n ઉમેરવામાંથી મુક્તિ આપે છે. તે *printf()* ની સરખામણીમાં સરળ છે.

puts() ની કેટલીક મર્યાદાઓ આ પ્રમાણે છે: તેમાં ફોર્મેટિંગ પર કોઈ નિયંત્રણ હોતું નથી. તે સંખ્યાઓ (numbers) અને અક્ષરો (characters) પ્રિન્ટ કરવા માટે યોગ્ય નથી.

puts() નું ઉદાહરણ

```
/* Program to illustrate use of puts() function */

#include <stdio.h>
int main() {
    char name[20] = "Pratham";
    puts("Welcome");
    puts(name);
    return 0;
}

Result:

Welcome
Pratham
```

નોંધ લો કે પ્રથમ *puts()* ફંક્શનમાં \n નો ઉપયોગ કર્યા વિના પણ, વિદ્યાર્થીનું નામ નવી લાઈન પર પ્રિન્ટ થાય છે.

fputs() ફંક્શન

C પ્રોગ્રામિંગમાં *fputs()* ફંક્શનનો ઉપયોગ પ્રોગ્રામના આઉટપુટને કાં તો ફાઇલ પર અથવા સ્ટાન્ડર્ડ આઉટપુટ સ્ટ્રીમ (stdout) પર લખવા માટે થાય છે. તે સ્ટાન્ડર્ડ I/O લાઈબ્રેરી, <*stdio.h*> માં સામેલ છે. *fputs()* વડે, આપણે નક્કી કરી શકીએ છીએ કે આપણે આપણું આઉટપુટ ક્યાં પ્રદર્શિત કરવા માંગીએ છીએ, પછી ભલે તે સ્ક્રીન પર હોય કે ફાઇલની અંદર. તે સ્ટ્રિંગના અંતે નવી લાઈન ઉમેરતું નથી. તે ફોર્મેટિંગ અથવા લાઈન બ્રેક વિના સ્ટ્રિંગ્સ છાપવા માટે ઉપયોગી છે.

fputs() નું ઉદાહરણ

```
    char state[20] = "Gujarat";
    fputs(state, stdout);
```

આ કોડ સ્ક્રીન પર Gujarat પ્રિન્ટ કરશે.

ધ્યાન રાખો કે *fputs()* નો ઉપયોગ આપણા પ્રોગ્રામના આઉટપુટને ડિસ્ક પર સેવ કરેલી ફાઈલમાં લખવા માટે પણ થઈ શકે છે. જો કે, ફાઈલ મેનેજમેન્ટની કામગીરી અહીં આવરી લેવામાં આવી નથી, કારણ કે તે આ પ્રકરણના કાર્યક્ષેત્રની બહાર છે.

આઉટપુટ ફંક્શનમાં એસ્કેપ સિક્વન્સ (Escape Sequences) નો ઉપયોગ

C પ્રોગ્રામિંગમાં એસ્કેપ સિક્વન્સ (Escape Sequences) એ ખાસ પ્રકારના અક્ષર સંયોજનો છે, જેનો ઉપયોગ *printf()* જેવા આઉટપુટ ફંક્શનમાં આઉટપુટનું ફોર્મેટ નિયંત્રિત કરવા માટે થાય છે. તેનો ઉપયોગ ટેબ (Tab), નવી લાઈન (Newline) અને કેરેજ રિટર્ન (Carriage Return – લાઈનની શરૂઆત પર જવું) જેવા અદૃશ્ય (નોન-પ્રિન્ટેબલ) અક્ષરો દર્શાવવા માટે પણ થાય છે. દરેક એસ્કેપ સિક્વન્સ હંમેશા બેકસ્લેશ (\) થી શરૂ થાય છે અને તેની પાછળ એક અથવા વધુ અક્ષરો આવે છે.

જ્યારે આપણે આઉટપુટને એક નિશ્ચિત માળખામાં (structured text - સ્ટ્રક્ચર્ડ ટેક્સ્ટ) પ્રિન્ટ કરવું હોય, ડેટા ગોઠવવું (align data) હોય અથવા સ્પેશિયલ અક્ષરો (special characters) બતાવવા હોય ત્યારે એસ્કેપ સિક્વન્સ ખૂબ ઉપયોગી સાબિત થાય છે. કલ્પના કરો કે આપણે આપણી સ્ટ્રિંગમાં નવી લાઈન, ટેબ સ્પેસ અથવા ડબલ ક્વોટ (“) પ્રિન્ટ કરવા માંગીએ છીએ. આપણે આ અક્ષરો સીધા ટાઈપ કરી શકતા નથી, કારણ કે C ભાષામાં તેમના ખાસ અર્થ હોય છે. એસ્કેપ સિક્વન્સ આપણને આ અક્ષરોના સામાન્ય અર્થમાંથી “પલાયન” (escape) કરવાની તક આપે છે — એટલે કે, આપણે આ અક્ષરોને શાબ્દિક (literal) રૂપમાં પ્રિન્ટ કરી શકીએ અથવા ચોક્કસ ક્રિયા કરાવી શકીએ. ટેબલ 9.3 સૌથી વધુ ઉપયોગમાં લેવાતા એસ્કેપ સિક્વન્સ અને તેમના અર્થ દર્શાવે છે.

એસ્કેપ સિક્વન્સ	અર્થ
\n	**Newline :** કર્સરને નવી લાઈનની શરૂઆતમાં ખસેડે છે.
\t	**Horizontal Tab :** ટેબ સ્પેસ દાખલ કરે છે.
\b	**Backspace :** કર્સરને એક જગ્યા પાછળ ખસેડે છે.
\r	**Carriage Return (start of line) :** કર્સરને તે જ લાઈનની શરૂઆતમાં ખસેડે છે (જો એ લાઈનમાં અક્ષરો હશે તો તેના પર લખશે).
\\	**Backslash :** બેકસ્લેશ (\) પ્રિન્ટ કરશે.
\"	**Double Quote :** ડબલ ક્વોટ (“) પ્રિન્ટ કરશે.
\'	**Single Quote :** સિંગલ ક્વોટ (‘) પ્રિન્ટ કરશે.
\0	**Null Character :** સ્ટ્રીંગનો અંત દર્શાવે છે, અને *printf()* પ્રિન્ટ નહીં કરે.

ટેબલ 9.3 : એસ્કેપ સિક્વન્સ

ચાલો, એસ્કેપ સિક્વન્સનો ઉપયોગ નીચે આપેલા C પ્રોગ્રામ દ્વારા સમજીએ:

```
/* Program to illustrate use of escape sequence characters */

#include <stdio.h>
int main() {

   printf("Hello, World! \n");            /* Use \n for a new line*/
```

```
    printf("This is on a new line.\n\n");   /* Two newlines */

    printf("Name\t:\tChirag Patel\n");   /* \t for horizontal tab */
    printf("Age \t:\t30\n");

    /* Use \" to print double quoes */
    printf("Teacher said: \"Welcome Students!\"\n");

    /* Use \\ to print a backslash */
    printf("System path is C:\\Users\\Documents\n");

    /* Use \b for backspace. 3 will no be printed */
    printf("123\b456\n");

    /* Use of carriage return (Line 2 will overwrite Line 1) */
    printf("Line 1.\rLine 2.\n");

    /* Combining multiple escape sequences */
    printf("\n\t--- Program Output ---\n");
    return 0;
}

Result:

Hello, World!
This is on a new line.

Name    :       Chirag Patel
Age     :       30
Teacher said: "Welcome Students!"
System path is C:\Users\Documents
12456
Line 2.

        --- Program Output ---
```

આગળના કોડમાં, `printf("Line 1.\rLine 2.\n");` વિધાન રસપ્રદ છે. આ પ્રોગ્રામ પ્રથમ "Line 1." પ્રિન્ટ કરે છે. ત્યારબાદ \r (કેરેજ રિટર્ન)ના કારણે કર્સર હાલની લાઈનની શરૂઆતમાં પાછો જાય છે. પછી "Line 2." પ્રિન્ટ થાય છે, જે અગાઉના "Line 1." ને ભૂંસીને તેના પર લખે છે.

અંતમાં, \n કર્સરને નવી લાઈન પર લઈ જાય છે, અને પ્રોગ્રામનો બાકીનો ભાગ આગળ ચલાવવામાં આવે છે.

ફોર્મેટેડ ઈનપુટ/આઉટપુટ ફંક્શનો (Formatted I/O Functions)

અગાઉના વિભાગોમાં આપણે C પ્રોગ્રામિંગમાં સામાન્ય રીતે ઉપયોગમાં લેવાતા I/O ફંક્શનોનો અભ્યાસ કર્યો છે. ફોર્મેટેડ ઈનપુટ/આઉટપુટ ફંક્શનો એ એવા ફંક્શનો છે જે આપણને ડેટાને નિર્ધારિત, વાંચવા યોગ્ય ફોર્મેટમાં વાંચવા અને લખવાની સવલત આપે છે.

printf() અને *scanf()* જેવા ફોર્મેટેડ I/O ફંક્શનોનો ઉપયોગ નિર્ધારિત ફોર્મેટમાં ડેટા દર્શાવવા અને વાંચવા માટે થાય છે. આ ફંક્શનો ફોર્મેટ સ્પેસિફાયર (Format Specifiers) અને કંટ્રોલ સ્ટ્રિંગ (Control String) નો ઉપયોગ કરીને ડેટા કેવી રીતે દર્શાવવો કે સ્વીકારવો તે નિયંત્રિત કરે છે.

ફોર્મેટ સ્પેસિફાયર એ ખાસ પ્રકારની ક્રમબદ્ધ અક્ષર શ્રેણી છે, જે પ્રતિશત ચિહ્ન (%) થી શરૂ થાય છે અને તેની પાછળ એક અથવા વધુ અક્ષરો આવે છે. આ અક્ષરો જણાવે છે કે કયા પ્રકારનો ડેટા વાંચવો કે લખવો છે, તેમજ વૈકલ્પિક રીતે ફોર્મેટિંગ માટેની સૂચનાઓ પણ આપી શકાય છે.

કંટ્રોલ સ્ટ્રિંગ એ I/O ફંક્શનનો પહેલો પેરામીટર હોય છે. તે એક સ્ટ્રિંગ શાબ્દિક (લિટરલ) હોય છે, જેમાં સામાન્ય અક્ષરો (જે એ રીતે જ પ્રિન્ટ/સ્કેન થાય છે) અને એક કે એકથી વધુ ફોર્મેટ સ્પેસિફાયર સામેલ હોય છે.

ફોર્મેટ સ્પેસિફાયર્સ (Format specifiers)ના ચાવીરૂપ પાસાઓ

તેની ચર્ચા નીચે પ્રમાણે છે :

- **ડેટા ટાઈપ સૂચન (Data Type Indication) :** ફોર્મેટ સ્પેસિફાયરનું મુખ્ય કાર્ય I/O ફંક્શનને ઈનપુટ માટેના સંભવિત ડેટા ટાઈપ વિશે જણાવવાનું છે, અથવા આઉટપુટ માટે કયા પ્રકારનો ડેટા પ્રિન્ટ થવાનો છે તે સૂચવવાનું છે. ટેબલ 9.4 માં આપેલા વિવિધ ફોર્મેટ સ્પેસિફાયર્સનો ઉપયોગ કરીને, આપણે I/O ફંક્શનો દ્વારા અલગ-અલગ પ્રકારના ડેટાને અસરકારક રીતે સંભાળી શકીએ છીએ.

ફોર્મેટ સ્પેસિફાયર	ઉપયોગ/વર્ણન
`%d`	પૂર્ણાંક (decimal - *int*)
`%f`	અપૂર્ણાંક સંખ્યા (*float*)
`%lf`	ડબલ- પ્રિસિઝન પૂર્ણાંક (*double*)
`%c`	એક અક્ષર (*char*)
`%s`	સ્ટ્રિંગ (અક્ષરોની હરોળ)
`%u`	ધન પૂર્ણાંક (Unsigned integer)
`%x`	હેક્ષાડેસીમલ પૂર્ણાંક (lowercase)
`%X`	હેક્ષાડેસીમલ પૂર્ણાંક (uppercase)
`%o`	ઓક્ટલ નંબર
`%ld`	લાંબો પૂર્ણાંક (Long integer)
`%lu`	ધન લાંબો પૂર્ણાંક (Unsigned long integer)
`%p`	પોઈન્ટર એડ્રેસ (Pointer address)

ટેબલ 9.4 : ફોર્મેટ સ્પેસિફાયર

- **ફોર્મેટિંગ વિકલ્પો (Formatting Options - Flags, Width, Precision) :** *printf()* અને *scanf()* ની કંટ્રોલ સ્ટ્રિંગમાં ફ્લેગ્સ (flags), પહોળાઈ (width), પ્રિસિઝન (precision) અને લેન્થ મોડિફાયર્સ (length modifiers) સામેલ હોઈ શકે છે, જેનાથી ડેટા કેવી રીતે ઈનપુટ લેવાશે અથવા પ્રિન્ટ થશે તે નિયંત્રિત કરી શકાય છે. નીચે ફોર્મેટ સ્પેસિફાયરનાં ઘટકો કેવી રીતે કાર્ય કરે છે તેની સામાન્ય ફોર્મેટ બતાવવામાં આવી છે :

```
%[flags][width][.precision]specifier
```

ફોર્મેટ સ્પેસિફાયર્સમાં આઉટપુટના દેખાવને નિયંત્રિત કરવા માટે વૈકલ્પિક ઘટકો (optional components) પણ હોઈ શકે છે:

- **Flags (ફ્લેગ્સ) :** આઉટપુટના વર્તનને બદલવા માટે તેનો ઉપયોગ થાય છે. ઉદાહરણ તરીકે, - લેફ્ટ-જસ્ટિફિકેશન માટે, + હંમેશા સાઈન બતાવવા માટે, 0 ઝીરો-પેડિંગ માટે.
- **Width (પહોળાઈ) :** આઉટપુટ માટે ઓછામાં ઓછી ફિલ્ડ પહોળાઈ નિર્ધારિત કરે છે. જો ડેટા ટૂંકો છે તો પેડિંગ થાય છે; જો લાંબો છે તો ટ્રંકેશન (truncate) નહીં થાય.
- **Precision (પ્રિસિઝન) :** ફ્લોટિંગ-પોઇન્ટ નંબર માટે, ડેસિમલ પોઈન્ટ પછીના અંકોની સંખ્યા નિયંત્રિત કરે છે. સ્ટ્રિંગ માટે, પ્રિન્ટ કરવાં માટે વધુમાં વધુ અક્ષરોની સંખ્યા નિર્ધારિત કરે છે.

- **ઈનપુટ માટે મેચિંગ (Matching for Input) :** જ્યારે *scanf()* જેવા ઈનપુટ ફંક્શન સાથે ઉપયોગ થાય છે, ત્યારે કંટ્રોલ સ્ટ્રિંગમાં ફોર્મેટ સ્પેસિફાયર્સને પેરામીટર લિસ્ટમાં રહેલા સંબંધિત ચલના ડેટા ટાઈપ સાથે ચોક્કસ મેચિંગ રાખવું જરૂરી છે. જો મેચ ન થાય, તો અણધાર્યું વર્તન અથવા ખોટો ડેટા પ્રિન્ટ થઈ શકે છે.

ફોર્મેટેડ (Formated) ઈનપુટ/આઉટપુટ ફંક્શનનું ઉદાહરણ

ચાલો ઉદાહરણ દ્વારા વિવિધ ફોર્મેટિંગ વિકલ્પોને સમજીએ.

```
/* Program to illustrate various formatting options. */

#include <stdio.h>
int main() {
    int num = 123;
    float price = 45.6789;
    char state[20] = "GUJARAT";

    printf("num = %5d \n", num);
    printf("num = %05d \n", num);

    printf("Price = %.3f \n", price);
    printf("Price = %8.2f \n", price);
    printf("Price = %-8.2f \n", price);

    printf("State = %9.3s \n", state);
    printf("State = %-9.3s \n", state);

    return 0;
}

Result:

num =   123
num = 00123
Price = 45.679
Price =    45.68
Price = 45.68
State =       GUJ
State = GUJ
```

ઉપરના કોડમાં ઉપયોગ કરાયેલા ફોર્મેટિંગ સ્પેસિફાયર્સની વિવિધ અસરનો ટેબલ 9.5માં સારાંશ આપેલો છે.

વિધાન	ફોર્મેટ સ્પેસિફાયર	આઉટપુટ	અસર
printf("%5d \n", num);	%5d	123	num ને 5-અક્ષર પહોળા ક્ષેત્રમાં જમણી બાજુએ પ્રિન્ટ કરશે
printf("%05d \n", num);	%05d	00123	5-અક્ષર પહોળા ક્ષેત્રમાં જમણી બાજુએ 0 પેડીંગ સાથે પ્રિન્ટ કરશે
printf("%.3f \n", price);	%.3f	45.679	3 ડેસીમલ ડીઝીટ સાથે જમણી બાજુએ પ્રિન્ટ કરશે
printf("%8.2f \n", price);	%8.2f	45.68	કુલ પહોળાઈ 8, જેમાં 2 ડેસીમલ ડીઝીટ અને જમણી બાજુએ પ્રિન્ટ કરશે
printf("-8.2f \n", price);	%-8.2f	45.68	કુલ પહોળાઈ 8, જેમાં 2 ડેસીમલ ડીઝીટ અને ડાબી બાજુએ પ્રિન્ટ કરશે
printf("%9.3s \n", state);	%9.3s	GUJ	કુલ 9 અક્ષરની પહોળાઈમાં પહેલાં 3 અક્ષરો જમણી બાજુએ પ્રિન્ટ કરશે
printf("%-9.3s \n", state);	%-9.3s	GUJ	કુલ 9 અક્ષરોની પહોળાઈમાં પહેલાં 3 અક્ષરો ડાબી બાજુએ પ્રિન્ટ કરશે

ટેબલ 9.5 : ફોર્મેટ સ્પેસિફાયર્સ અને તેની અસરનો સારાંશ

સારાંશ

આ પ્રકરણમાં, આપણે C પ્રોગ્રામિંગમાં ઈનપુટ અને આઉટપુટ ઓપરેશનના મૂળભૂત ખ્યાલોનો અભ્યાસ કર્યો. આપણે શીખ્યા કે કેવી રીતે સ્ટાન્ડર્ડ લાઈબ્રેરી <stdio.h> કીબોર્ડમાંથી ઈનપુટ વાંચવા અને સ્ક્રીન પર આઉટપુટ દર્શાવવા સક્ષમ બનાવે છે. મુખ્ય ઈનપુટ ફંક્શનો જેમ કે scanf(), getchar() અને fgets() યુઝરને વિવિધ પ્રકારના ડેટા દાખલ કરવા માટે વપરાય છે, જ્યારે આઉટપુટ ફંક્શનો જેમ કે printf(), putchar(), puts() અને fputs() મૂલ્યો યોગ્ય ફોર્મેટમાં આઉટપુટ દર્શાવવામાં મદદ કરે છે. આપણે %d, %f, %c, %s જેવા ફોર્મેટ સ્પેસિફાયર્સ અને ફીલ્ડ પહોળાઈ, પ્રિસિઝન, અને એલાઈનમેન્ટ જેવા ફોર્મેટિંગ નિયંત્રણોની ભૂમિકા પણ શીખી. પ્રકરણ દરમિયાન આપણે વિવિધ એસ્કેપ સિક્વેન્સ (જેમ કે \n, \t, અને \\) શીખ્યા, જે આપણી ફોર્મેટિંગ જરૂરિયાત મુજબ આઉટપુટ બતાવવા ખૂબ ઉપયોગી છે. પ્રકરણમાં આવરી લેવામાં આવેલા વિવિધ I/O ફંક્શનો અને ઓપરેશનો C પ્રોગ્રામમાં યુઝર સાથે સંવાદ કરવાની માળખાકીય ભૂમિકા રચે છે, જે તેને વાસ્તવિક જગતની એપ્લિકેશન માટે અનિવાર્ય બનાવે છે.

સ્વાધ્યાય

1. પ્રોગ્રામિંગમાં I/O ઓપરેશનનો અર્થ શું છે?
2. C પ્રોગ્રામિંગમાં ફોર્મેટ સ્પેસિફાયર્સનો ઉપયોગ જણાવો. scanf() અને printf() માં ફોર્મેટ સ્પેસિફાયર્સ શું છે? %d, %f, %c, અને %s માટે ઉદાહરણ આપો.
3. કીબોર્ડનો ઉપયોગ કરીને યુઝર પાસેથી ઈનપુટ વાંચવા માટે C માં ઉપયોગમાં લેવાતા કોઈ ત્રણ લોકપ્રિય ફંક્શનો દર્શાવો.

4. C પ્રોગ્રામિંગમાં એવા ઈનપુટ ફંક્શનનું નામ જણાવો, જે યુઝરને સંપૂર્ણ ટેક્સ્ટ-લાઈન દાખલ કરવાની મંજૂરી આપે, જેમાં શબ્દોની વચ્ચે ખાલી જગ્યા પણ હોય. જ્યારે પૂર્ણ વાક્યો અથવા બહુ-શબ્દ ઈનપુટ સાથે કામ કરતા હોય ત્યારે તે scanf() કરતાં કેમ વધારે યોગ્ય છે તે સમજાવો. જરૂરી હેડર ફાઈલ અને આ ફંક્શન ઉપયોગ કરતી વખતે કોઈ સલામતી બાબતનો ખ્યાલ રાખવો પડશે કે કેમ તે બાબતે જણાવો.

5. આપેલા એસ્કેપ સિક્વેન્સ અક્ષરોનો ઉપયોગ યોગ્ય ઉદાહરણ સાથે જણાવો,: \n, \t અને \b.

6. printf() અને scanf() ફંક્શનનો ઉપયોગ યોગ્ય C પ્રોગ્રામ સાથે સમજાવો.

7. લોકપ્રિય ઈનપુટ અને આઉટપુટ ડિવાઈસના કોઈ ત્રણ નામ યાદીબદ્ધ કરો.

8. C પ્રોગ્રામિંગમાં <stdio.h> ની ભૂમિકા અંગે સંક્ષિપ્ત નોંધ લખો.

9. printf() અને putchar() ફંક્શન વચ્ચેનો તફાવત જણાવો.

10. સંખ્યાઓને ગોઠવવા કે ડેસીમલ પ્રિસિઝન નિયંત્રિત કરવા printf() ના ઉપયોગથી આઉટપુટ કેવી રીતે ફોર્મેટ કરી શકાય? %5d, %2f નો ઉપયોગ દર્શાવતા ઉદાહરણ આપો.

11. સાચું કે ખોટું જણાવો.

(1) scanf() ફંક્શનનો ઉપયોગ સ્ક્રીન પર ફોર્મેટેડ આઉટપુટ દર્શાવવા માટે થાય છે.

(2) એસ્કેપ સિક્વન્સ \t નો ઉપયોગ આઉટપુટમાં નવી લાઈન દાખલ કરવા માટે થાય છે.

(3) getchar() ફંક્શનનો ઉપયોગ કીબોર્ડમાંથી એક જ અક્ષરનો ઈનપુટ વાંચવા માટે થાય છે.

(4) gets() ફંક્શનનો ઉપયોગ ખાલી જગ્યા સહિતની સંપૂર્ણ લાઈન યુઝર ઈનપુટ તરીકે વાંચવા માટે કરી શકાય છે.

(5) puts() અને fputs() જેવા ફંક્શનનો ઉપયોગ C માં ઈનપુટ ઓપરેશન માટે થાય છે.

12. ખાલી જગ્યાઓ પૂરો.

(1) ફોર્મેટ સ્પેસિફાયર %f નો ઉપયોગ ____________ ડેટા ટાઈપ (મૂલ્ય) પ્રિન્ટ કરવા માટે થાય છે.

(2) printf() નો ઉપયોગ કરીને અક્ષરોની સ્ટ્રિંગ પ્રિન્ટ કરવા માટે, ફોર્મેટ સ્પેસિફાયર ____________ નો ઉપયોગ કરીએ છીએ.

(3) એસ્કેપ સિક્વન્સ \n નો ઉપયોગ આઉટપુટમાં ____________ દાખલ કરવા માટે થાય છે.

(4) ડબલ ક્વોટ ચિહ્ન (") પ્રિન્ટ કરવા માટે, સ્ટ્રિંગની અંદર એસ્કેપ સિક્વન્સ ____________ નો ઉપયોગ કરીએ છીએ.

(5) ____________ ઈનપુટ ફંક્શન C પ્રોગ્રામિંગમાં કન્સોલમાંથી પૂર્ણ લાઈન (બહુ-શબ્દ) વાંચવા માટે ઉપયોગ કરી શકાય છે.

13. બહુવિકલ્પી પ્રશ્નો. સૌથી યોગ્ય જવાબ પસંદ કરો.

(1) નીચેનામાંથી કયું ફંક્શન એક જ અક્ષર પ્રિન્ટ કરવા માટે વપરાશે?

(a) puts() (b) fputs() (c) printf() (d) putchar()

(2) નીચેનાનું આઉટપુટ શું હશે?

```
printf("Welcome %d", 2025);
```

(a) Welcome (b) Welcome %d

(c) Welcome 2025 (d) Welcome %d, 2025

(3) નીચેના C વિધાનોનું આઉટપુટ શું હશે?

```
putchar('A');
putchar('\n');
```

(a) Nothing (b) A

(c) 'A\n' (d) A પછી નવી લાઈન

(4) નીચેનામાંથી કયું વિધાન Hello "World" પ્રિન્ટ કરશે?

(a) printf("Hello "World"); (b) printf("Hello \"World\"");

(c) printf('Hello "World"'); (d) printf("Hello World);

(5) C માં કેરેક્ટર ચલને પ્રિન્ટ કરવા માટેનું સાચું ફોર્મેટ સ્પેસિફાયર કયું છે?

(a) %ch (b) %c (c) %s (d) %char

(6) નીચેના વિધાનનું આઉટપુટ ધારો:

```
printf("ABC\bD");
```

(a) ABCD (b) ABD (c) ABC\bD (d) AB CD

(7) એસ્કેપ સિક્વન્સ \t નો હેતુ શું છે?

(a) newline દાખલ કરવાનો (b) space દાખલ કરવાનો

(c) tab space દાખલ કરવાનો (d) અક્ષરને ભૂંસવાનો

(8) નીચેના C પ્રોગ્રામનો આઉટપુટ શું હશે?

```
#include <stdio.h>
int main() {
      printf("C:\\Program Files\\");
    return 0;
}
```

(a) C:\Program Files\ (b) C:Program Files

(c) C://Program Files// (d) Error

(9) નીચેના C પ્રોગ્રામનો આઉટપુટ શું હશે?

```
#include <stdio.h>
int main() {
    char str[ ] = "End\0Start";
    printf("%s", str);
    return 0;
}
```

(a) End (b) End\0Start (c) Start (d) End\0

(10) એસ્કેપ સિક્વન્સ \0 અર્થ શો છે?

(a) 0 પ્રિન્ટ કરે છે (b) સ્ટ્રીંગનો અંત દર્શાવે છે

(c) ખાલી જગ્યા દાખલ કરે છે (d) કઈ જ પ્રિન્ટ કરતુ નથી

પ્રાયોગિક સ્વાધ્યાય

1. એક એવો C પ્રોગ્રામ લખો જે કીબોર્ડમાંથી એક અક્ષર વાંચે અને તેને putchar() ફંક્શનનો ઉપયોગ કરીને પ્રિન્ટ કરે.

2. એક એવો C પ્રોગ્રામ લખો જે યુઝરનો રોલ નંબર અને ટકા વાંચે અને પ્રદર્શિત કરે. તમારો પ્રોગ્રામ અમલ કરવા માટે int અને float ચલોનો ઉપયોગ કરો.

3. એક એવો C પ્રોગ્રામ લખો જે વિદ્યાર્થીને તેમનું સંપૂર્ણ સરનામું ટાઈપ કરવા માટે પૂછશે, જેમાં સોસાયટીનું નામ, શહેર, રાજ્ય અને પિનકોડ સામેલ હોય. જરૂરી અલગ ચલો ઘોષિત કરો. છેલ્લે, તે સરનામું સ્ક્રીન પર પ્રિન્ટ કરે, અને વાંચન સુવિધા માટે \t (ટેબ) અને \n (નવી લાઈન) કેરેક્ટરનો ઉપયોગ કરીને તેને અનેક લાઈનોમાં વિભાજિત કરે.

4. એક એવો C પ્રોગ્રામ લખો જે વ્યક્તિનું પ્રથમ નામ અને છેલ્લું નામ બે અલગ scanf() ફંક્શનનો ઉપયોગ કરીને વાંચે, અને પછી એક જ printf() ફંક્શનનો ઉપયોગ કરીને પૂર્ણ નામ પ્રિન્ટ કરે.

5. એક એવો C પ્રોગ્રામ લખો જે વ્યક્તિનું નામ (સ્ટ્રિંગ), ઉંમર (પૂર્ણાંક), ઊંચાઈ (અપૂર્ણાંક), અને શહેર (સ્ટ્રિંગ) માટે ચલ ઘોષણા કરે. આ ચલોને કેટલીક નમૂનાની માહિતી સાથે આરંભ કરો. પછી, printf() નો ઉપયોગ યોગ્ય ફોર્મેટ સ્પેસિફાયર્સ (જેમ કે %s, %d, %.2f) ઉપયોગ કરીને આ માહિતી સુવ્યવસ્થિત અને વાંચનીય રીતે પ્રદર્શિત કરો, ઉદાહરણ તરીકે: "Name: [Name], Age: [Age] years, Height: [Height]m, City: [City]."

6. એક એવો C પ્રોગ્રામ લખો જે યુઝર પાસેથી લગભગ 100 અક્ષરોનો પેરાગ્રાફ વાંચે. યુઝર દ્વારા દાખલ કરેલી સામગ્રી વાંચવા અને પ્રિન્ટ કરવા માટે યોગ્ય ઈનપુટ અને આઉટપુટ ફંક્શનનો ઉપયોગ કરો.

7. એક એવો C પ્રોગ્રામ લખો જે લંબાઈ અને પહોળાઈ ઈનપુટ તરીકે વાંચે. આ લંબચોરસ આકારનું ક્ષેત્રફળ (લંબાઈ × પહોળાઈ) અને પરિઘ (2 × (લંબાઈ + પહોળાઈ)) ગણાવીને પ્રદર્શિત કરો. જવાબ નીચે દર્શાવેલા ફોર્મેટમાં પ્રસ્તુત કરો:

```
Length entered: 10.5 units
Breadth entered: 5.0 units

Calculated Area: 52.50 square units
Calculated Perimeter: 31.00 units
```

8. એક એવો C પ્રોગ્રામ લખો જે યુઝર પાસેથી તાપમાન સેલ્સિયસમાં ઈનપુટ તરીકે વાંચે અને તેને નીચેના સૂત્રનો ઉપયોગ કરીને ફેરનહીટમાં રૂપાંતરિત કરે:

```
F = (C × 9/5) + 32
```

અહીં

F: ફેરનહીટમાં તાપમાન દર્શાવે છે.

C: સેલ્સિયસમાં તાપમાન દર્શાવે છે.

પરિણામ %.2f નો ઉપયોગ કરીને બે ડેસીમલ સ્થાન સાથે પ્રદર્શિત કરો.

9. નીચેનો C પ્રોગ્રામ વિદ્યાર્થીનું નામ વાંચે અને પ્રિન્ટ કરે છે

```
#include <stdio.h>
int main() {
    char name[30];
    printf("Enter your name: ");
    scanf("%s", name);

    printf("Your name is: ");
    puts(name);
    return 0;
}
```

પરંતુ જ્યારે આપણે કોડનો અમલ કરીએ અને નામ તરીકે "Jia Patel" દાખલ કરીએ, ત્યારે તે માત્ર પ્રથમ નામ "Jia" પ્રિન્ટ કરે છે. આ કોડથી સંપૂર્ણ નામ પ્રિન્ટ ન કરી શકવાનું કારણ ઓળખો.

10. એવો C પ્રોગ્રામ લખો જે મૂળ રકમ (P), વ્યાજ દર (R) અને વર્ષોની સંખ્યા (N) ના આપેલા મૂલ્યો આધારે સાદુ વ્યાજ ગણે અને પ્રિન્ટ કરે. સાદું વ્યાજ શોધવા માટે નીચેનું સૂત્ર વાપરોઃ

```
SI = P*R*N/100
```

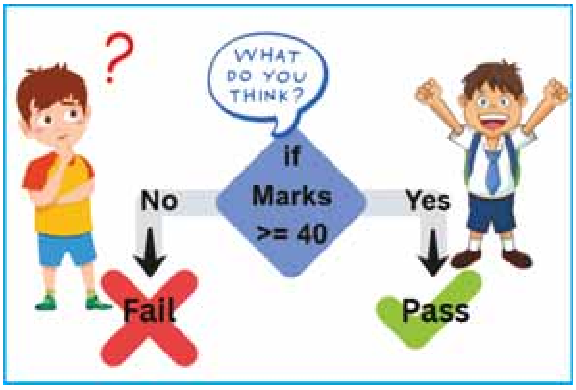
નિર્ણય લેવાની નિયંત્રણ સંરચનાઓ

પરિચય

અગાઉના પ્રકરણમાં, આપણે C પ્રોગ્રામિંગના ઈનપુટ/આઉટપુટ ફંક્શનના ઉપયોગ વિશે ચર્ચા કરી છે. આ પ્રકરણ C પ્રોગ્રામિંગમાં નિર્ણય લેવાની નિયંત્રણ સંરચનાઓ (Decision Making Control Structures) પર ધ્યાન કેન્દ્રિત કરે છે. અત્યાર સુધી ચર્ચા કરાયેલા મોટાભાગના C પ્રોગ્રામ અમલ માટે ક્રમિક સંરચનાને અનુસરે છે. આનો અર્થ એ છે કે પ્રોગ્રામમાં જે ક્રમમાં વિધાનો (statements) દેખાય છે, તે જ ક્રમમાં તે તેનો અમલ (execute) કરવામાં આવે છે. પરંતુ, વાસ્તવિક જીવનના પ્રોગ્રામમાં, આપણે કદાચ અમુક શરતોના આધારે, આપણા પ્રોગ્રામના ફક્ત કેટલાક જ વિધાનોનો જ અમલ કરવા માગીએ છીએ. આ હેતુ માટે, સૂચનાઓના અમલના ક્રમના પ્રવાહને બદલવો જરૂરી છે. જેના માટે C ભાષા ખાસ વિધાનો પૂરા પાડે છે, જે પ્રોગ્રામની અંદરની સૂચનાઓના પ્રવાહમાં ફેરફાર કરવા સક્ષમ છે. આ વિધાનોને નિર્ણય અને નિયંત્રણ સંરચના વિધાનો તરીકે ઓળખવામાં આવે છે.

નિર્ણય લેવાની સંરચનાઓ (Decision Making Control Structures)

નિર્ણય લેવાની સંરચનાઓ કોઈપણ પ્રોગ્રામિંગ ભાષામાં પાયાના નિર્માણ ઘટકોમાં આવે છે. તે પ્રોગ્રામને અમુક શરતોના આધારે પસંદગીઓ કરવા, કાર્યોનું પુનરાવર્તન કરવા અને આપણે જે અમલ કરવા માંગીએ છીએ તેના મુજબ અમલના પ્રવાહને નિયંત્રિત કરવા દે છે.



આકૃતિ 10.1 : પ્રોગ્રામમાં નિર્ણય લેવાની સ્થિતિ

નિર્ણય સંરચનાઓ (Decision Structures)

પ્રોગ્રામિંગમાં નિર્ણય સંરચનાઓને પસંદગી સંરચનાઓ (Selection structures) તરીકે પણ ઓળખવામાં આવે છે. નિર્ણય સંરચનાઓનો મુખ્ય ઉદ્દેશ્ય પ્રોગ્રામના અમલ દરમિયાન નિર્ણય લેવામાં મદદ કરવાનો છે. જ્યારે કોઈ ચોક્કસ શરત સાચી કે ખોટી છે, તેના મૂલ્યાંકનના આધારે પ્રોગ્રામ જુદા જુદા વિધાનોનો અમલ કરે છે. જુઓ આકૃતિ 10.1.

ઉદાહરણ : નીચેના કોડમાં *printf()* ("Pass" અથવા "Fail" માંથી કોઈપણ એક) નો અમલ marks ચલમાં સંગ્રહિત મૂલ્ય પર આધારિત છે.

```
int marks = 50;
if (marks >= 40)
    printf("Pass");
else
    printf("Fail");
```

C પ્રોગ્રામિંગમાં સામાન્ય નિર્ણય સંરચનાઓ નીચે મુજબ છે:

- *if* વિધાન
    - if-else વિધાન
    - else-if ladder
- switch વિધાન

નિયંત્રણ સંરચનાઓ (Control Structures)

નિયંત્રણ સંરચનાઓ એ પ્રોગ્રામિંગના એવા ખ્યાલો છે જે પ્રોગ્રામમાં આપેલા વિવિધ વિધાનોના અમલ પ્રવાહને નિયંત્રિત કરે છે. બધી જ નિર્ણય સંરચનાઓ (જેવી કે *if, if-else, switch*) એ નિયંત્રણ સંરચનાઓના પ્રકાર જ છે. ટેબલ 10.1 વિવિધ નિયંત્રણ સંરચનાઓના પ્રકારો દર્શાવે છે.

પ્રકાર	વર્ણન
ક્રમિક (Sequential)	મૂળભૂત પ્રવાહ (Default flow); પ્રોગ્રામના બધા વિધાનોનો એક પછી એક એમ ક્રમશઃ અમલ થાય છે.
નિર્ણય લેનાર (Decision-making)	*if, if-else, switch* વગેરે વિધાનોનો ઉપયોગ કરીને શરતના આધારે કોડના ભાગનો અમલ કરે છે.
લૂપિંગ (Looping) (પુનરાવર્તન/ Itertaion)	*for, while, do-while* જેવા વિધાનોનો ઉપયોગ કરીને જ્યાં સુધી શરત સાચી હોય ત્યાં સુધી કોડનું પુનરાવર્તન કરે છે.
જમ્પિંગ (નિયંત્રણનું સ્થાનાંતરણ/transfer of control)	*break, contiume, goto* અને *return* જેવા વિધાનોનો ઉપયોગ કરીને પ્રોગ્રામના અન્ય ભાગમાં નિયંત્રણનું સ્થાનાંતરણ કરે છે.

ટેબલ 10.1 : નિયંત્રણ સંરચનાઓના પ્રકારો

ચાલો આપણે દરેક પ્રકારની નિયંત્રણ સંરચનાઓ ઉદાહરણ સાથે સમજીએ.

ક્રમિક નિયંત્રણ સંરચનાઓ (Sequential Control Structures)

C પ્રોગ્રામિંગમાં, ક્રમિક નિયંત્રણ સંરચનાનો અર્થ એ છે કે પ્રોગ્રામ લખેલા વિધાનોનો સમાન ક્રમમાં, પહેલાં પછી બીજું, બીજા પછી ત્રીજું, એમ ક્રમમાં અમલ થાય છે. ક્રમિક નિયંત્રણ સંરચના સૌથી સરળ નિયંત્રણ સંરચના છે. તે પ્રોગ્રામના કોડનો ઉપરથી નીચે સુધી ક્રમશઃ અમલ કરે છે. પ્રોગ્રામમાં કોઈ નિર્ણય અથવા પુનરાવર્તિત ક્રિયાઓ જરૂરી ન હોય તેવા સરળ પ્રોગ્રામ લખવા તે ઉપયોગી છે.

ક્રમિક નિયંત્રણ સંરચનાના મુખ્ય લક્ષણો

- **ક્રમિક અમલ:** પહેલા વિધાન પછી બીજું વિધાન એમ ક્રમશઃ ચલાવવામાં આવે છે.
- **વાંચવા અને સમજવામાં સરળ:** આવા પ્રોગ્રામો વાંચવા અને સમજવા માટે સરળ હોય છે.
- **શરતો અથવા લૂપનો ઉપયોગ નહીં:** શરતી વિધાનો (જેમ કે *if, if-else*) અથવા લૂપીંગ (*for, while*) નો ઉપયોગ થતો નથી.
- **સરળ કાર્યો માટે ઉપયોગ:** ઇનપુટ, આઉટપુટ અથવા ગણતરીઓ જેવા સરળ કાર્યો માટે ઉપયોગ થાય છે.

ક્રમિક નિયંત્રણ સંરચનાનું ઉદાહરણ

```
/* Program to illustrate use of Sequential Control Structure */
#include <stdio.h>
int main() {
    int a, b, sum;

    printf("Enter two numbers: ");
```

```
    scanf("%d %d", &a, &b);

    sum = a + b;

    printf("Sum = %d\n", sum);
    return 0;
}
Result:

Enter two numbers: 5 6
Sum = 11
```

આ પ્રોગ્રામમાં મુખ્ય તર્ક (logic) એ ત્રણ *int* પ્રકારના ચલ (variables) જાહેર કરવાનો, પછી યુઝર પાસેથી ઈનપુટ તરીકે બે સંખ્યાઓ વાંચવાનો, તેનો સરવાળો કરવાનો અને પછી પરિણામ દર્શાવવાનો છે. ધ્યાન રાખો કે આ પ્રોગ્રામમાં, તમામ વિધાનો એક સ્પષ્ટ ક્રમને અનુસરીને એક પછી એક અમલમાં આવે છે. ચલોની ઘોષણા (declaration), ઈનપુટ માટે પૂછવું, ઈનપુટ વાંચવું, સરવાળો કરવો અને સરવાળો છાપવો. તેથી, આવા પ્રોગ્રામને ક્રમિક નિયંત્રણ સંરચના પ્રોગ્રામ તરીકે વર્ગીકૃત કરવામાં આવે છે

નિર્ણય લેવાના વિધાનો (Decision Making Statements)

C પ્રોગ્રામિંગમાં નિર્ણય લેવાના (પસંદગીના) વિધાનો પ્રોગ્રામને ચોક્કસ શરતોના આધારે પસંદગી કરવાની મંજૂરી આપે છે. આ વિધાનો પ્રોગ્રામને શરત સાચી કે ખોટી છે તેના આધારે કોડનો કયો બ્લોક ચલાવવો તે નક્કી કરવામાં મદદ કરે છે. સામાન્ય નિર્ણય લેવાના વિધાનોમાં *if, if-else,* નેસ્ટેડ *if* (nested *if*), અને *switch* નો સમાવેશ થાય છે. તે પ્રોગ્રામના પ્રવાહને નિયંત્રિત કરવા અને આપણી જરૂરિયાત મુજબની વિવિધ પરિસ્થિતિઓને સંભાળવા માટે ઉપયોગી છે. નિર્ણય લેવાના વિધાનોને પસંદગીના વિધાનો (Selection statements) અથવા બ્રાન્ચિંગ વિધાનો (Branching statements) તરીકે પણ ઓળખવામાં આવે છે.

ચાલો, નિર્ણય લેવાના દરેક પ્રકારના વિધાનને ઉદાહરણો સાથે સમજીએ.

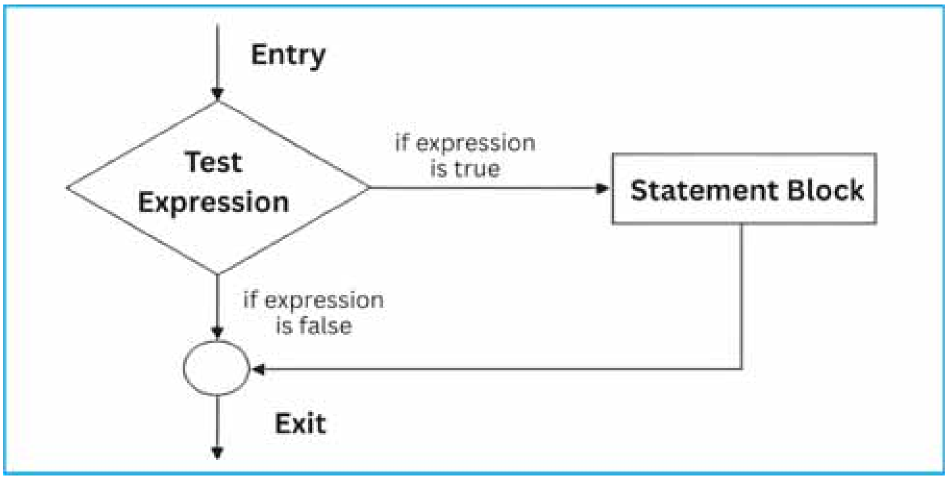
if વિધાન (if statement)

if વિધાન C પ્રોગ્રામિંગમાં સૌથી વધુ ઉપયોગમાં લેવાતું અને સરળ નિર્ણય લેવાનું વિધાન છે. *if* વિધાનની વાક્યરચના (Syntax) નીચે મુજબ છે:

```
if(test expression) {
  Statement block of code;
}
```

if વિધાન કૌંસ ( ) ની અંદરની પદાવલી (test expression) ને તપાસે છે. જો તે સાચું હોય, તો છગડિયા કૌંસ { } ની અંદરનો કોડ અમલ કરવામાં આવે છે. જો શરતી વિધાન ખોટું હોય તો બ્લોકની અંદરનો કોડ અવગણવામાં આવે છે અને પ્રોગ્રામની આગામી સૂચનાનો અમલ કરે છે.

અહીં, ચકાસણી માટેની પદાવલી C ભાષાની કોઈપણ માન્ય પદાવલી (એક્સપ્રેશન) હોઈ શકે છે. જો *if* બ્લોકમાં એક જ વિધાન (statement) હોય તો, { } વૈકલ્પિક છે. { } ની અંદરના કોડના બ્લોકને સ્ટેટમેન્ટ બ્લોક (statement block) કહેવામાં આવે છે. સ્ટેટમેન્ટ બ્લોકમાં C ભાષાનું કોઈપણ માન્ય વિધાન અથવા વિધાનો હોઈ શકે છે. નોંધ લો કે ( ) અંદરના શરતી વિધાન (test expression)નો ભાગ અર્ધવિરામથી સમાપ્ત થવો ન જોઈએ. આકૃતિ 10.2 સાદા *if* વિધાનનો ફ્લોચાર્ટ દર્શાવે છે.



આકૃતિ 10.2 : *if* વિધાનનો ફ્લોચાર્ટ

નોંધ: C પ્રોગ્રામિંગ ભાષા શૂન્ય સિવાયની અને નલ (null) સિવાયની કોઈપણ કિંમતને સાચી માને છે. તેથી, સૂત્ર (5 + 5) ને સાચું ગણવામાં આવે છે. તેવી જ રીતે, શૂન્ય અથવા નલ (null) કિંમતને ખોટી માનવામાં આવે છે. તેથી, જે શરત કે સૂત્રનું મૂલ્યાંકન શૂન્ય થાય છે (દા.ત., 5 - 5), તેને ખોટું ગણવામાં આવે છે.

if વિધાનનું ઉદાહરણ

```
/* Program to illustrate use of if  statement. It will check
   whether a given number is positive or not using if  statement.
*/
#include<stdio.h>
int main() {
    int number;
    printf("Enter number:");
    scanf("%d",&number);

    if (number > 0)
        printf("Number is positive.");

    if (number <= 0)
        printf("Number is not positive.");
    return 0;
}
Result:

Enter number:5
Number is positive.
```

આ પ્રોગ્રામ આપેલ સંખ્યા ધન છે કે નહીં તે ચકાસવા માટે *if* વિધાનના ઉપયોગનું નિદર્શન કરે છે. તે એક પૂર્ણાંક (integer) ચલ number ઘોષિત કરે છે. તે પછીના *printf()* અને *scanf()* ફંક્શન લેબલ સાથે એક સંખ્યા વાંચવા માટે ઉપયોગ થાય છે. પ્રથમ *if* તપાસે છે કે દાખલ કરેલ સંખ્યા 0 કરતાં મોટી છે કે નહીં (number > 0); જો તે સાચી હોય, તો તે "Number is positive." ("સંખ્યા ધન છે.") એમ પ્રિન્ટ કરે છે. બીજી *if* શરત તપાસે છે કે સંખ્યા 0 કરતાં નાની અથવા 0 જેટલી છે કે નહીં (number <= 0); જો તે સાચી

હોય, તો તે "Number is not positive." ("સંખ્યા ધન નથી.") એમ પ્રિન્ટ કરે છે. આ પ્રોગ્રામ બે અલગ *if* વિધાનોનો ઉપયોગ કરે છે, તેથી તે બંને શરતોનું મૂલ્યાંકન સ્વતંત્ર રીતે કરે છે.

ચાલો સમજીએ કે આ જ પ્રોગ્રામને *if-else* વિધાનનો ઉપયોગ કરીને વધુ સરળ અને કાર્યક્ષમ રીતે કેવી રીતે લખી શકાય છે.

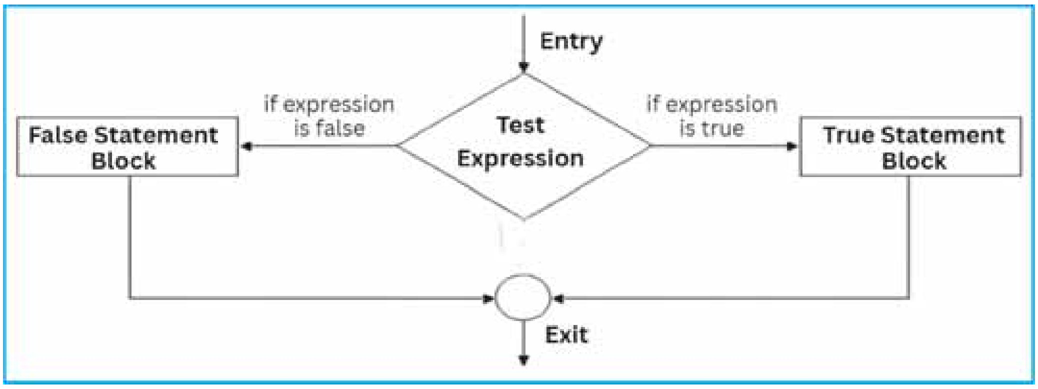
if-else વિધાન (if-else statement)

C પ્રોગ્રામિંગમાં *if-else* વિધાન એ એક પ્રકારનું નિર્ણય લેવાની નિયંત્રણ સંરચના છે, જેનો ઉપયોગ આપેલી પદાવલીના આધારે બે વિકલ્પોમાંથી એક કોડ બ્લોક ને અમલ કરવા માટે થાય છે. જો પદાવાલીમાંની શરત સાચી હોય તો *if* બ્લોકની અંદરનો કોડ અમલી થાય છે અન્યથા *else* બ્લોકની અંદરનો કોડ અમલી થાય છે. તે શરતી નિર્ણયોના આધારે પ્રોગ્રામના પ્રવાહને નિયંત્રિત કરવામાં મદદ કરે છે. આ વિધાનનો ઉપયોગ સામાન્ય રીતે તુલના, માન્યતા (validations) અને બ્રાન્ચિંગ (branching) જેવી શરતો તપાસવા માટે થાય છે.

if-else વિધાનની વાક્યરચના નીચે મુજબ છે:

```
if(test expression) {
    True statement-block of code;
}
else {
    False statement-block of code;
}
```

if વિધાન કૌંસ ( ) માં રહેલા ટેસ્ટ પદાવલી (test expression) ની તપાસ કરે છે. જો તે સાચી હોય તો છગડિયા કૌંસમાં આપેલો કોડ (True statement-block) અમલમાં મૂકાય છે. જો તે ખોટી હોય, તો *else* બ્લોકમાં આપેલો કોડ (False statement-block) અમલમાં મૂકાય છે. આકૃતિ 10.3 *if-else* વિધાનનો ફ્લોચાર્ટ દર્શાવે છે.



આકૃતિ 10.3 : *if-else* નો ફ્લોચાર્ટ

if-else વિધાનનું ઉદાહરણ

```
/* Program to check whether a given number is positive or not using
the if-else statement. */

#include<stdio.h>
int main() {
    int number;

    printf("Enter number:");
    scanf("%d",&number);
```

```
        if (number > 0)
            printf("Number is positive.");
        else
            printf("Number is not positive.");
        return 0;
    }

    Result:

    Enter number:-5
    Number is not positive.
```

આ પ્રોગ્રામ યુઝર પાસેથી એક પૂર્ણાંક ઈનપુટ વાંચે છે અને તે સંખ્યા ધન છે કે નહીં તે ચકાસવા માટે *if-else* વિધાનનો ઉપયોગ કરે છે. તે સૌપ્રથમ number નામનો એક પૂર્ણાંક ચલ ઘોષિત કરે છે અને પછી *printf()* નો ઉપયોગ કરીને યુઝરને મૂલ્ય દાખલ કરવા માટે પૂછે છે, ત્યારબાદ *scanf()* વડે ઈનપુટ વાંચે છે. *if* શરત તપાસે છે કે શું number 0 થી મોટો છે; જો તે સાચું હોય, તો તે "Number is positive." પ્રિન્ટ કરે છે. જો શરત ખોટી હોય (એટલે કે, સંખ્યા શૂન્ય અથવા ઋણ છે), તો *else* બ્લોક અમલમાં મુકાય છે અને "Number is not positive." પ્રિન્ટ કરે છે. આ નિયંત્રણ સંરચના ખાતરી કરે છે કે બે *printf()* માંથી ફક્ત એક જ અમલમાં મુકાય છે, જે અગાઉના *if* વિધાનના ઉદાહરણમાં બતાવ્યા પ્રમાણે બે અલગ અલગ *if* નિવેદનોનો ઉપયોગ કરવા કરતાં આઉટપુટને વધુ કાર્યક્ષમ અને વાંચવા યોગ્ય બનાવે છે.

નેસ્ટેડ-if વિધાન (Nested-if Statement)

નેસ્ટેડ-*if* વિધાન (Nested-*if* statement) એટલે એક *if* નિવેદન જે બીજા *if* અથવા *else* બ્લોક ની અંદર અસ્તિત્વ ધરાવે છે. આ માળખું જટિલ નિર્ણય લેવાની પ્રક્રિયા પર વધુ ચોક્કસ નિયંત્રણની મંજૂરી આપે છે, જ્યાં એક શરત બીજી શરતનાં પરિણામ પર આધારિત હોય છે. નેસ્ટેડ-*if* વિધાનો જ્યારે નિર્ણય શરતોના બહુવિધ સ્તરો પર આધારિત હોય ત્યારે ઉપયોગી થાય છે.

નેસ્ટેડ-if વિધાનનું ઉદાહરણ

```
    /* Program to illustrate use of nested-if  statement.
       Find the largest of given 3 numbers. */

    #include <stdio.h>
    int main() {
        int number1, number2, number3;

        printf("Enter three numbers:");
        scanf("%d %d %d", &number1, &number2, &number3);

        if(number1 > number2)
        {
            if(number1 > number3)
                printf("Number1 is the biggest number.");
            else
                printf("Number3 is the biggest number.");
```

```
        }
        else
        {
            if(number2 > number3)
                printf("Number2 is the biggest number.");
            else
                printf("Number3 is the biggest number.");
        }
        return 0;
    }

    Result:
    Enter three numbers:5 10 15
    Number3 is the biggest number.
```

આ C પ્રોગ્રામ યુઝર પાસેથી ત્રણ સંખ્યાઓ ઈનપુટ તરીકે લે છે અને સૌથી મોટી સંખ્યા શોધવા અને પ્રિન્ટ કરવા માટે નેસ્ટેડ-*if* નિવેદનોનો ઉપયોગ કરે છે. સૌપ્રથમ, તે ચકાસે છે કે શું number1 એ number2 કરતાં મોટો છે. જો આ સાચું હોય, તો તે પછી ચકાસે છે કે શું number1 એ number3 કરતાં પણ મોટો છે. જો આ બંને શરતો સાચી હોય, તો તે પ્રિન્ટ કરે છે કે number1 સૌથી મોટો છે. જો નહીં, તો તે પ્રિન્ટ કરે છે કે number3 સૌથી મોટો છે. જ્યારે પ્રથમ શરત (number1 > number2) ખોટી હોય, ત્યારે તે બીજી શરત (number2 > number3) ચકાસે છે. જો હા, તો તે પ્રિન્ટ કરે છે કે number2 સૌથી મોટો છે. અન્યથા, તે પ્રિન્ટ કરે છે કે number3 સૌથી મોટો છે.

else-if લેડર (else-if Ladder)

else-if લેડરનો ઉપયોગ ત્યારે થાય છે જ્યારે આપણે એક પછી એક બહુવિધ શરતો ચકાસવાની જરૂર હોય. તે ઘણા સંભવિત વિકલ્પોમાંથી એક વિકલ્પ પસંદ કરવામાં મદદ કરે છે. શરતોની ચકાસણી ઉપરથી નીચે તરફ કરવામાં આવે છે, અને જે પ્રથમ સાચી શરત મળે, તેના માટેનો કોડનો બ્લોક અમલમાં મૂકાય છે. જો એક પણ શરત સાચી ન હોય, તો છેલ્લો *else* બ્લોક અમલમાં મૂકાય છે.

else-if લેડરનું ઉદાહરણ

ધારો કે આપણે વિદ્યાર્થીના ગુણના આધારે જુદા જુદા ગ્રેડ (A, B, C, D, F) પ્રિન્ટ કરવા માંગીએ છીએ. આ પ્રકારની પરિસ્થિતિઓમાં *else-if* લેડર ઉપયોગી છે.

```
/* Program to illustrate use of else-if ladder statement.
   Print student's grade based on marks. */

#include <stdio.h>
int main() {
    int marks = 75;    /* Assume that student has scored 75 */
    if(marks >= 90)
         printf("Grade: A");
    else if(marks >= 80)
            printf("Grade: B");
         else if(marks >= 70)
```

```
                    printf("Grade: C"); /*This will be executed*/
                else if(marks >= 40)
                        printf("Grade: D");
                    else
                        printf("Grade: F");
        return 0;
    }
    Result:
    Grade: C
```

આ C પ્રોગ્રામ ગુણના આધારે વિદ્યાર્થીનો ગ્રેડ નક્કી કરવા અને પ્રિન્ટ કરવા માટે *else-if* લેડરનો ઉપયોગ કરે છે. marks ચલને 75 નું મૂલ્ય આપવામાં આવ્યું છે. પ્રોગ્રામ સૌપ્રથમ ચકાસે છે કે શું marks 90 કે તેથી વધુ છે (ગ્રેડ A), પછી ચકાસે છે કે શું તે 80 કે તેથી વધુ છે (ગ્રેડ B), અને પછી ચકાસે છે કે શું તે 70 કે તેથી વધુ છે (ગ્રેડ C). કેમકે 75 એ 70 કરતા વધારે છે પરંતુ 80 કરતા ઓછો છે, તેથી ગ્રેડ C માટેની શરત સાચી છે, તેથી તે "Grade: C" પ્રિન્ટ કરે છે. ઉપરથી નીચે સુધી એક સમયે માત્ર એક જ શરત ચકાસવામાં આવે છે, અને જ્યારે યોગ્ય મેચ મળી આવે છે, ત્યારે તે સંબંધિત બ્લોકનો અમલ થાય છે અને બાકીના છોડી દેવામાં આવે છે.

switch વિધાન (switch Statement)

C પ્રોગ્રામિંગમાં *switch* વિધાન વ્યાપકપણે ઉપયોગમાં લેવાતું નિયંત્રણ સંરચના છે જે આપણને મૂલ્યના આધારે કોડના વિવિધ બ્લોક્સને અમલમાં મૂકવાની મંજૂરી આપે છે. જ્યારે આપણી પાસે બહુવિધ સંભવિત અમલના માર્ગો હોય, ત્યારે તે *if-else, if-else* નિવેદનોની લાંબી શૃંખલાનો વૈકલ્પિક ઉપાય છે. *switch* વિધાનની વાક્યરચના નીચે પ્રમાણે છે.

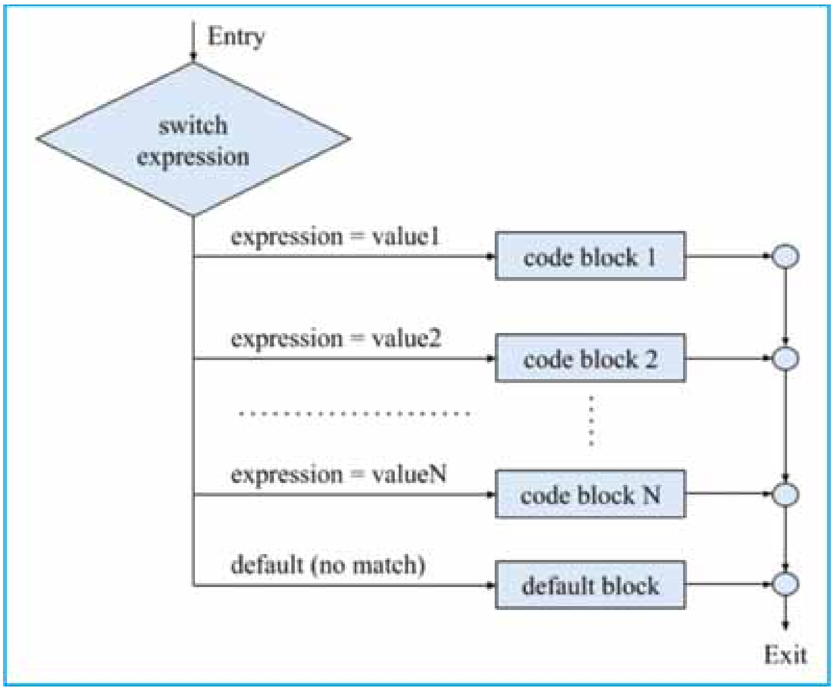
```
switch (expression)
{
    case value1:
        code block1
        break;
    case value2:
        code block2
        break;
      .....
      .....
    case valueN:
        code blockN
        break;
    default:
        default code block (optional)
}
```

switch વિધાનના વિવિધ અંગોની લાક્ષણિકતા નીચે મુજબ છે:

- **પદાવલી (Expression) :** *switch* સ્ટેટમેન્ટમાં પદાવલીની કિંમત integer, character કે enum ડેટા ટાઈપ હોવી જોઈએ, જેના પરથી નક્કી થશે કે કયા કેસ બ્લોકનો અમલ થશે.
- **બહુવિધ કેસ લેબલ્સ (Multiple *case* Labels) :** આપણે *switch* ની અંદર ઘણાં કેસ લેબલને વ્યાખ્યાયિત કરી શકીએ છીએ, જેમાં દરેક કેસ પછી એક અચલ મૂલ્ય (constant value) હોય છે. જ્યારે *switch* નું મૂલ્ય કોઈ કેસ લેબલના મૂલ્ય સાથે મેચ થાય છે, ત્યારે અનુરૂપ કોડ બ્લોકનો અમલ થાય છે.

- ***break* કીવર્ડ :** *break* વિધાન *switch* બ્લોકની અંદર મહત્વપૂર્ણ છે. એકવાર મેચ થતો કેસ મળી જાય અને તેનો કોડ અમલ થઈ જાય, પછી *break* વિધાન *switch* ને સમાપ્ત કરે છે. *break* વિના, ત્યારબાદના કેસનો કોડ પણ અમલી થશે, જે અનિચ્છનીય છે. જો *break* ભૂલી જવાય તો તે ઘણીવાર અનિચ્છનીય ભૂલ સાબિત થાય છે.
- ***default* કેસ (વૈકલ્પિક પરંતુ ભલામણપાત્ર) :** *default* કેસ વૈકલ્પિક છે પરંતુ તે ઇચ્છનીય છે. જ્યારે આપેલા કેસમાંથી એક પણ સાચો કેસ ન મળે ત્યારે આ કેસનો અમલ થાય છે.

આકૃતિ 10.4માં *switch* વિધાનનું કાર્ય ફ્લોચાર્ટ સાથે દર્શાવેલ છે.



આકૃતિ 10.4 : *switch* વિધાનનો ફ્લોચાર્ટ

ચાલો *switch* વિધાનને ઉદાહરણ સાથે સમજીએ.

switch વિધાનનું ઉદાહરણ

```
/* Program to illustrate use of Switch statement. It prints the
corresponding weekday based on a given number. */

#include <stdio.h>
int main() {
    int day = 3;

    switch(day) {
        case 1:
            printf("Monday");
            break;
        case 2:
            printf("Tuesday");
            break;
        case 3:
            printf("Wednesday");
```

```
                break;
            default:
                printf("Invalid day");
        }
        return 0;
    }

Result:
Wednesday
```

આ C પ્રોગ્રામ day નામના ચલના મૂલ્યના આધારે અઠવાડિયાના દિવસનું નામ પ્રિન્ટ કરવા માટે *switch* વિધાનનો ઉપયોગ કરે છે. day ચલનું મૂલ્ય 3 પર સેટ કરેલું છે, તેથી પ્રોગ્રામ સામ્યતા શોધવા માટે દરેક *case* ની તપાસ કરે છે. જ્યારે તે case 3 પર પહોંચે છે, ત્યારે તે "Wednesday" પ્રિન્ટ કરે છે અને *break* નિવેદનનો ઉપયોગ કરીને *switch* માંથી બહાર નીકળી જાય છે. જો day નું મૂલ્ય એક પણ *case* સાથે મેચ ન થાય, તો *default* વિભાગ અમલમાં મુકાય છે અને "Invalid day" પ્રિન્ટ કરે છે. આ પ્રોગ્રામને સોમવાર (Monday) થી રવિવાર (Sunday) સુધીના અઠવાડિયાના દરેક દિવસને આવરી લેતા સાતેય *case* નો સમાવેશ કરવા માટે વિસ્તૃત કરી શકાય છે.

પુનરાવર્તન વિધાનો (Looping Statements)

C પ્રોગ્રામિંગમાં પુનરાવર્તન (લૂપિંગ) વિધાનો મૂળભૂત નિયંત્રણ પ્રવાહ સંરચનાઓ છે જે આપણને કોડના એક બ્લોકને વારંવાર અમલ કરવાની મંજૂરી આપે છે. જ્યારે આપણે એક જ કાર્ય ઘણી વખત કરવું હોય અને તે જ કોડ ફરીથી લખવો ન પડે, ત્યારે તે આવશ્યક છે. કલ્પના કરો કે આપણે 1 થી 100 સુધીના નંબરો પ્રિન્ટ કરવાની જરૂર છે; 100 વાર *printf()* લખવાને બદલે, આપણે તે કાર્યક્ષમ રીતે કરવા માટે લૂપનો ઉપયોગ કરી શકીએ છીએ. આ વિધાનો પુનરાવર્તિત કાર્યોને સ્વચાલિત કરવામાં મદદ કરે છે, જે આપણા પ્રોગ્રામને વધુ સંક્ષિપ્ત બનાવે છે.

લૂપીંગ વિધાનોના પ્રકાર

C ત્રણ પ્રકારના લૂપિંગ વિધાનો પ્રદાન કરે છે, જેમાંથી દરેક અલગ-અલગ પરિસ્થિતિઓ માટે યોગ્ય છે:

1. ***for* લૂપ :** *for* લૂપ એ એક પૂર્વ-પરીક્ષણ લૂપ (pre-test loop) છે, જેનો ઉપયોગ સામાન્ય રીતે ત્યારે થાય છે જ્યારે આપણે અગાઉથી જાણતા હોઈએ છીએ કે આપણે કોઈ કાર્યને કેટલી વાર પુનરાવર્તિત કરવા માંગીએ છીએ.
2. ***while* લૂપ :** *while* લૂપ પણ એક પૂર્વ-પરીક્ષણ લૂપ છે, પરંતુ તે ત્યારે વધુ યોગ્ય છે જ્યારે પુનરાવર્તનોની સંખ્યા અજ્ઞાત હોય અને તે કોઈ ચોક્કસ શરત પૂરી થવા પર આધાર રાખે છે.
3. ***do-while* લૂપ :** *do-while* લૂપ એ પશ્ચાત-પરીક્ષણ લૂપ (post-test loop) છે, જેનો અર્થ છે કે તેની શરતનું મૂલ્યાંકન લૂપનું બોડી ઓછામાં ઓછી એક વાર અમલ થયા પછી કરવામાં આવે છે.

ચાલો આપણે દરેક લૂપના અગત્યના લક્ષણો વિષે ચર્ચા કરીએ.

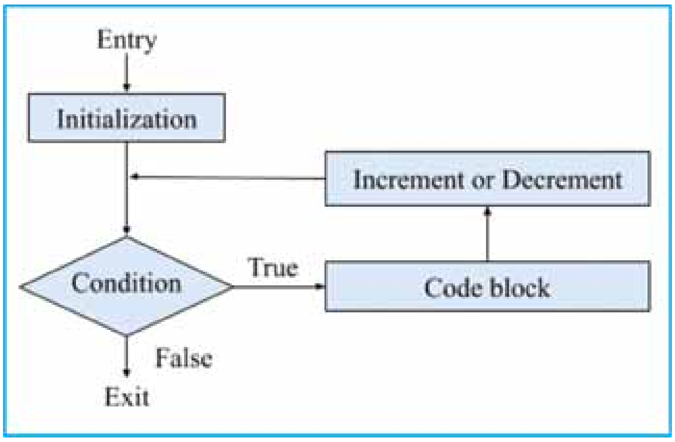
for લૂપ (for Loop)

for લૂપ એ એક પૂર્વ-પરીક્ષણ લૂપ છે જેનો ઉપયોગ સામાન્ય રીતે ત્યારે થાય છે જ્યારે આપણે કોઈ કાર્યને પુનરાવર્તિત કરવા માટે નિશ્ચિત સંખ્યામાં પુનરાવર્તનો ધ્યાનમાં રાખીએ છીએ. તે પ્રવેશ-નિયંત્રણ લૂપ (entry-controlled loop) તરીકે પણ ઓળખાય છે. તે સંક્ષિપ્ત છે અને શરૂઆત (initialization), શરત તપાસ (condition checking), અને વધારો/ઘટાડો (increment/decrement) ને એક લાઈનમાં જ ધરાવે છે.

for વિધાનની વાક્યરચના નીચે પ્રમાણે છે:

```
for (initialization; condition; increment/decrement) {
    code-block;  /* body of for loop */
}
```

આકૃતિ 10.5 ફ્લોચાર્ટની મદદથી for લૂપનું કાર્ય સમજાવે છે.



આકૃતિ 10.5 : *for* લૂપનો ફ્લોચાર્ટ

for લૂપ સંબંધિત મુખ્ય બાબતો

- **શરૂઆત (Initialization) :** આ ભાગનો લૂપની શરૂઆતમાં ફક્ત એક જ વાર અમલ થાય છે, જ્યાં લૂપને નિયંત્રિત કરતા ચલને મૂલ્ય આપવામાં આવે છે (જેમ કે i = 0;).
- **શરત (Condition) :** આની ચકાસણી દરેક પુનરાવર્તન (iteration) પહેલાં કરવામાં આવે છે. જો તે સાચી થાય, તો લૂપનું બોડી અમલ થાય છે; જો તે ખોટી થાય, તો લૂપ સમાપ્ત થઈ જાય છે.
- **વધારો/ઘટાડો (Increment/Decrement) :** આ ભાગનો લૂપનું બોડી અમલ થયા પછી દરેક પુનરાવર્તનના અંતે અમલ થાય છે. વધારો અથવા ઘટાડાની આ પ્રક્રિયા લૂપ નિયંત્રણ ચલને અપડેટ કરે છે, એટલે કે તેને પછીની શરત તપાસ માટે તૈયાર કરે છે. આ ચક્ર ત્યાં સુધી ચાલુ રહે છે જ્યાં સુધી શરત ખોટી ન થઈ જાય. શરત ખોટી થાય ત્યારે લૂપ સમાપ્ત થાય છે અને નિયંત્રણ લૂપ પછીના આગલા વિધાન પર જાય છે.

for લૂપનું ઉદાહરણ

```
/* Program to print 1 to 5 numbers using a for Loop. */

#include <stdio.h>
int main() {
    int i;
    for(i = 1; i <= 5; i++) {
        printf("%d ", i);
    }
    return 0;
}
Result:
1 2 3 4 5
```

આ પ્રોગ્રામ 1 થી 5 સુધીની સંખ્યાઓ પ્રિન્ટ કરવા માટે *for* લૂપનો ઉપયોગ કરે છે. લૂપની શરૂઆત (i = 1) સાથે શરૂ થાય છે, જે પ્રારંભિક કિંમત સેટ કરે છે. શરત (i <= 5) સુનિશ્ચિત કરે છે કે લૂપ ત્યાં સુધી ચાલે જ્યાં સુધી i ની કિંમત 5 કરતાં નાની અથવા બરાબર હોય. દરેક પુનરાવર્તન પછી, i++ વિધાન i ની કિંમતમાં 1 નો વધારો કરે છે. લૂપની અંદર, *printf()* દ્વારા i ની વર્તમાન કિંમતને પ્રિન્ટ કરવામાં આવે છે. જ્યારે ચલ i ની કિંમત 6 થઈ જાય છે, ત્યારે શરત (i <= 5) ખોટી પડે છે અને લૂપ સમાપ્ત થઈ જાય છે.

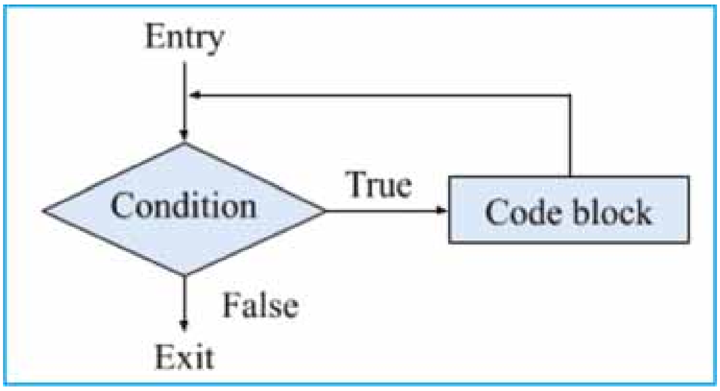
while લૂપ (while Loop)

while લૂપ પણ એક પૂર્વ-પરીક્ષણ લૂપ છે, પરંતુ તે ત્યારે વધુ યોગ્ય છે જ્યારે પુનરાવર્તનોની સંખ્યા અજ્ઞાત હોય અને તે કોઈ ચોક્કસ શરત પૂરી થવા પર આધાર રાખતી હોય. જ્યાં સુધી તેની શરત સાચી રહે છે, ત્યાં સુધી લૂપ ચાલુ રહે છે. *while* લૂપને પ્રવેશ-નિયંત્રિત લૂપ તરીકે પણ ઓળખવામાં આવે છે.

while વિધાનની વાક્યરચના નીચે પ્રમાણે છે:

```
while (condition) {
    code-block;     /* body of while loop */
}
```

શરતનું મૂલ્યાંકન દરેક પુનરાવર્તન પહેલાં કરવામાં આવે છે. જો તેનું મૂલ્યાંકન સાચું થાય, તો લૂપની બોડીનો અમલ થાય છે, નહીં તો લૂપ સમાપ્ત થાય છે. આકૃતિ 10.6 ફ્લોચાર્ટ દ્વારા *while* લૂપની કાર્ય પધ્ધતિ દર્શાવે છે.



આકૃતિ 10.6 : *while* લૂપનો ફ્લોચાર્ટ

ચાલો, નીચેના પ્રોગ્રામ દ્વારા *while* લૂપનું કાર્ય સમજીએ

while લૂપનું ઉદાહરણ

```
/* Program to illustrate use of while loop.
   Program will read and print numbers until a user enters 0.
*/
#include <stdio.h>
int main(){
    int num;

    printf("Enter a number (0 to quit): ");
    scanf("%d", &num);

    while (num != 0) {
        printf("You entered: %d \n", num);
        printf("Enter a number (0 to quit): ");
        scanf("%d", &num);
    }

    printf("Program exited as you have pressed 0.");
    return 0;
}
```

```
Result:

Enter a number (0 to quit): 1
You entered: 1
Enter a number (0 to quit): 5
You entered: 5
Enter a number (0 to quit): 0
Program exited as you have pressed 0.
```

આ પ્રોગ્રામ યુઝર પાસેથી નંબરો સ્વીકારવાનું ચાલુ રાખવા માટે *while* લૂપનો ઉપયોગ કરે છે જ્યાં સુધી વપરાશકર્તા 0 દાખલ ન કરે. તે યુઝરને એક નંબર દાખલ કરવા માટે કહીને શરૂ થાય છે અને *scanf()* નો ઉપયોગ કરીને ઈનપુટ વાંચે છે. જો દાખલ કરેલ નંબર 0 ન હોય, તો તે નંબરને *printf()* નો ઉપયોગ કરીને પ્રિન્ટ કરે છે અને બીજું ઈનપુટ માંગે છે. આ ચક્ર ત્યાં સુધી ચાલુ રહે છે જ્યાં સુધી ઈનપુટ 0 ન હોય. એકવાર યુઝર 0 દાખલ કરે, પછી શરત num != 0 (નંબર 0 નથી) ખોટી પડે છે, તેથી લૂપ અટકી જાય છે અને પ્રોગ્રામ એક સંદેશ પ્રિન્ટ કરે છે કે તે બહાર નીકળી ગયો છે.

do-while લૂપ (do-while Loop)

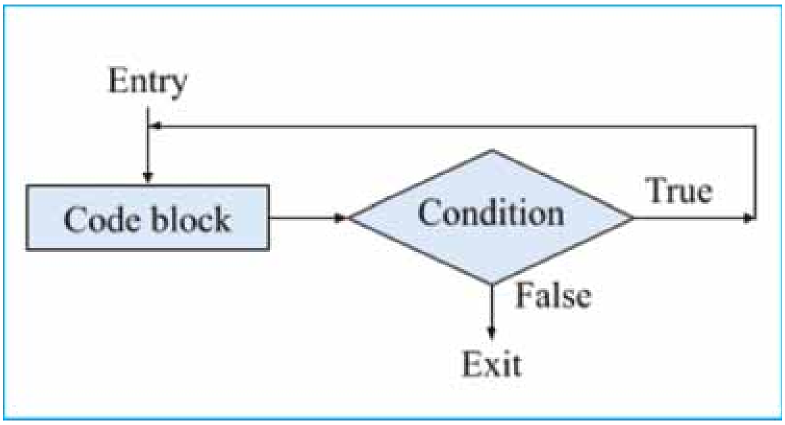
C ભાષામાં *do-while* લૂપ એક એવી લૂપ છે જેમાં કોડના બ્લોકનો ઓછામાં ઓછો એક વાર અમલ થાય, કારણ કે શરતનું મૂલ્યાંકન લૂપના બોડી પછી કરવામાં આવે છે. આપેલ શરત સાચી હોય ત્યાં સુધી તે અમલ થવાનું ચાલુ રાખે છે. આ લૂપ એવી પરિસ્થિતિઓ માટે ઉપયોગી છે જ્યાં શરત તપાસતા પહેલા કોડ ચાલવો જરૂરી હોય. ઉદાહરણ તરીકે, તેનો ઉપયોગ મેનૂ (menu) દર્શાવવા માટે થઈ શકે છે, અને ત્યારબાદ, યુઝરના ઈનપુટના આધારે પ્રોગ્રામનો બાકીનો ભાગ ચાલે છે. *do-while* લૂપની વાક્યરચના નીચે પ્રમાણે છે:

```
do {
    code-block;     /* body of do-while loop */
} while (condition);
```

શરતનું મૂલ્યાંકન દરેક પુનરાવર્તન પછી કરવામાં આવે છે. જો શરતનું મૂલ્યાંકન સાચું થાય, તો લૂપની અંદરના કોડનો ફરીવાર અમલ થાય છે, નહીં તો લૂપ સમાપ્ત થાય છે.

do-while લૂપને નિર્ગમ-નિયંત્રિત (exit-controlled) તરીકે વર્ગીકૃત કરવામાં આવે છે કારણ કે શરતનું મૂલ્યાંકન લૂપના બોડીના અમલ પછી કરવામાં આવે છે. તેનાથી વિપરીત, *for* લૂપ અને *while* લૂપને પ્રવેશ-નિયંત્રિત લૂપ ગણવામાં આવે છે કારણ કે લૂપની અંદરનો કોડ ચાલે તે પહેલાં શરતોનું મૂલ્યાંકન કરવામાં આવે છે.

આકૃતિ 10.7 *do-while* લૂપનું કાર્ય ફ્લોચાર્ટ સાથે દર્શાવે છે.



આકૃતિ 10.7 : *do-while* લૂપનો ફ્લોચાર્ટ

ચાલો, નીચેના પ્રોગ્રામ દ્વારા *do-while* લૂપનું કાર્ય સમજીએ.

do-while લૂપનું ઉદાહરણ

```
/* Program to illustration  use of do-while.
   Program performs Addition or Subtraction based on user choice. */
#include <stdio.h>
int main() {
    int n1=10, n2=5;
    int choice;

    do {
            printf("\n1. Addition \n2. Subtraction \n3. Exit\n");

            printf("Enter your choice: ");
            scanf("%d", &choice);

            if(choice == 1)
                 printf("Addition (n1+n2) = %d \n", n1+n2);

            if(choice == 2)
                 printf("Subtraction (n1-n2) = %d \n", n1-n2);

    } while (choice != 3);

    printf("You have pressed 3. Exiting Program...");
    return 0;
}

Result:
1. Addition
2. Subtraction
3. Exit
Enter your choice: 1
Addition (n1+n2) = 15

1. Addition
2. Subtraction
3. Exit
Enter your choice: 2
Subtraction (n1-n2) = 5

1. Addition
2. Subtraction
3. Exit
Enter your choice: 3
You have pressed 3. Exiting Program...
```

આ પ્રોગ્રામ યુઝરની પસંદગીના આધારે વારંવાર ગાણિતિક ક્રિયાઓ કરવા માટે *do-while* લૂપનો ઉપયોગ દર્શાવે છે. તે n1 = 10 અને n2 = 5 એમ બે ચલ વાપરે છે અને પછી ત્રણ વિકલ્પો સાથેનો એક મેનૂ દર્શાવે છે: Addition (સરવાળો), Subtraction (બાદબાકી), અને Exit (બહાર નીકળો). *do-while* લૂપની અંદર પ્રોગ્રામ યુઝર પાસેથી ઇનપુટ લે છે અને અનુરૂપ ઓપરેશન કરે છે: જો યુઝર 1 દાખલ કરે, તો તે n1 અને n2 નો સરવાળો પ્રિન્ટ કરે છે, જો 2 દાખલ કરે, તો તે તેમની બાદબાકી પ્રિન્ટ કરે છે. જ્યાં સુધી યુઝર 3 દાખલ ન કરે ત્યાં સુધી લૂપ ચાલુ રહે છે, આ સમયે પ્રોગ્રામ એક સંદેશ પ્રિન્ટ કરે છે અને સમાપ્ત થાય છે. *do-while* નો ઉપયોગ એ સુનિશ્ચિત કરે છે કે જો બહાર નીકળવાનો વિકલ્પ 3 સૌથી પહેલા દાખલ કરવામાં આવે તો પણ મેનૂ ઓછામાં ઓછું એક વાર દર્શાવવામાં આવે છે.

જમ્પિંગ વિધાનો (Jumping Statements)

C ભાષામાં જમ્પિંગ વિધાનોનો ઉપયોગ સામાન્ય ક્રમિક પ્રવાહને પ્રોગ્રામના નિયંત્રણને અવગણીને એક ભાગમાંથી બીજા ભાગમાં સ્થાનાંતરિત કરવા માટે થાય છે. આ વિધાનો ચોક્કસ પધ્ધતિના આધારે અમલના પ્રવાહને બદલવામાં મદદ કરે છે.

ચાલો, આપણે જમ્પિંગ વિધાનો *break, continue* અને *goto* ને ઉદાહરણ સાથે સમજીએ.

break

break નો ઉપયોગ લૂપ અથવા *switch-case* તરત જ સમાપ્ત કરવા માટે થાય છે. જ્યારે તેનો અમલ થાય છે, ત્યારે નિયંત્રણ લૂપ અથવા *switch* પછીના સ્ટેટમેન્ટ પર સ્થાનાંતરિત થાય છે. નીચે આપેલ C કોડનું અવલોકન કરો:

```
for (int i = 1; i <= 5; i++) {
    if  (i == 3)
        break;
    printf("%d ", i);
}
/* output: 1 2  */
```

આ C કોડમાં *for* લૂપ 1 થી 5 સુધીના નંબરો માટે લખેલી છે. જો કે, લૂપની અંદર એક *if* શરત છે જે તપાસે છે કે i નું મૂલ્ય 3 છે કે નહીં. જ્યારે આ શરત સાચી બને છે, ત્યારે *break* નો અમલ થાય છે, જે લૂપને તરત જ સમાપ્ત કરી દે છે. પરિણામે, લૂપ ફક્ત i = 1 અને i = 2 માટે જ ચાલે છે, અને તે મૂલ્યો પ્રિન્ટ કરે છે. જ્યારે i = 3 થાય છે, ત્યારે *break* લૂપના વધુ અમલને અટકાવે છે, તેથી નંબરો 3, 4, અને 5 પ્રિન્ટ થતા નથી. તેથી, કોડનો આઉટપુટ 1 2 છે.

continue

continue વિધાનનો ઉપયોગ લૂપના વર્તમાન પુનરાવર્તનને (current iteration) છોડીને, પછીના પુનરાવર્તન (next iteration) પર જવા માટે થાય છે. આ વિધાનમાં, લૂપ સમાપ્ત થતી નથી; તેમાં માત્ર અત્યારનું પુનરાવર્તન અવગણવામાં આવે છે.

continue નું ઉદાહરણ

```
for (int i = 1; i <= 5; i++) {
    if  (i == 3)
        continue;
    printf("%d ", i);
}
/* Output: 1 2 4 5 */
```

આ કોડમાં, લૂપની અંદર એક *if* વિધાન તપાસે છે કે શું ચલ i ની કિંમત 3 છે. જો શરત સાચી પડે છે, તો *continue* વિધાન અમલમાં આવે છે, જે વર્તમાન પુનરાવર્તનને છોડી દે છે અને *printf()* ને અમલમાં લાવ્યા વિના પછીના પુનરાવર્તન પર જાય છે. પરિણામે, નંબર 3 પ્રિન્ટ થતો નથી. અન્ય તમામ નંબરો (1, 2, 4, 5) સામાન્ય રીતે પ્રિન્ટ થાય છે. તેથી, આ કોડ પ્રિન્ટ કરશેઃ 1 2 4 5.

goto

goto વિધાનનો ઉપયોગ પ્રોગ્રામના લેબલ કરેલા ભાગ પર જવા માટે થાય છે. આધુનિક પ્રોગ્રામિંગમાં તેનો ઉપયોગ ટાળવામાં આવે છે, કારણ કે તે કોડને અસંરચિત (unstructured) અને ગૂંચવાડાભર્યો બનાવી શકે છે.

goto નું ઉદાહરણ

```
int RollNo = 1;
if  (RollNo == 1)
     goto ABC;
printf("My name is Purv");
ABC:
printf("My name is Mudra");

/* Output: My name is Mudra */
```

આ C કોડ *goto* વિધાનના ઉપયોગને દર્શાવે છે. ચલ RollNo ને 1 કિંમત આપવામાં આવે છે. *if* શરત તપાસે છે કે શું RollNo == 1 છે, જે સાચું છે, તેથી પ્રોગ્રામ *goto* ABC; નો અમલ કરે છે. આનાથી સીધું ABC: લેબલ પર જમ્પ થાય છે, અને આઉટપુટ તરીકે "My name is Mudra" પ્રિન્ટ થાય છે. *printf*("My name is Purv"); વાળી લાઈન છોડી દેવામાં આવે છે અને તેનો અમલ થતો નથી.

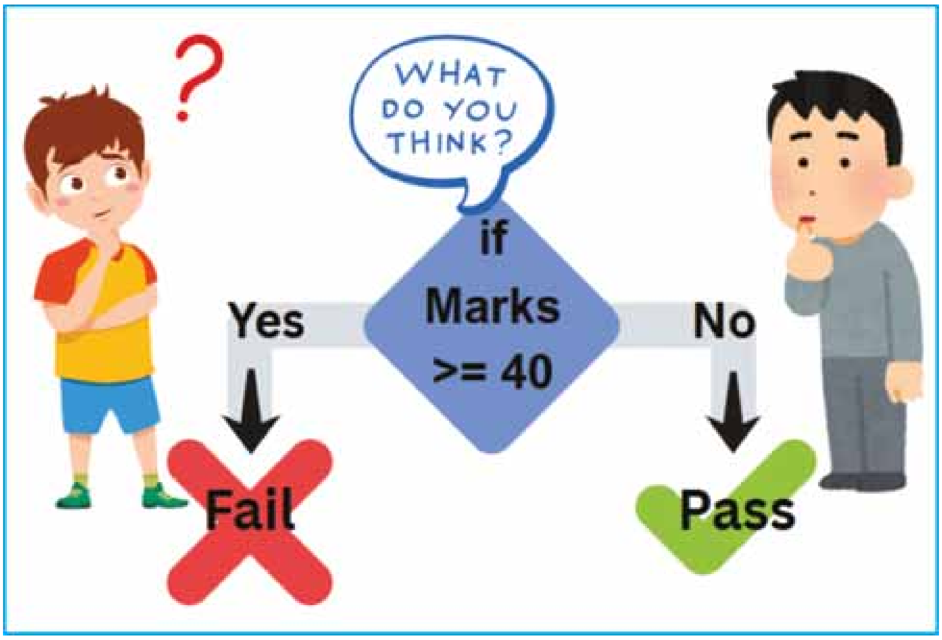
સારાંશ

આ પ્રકરણમાં, આપણે C પ્રોગ્રામિંગમાં આવશ્યક નિર્ણય લેવાની નિયંત્રણ સંરચનાઓ વિશે જાણ્યું, જે ચોક્કસ શરતોના આધારે પ્રોગ્રામના પ્રવાહને સંચાલિત કરવા માટે મહત્વપૂર્ણ છે. આપણે નિર્ણય લેવાના વિધાનો જેવા કે if, if-else, લેડર else-if અને switch નો અભ્યાસ કર્યો. આ વિધાનો પ્રોગ્રામરોને તાર્કિક માર્ગો બનાવવાની મંજૂરી આપે છે જે વિવિધ ઇનપુટ અને પરિસ્થિતિને જરૂર પ્રમાણે પ્રતિભાવ આપે છે, જેનાથી વાસ્તવિક દુનિયાની નિર્ણય પ્રક્રિયાઓનું નિરૂપણ કરવામાં મદદ મળે છે. આપણે લૂપિંગ વિધાનો જેવા કે for, while અને do-while નો પણ અભ્યાસ કર્યો. જેનો ઉપયોગ કાર્યોને પુનરાવર્તિત કરવા માટે થાય છે. વધુમાં, આપણે શીખ્યા કે break, continue અને goto જેવા જમ્પિંગ વિધાનોનો ઉપયોગ પ્રોગ્રામની અંદર નિયંત્રણના પ્રવાહને કેવી રીતે સંચાલિત કરવા માટે કરી શકાય છે. આ નિયંત્રણ સંરચનાઓને સમજી અને લાગુ કરીને, પ્રોગ્રામરો વાસ્તવિક દુનિયાની વિવિધ પરિસ્થિતિ પ્રમાણે પ્રતિભાવ આપતા ફ્લેક્ષિબલ અને કાર્યક્ષમ C પ્રોગ્રામ્સ બનાવી શકે છે.

સ્વાધ્યાય

1. C પ્રોગ્રામિંગના નિર્ણય લેવાની સંરચનાઓની યાદી બનાવો.
2. C માં if અને switch વિધાન વચ્ચેનો મૂળભૂત તફાવત સમજાવો. એવા સંજોગોની ચર્ચા કરો જ્યાં દરેકનો ઉપયોગ કરવો વધુ યોગ્ય રહેશે.
3. નેસ્ટેડ-if ના ખ્યાલનું વર્ણન કરો. એક વ્યવહારુ ઉદાહરણ આપો જ્યાં નેસ્ટેડ-if જરૂરી હોય.

4. **નીચે આપેલા ચિત્રમાં શું ખોટું છે તે ઓળખો:**



5. C પ્રોગ્રામિંગમાં while અને do-while લૂપની તુલના કરો. દરેક લૂપ પ્રકાર માટે, તેના મુખ્ય ઉપયોગને સમજાવો અને તેની વાક્યરચના દર્શાવતું એક નાનું કોડ સહિત ઉદાહરણ આપો. ઉપરાંત, કયા સંજોગોમાં while અને do-while લૂપનો ઉપયોગ કરવો જોઈએ તે યોગ્ય ઉદાહરણ સાથે સમજાવો.
6. C પ્રોગ્રામિંગમાં લૂપીંગ વિધાનોનો ઉપયોગ સમજાવો.
7. break અને continue વિધાનો લૂપની કામગીરીને કેવી રીતે પ્રભાવિત કરે છે? બંને માટે એક ઉદાહરણ આપો જે while લૂપની અંદર તેના પ્રભાવને દર્શાવે.
8. switch વિધાનનો ઉપયોગ યોગ્ય C પ્રોગ્રામ સાથે સમજાવો.
9. switch માં default નો ઉપયોગ શું છે?
10. C પ્રોગ્રામિંગમાં gotoની ભૂમિકા પર સંક્ષિપ્ત નોંધ લખો
11. **સાચું કે ખોટું જણાવો.**
    (1) C પ્રોગ્રામિંગમાં if વિધાન શરત સાચી હોય ત્યારે જ કોડનો બ્લોક અમલ કરે છે.
    (2) if-else વિધાનમાં else બ્લોક હંમેશા અમલ થાય છે, ભલે આપેલી શરત સાચી હોય કે ખોટી.
    (3) for લૂપનો ઉપયોગ ત્યારે થાય છે જ્યારે પુનરાવર્તનની સંખ્યા પહેલેથી જ જાણીતી હોય.
    (4) do-while લૂપ બોડીનો અમલ કરતા પહેલાં જ શરત ચકાસે છે.
    (5) continue વિધાનનો ઉપયોગ પ્રોગ્રામમાંથી તરત જ બહાર નીકળવા માટે થાય છે.
12. **ખાલી જગ્યાઓ પૂરો.**
    (1) C પ્રોગ્રામિંગમાં ____________ નિર્ણય લેતું વિધાન આપેલી શરત સાચી હોય ત્યારે જ કોડ બ્લોકનો અમલ કરવા માટે ઉપયોગમાં લેવાય છે.
    (2) if-else નિવેદનમાં, જો if શરત ખોટી હોય, તો ____________ બ્લોકની અંદરનો કોડ અમલમાં આવે છે.

(3) switch બ્લોકની અંદર, દરેક અચલ મૂલ્ય ____________ લેબલ પછી દર્શાવવામાં આવે છે.

(4) ____________ લૂપ બોડીનો અમલ કરતા પહેલાં શરત ચકાસે છે.

(5) ____________ વિધાનનો ઉપયોગ લૂપ અથવા switch બ્લોકમાંથી તરત બહાર નીકળવા માટે થાય છે.

13. બહુવિકલ્પી પ્રશ્નો. સૌથી યોગ્ય જવાબ પસંદ કરો.

(1) નીચેનામાંથી કઈ પદાવલીનું મુલ્ય સાચું (true) થશે?

(a) (10 - 10) (b) (0) (c) (2 * 0) (d) (7 % 2)

(2) જો C પ્રોગ્રામિંગમાં if વિધાનની શરત (3 - 3) હોય તો શું થશે?

(a) if ની અંદરનો કોડ અમલી થશે (b) else ની અંદરનો કોડ અમલી થશે

(c) કમ્પાઈલર ત્રુટી દર્શાવશે (d) 0 પ્રિન્ટ થશે

(3) નીચેના પૈકી કયું વિધાન લૂપનું હાલનું પુનરાવર્તન છોડીને આગળનું પુનરાવર્તન ચાલુ રાખવા માટે વપરાય છે?

(a) break (b) exit (c) continue (d) goto

(4) C ભાષામાં નીચેના પૈકી કઈ લૂપ ઓછામાં ઓછી એક વખત તો ચોક્કસ ચાલે છે?

(a) for લૂપ (b) while લૂપ (c) do-while લૂપ (d) switch

(5) નીચેના C કોડનો આઉટપુટ શું હશે?

```
int i;
for(i = 0; i < 3; i++) {
    printf("%d ", i);
}
```

(a) 1 2 3 (b) 0 1 2 (c) 0 1 2 3 (d) અનંત લૂપ

(6) જ્યારે પુનરાવર્તનની સંખ્યા પહેલેથી જ નક્કી હોય ત્યારે કઈ લૂપ સૌથી યોગ્ય ગણાય છે?

(a) switch (b) do-while (c) for (d) if

(7) break વિધાનનો લૂપની અંદર શો હેતુ છે?

(a) એક પુનરાવર્તન અવગણવું (b) પ્રોગ્રામનો અમલ બંધ કરવો

(c) હાલની લૂપમાંથી બહાર નીકળવું (d) લૂપની શરૂઆતમાં જવા

(8) switch પદાવલીમાં નીચેનામાંથી કયો ડેટા પ્રકાર યોગ્ય છે?

(a) float (b) double (c) int (d) string

(9) નીચેનામાંથી કઈ લૂપ પ્રવેશ-નિયંત્રિત છે?

(a) do-while (b) goto (c) while (d) break

(10) નીચનામાંથી કયું માન્ય લૂપીંગ વિધાન છે?

(a) select (b) return (c) switch (d) for

પ્રાયોગિક સ્વાધ્યાય

1. આપેલ સંખ્યા ધન, ઋણ કે શૂન્ય છે તે તપાસવા માટે if-else વિધાનનો ઉપયોગ કરીને એક પ્રોગ્રામ લખો.
2. લેડર if-else નો ઉપયોગ કરીને ગ્રેડ ગણતરી કરતો પ્રોગ્રામ લખો. માર્ક્સ સ્વીકારો અને નીચેના નિયમો મુજબ ગ્રેડ દર્શાવોઃ
   - 80 અથવા તેથી વધુ માર્ક્સ માટે ગ્રેડ A
   - 60 થી 79 માર્ક્સ માટે ગ્રેડ B
   - 40 થી 59 માર્ક્સ માટે ગ્રેડ C
   - 40 કરતાં ઓછા માર્ક્સ માટે ગ્રેડ D
3. switch-case નો ઉપયોગ કરીને સાદું કેલ્ક્યુલેટર બનાવવા માટેનો પ્રોગ્રામ લખો. પ્રોગ્રામ બે સંખ્યાઓ અને એક ઓપરેટર (+, –, *, /) સ્વીકારે છે, switch દ્વારા સંબંધિત ક્રિયા કરે છે અને પરિણામ દર્શાવે છે.
4. એક પ્રોગ્રામ લખો જે સંખ્યા વાંચે અને for loop નો ઉપયોગ કરીને તેનું ગુણાકાર ટેબલ દર્શાવે.
5. 1 થી 10 સુધીની સંખ્યાઓ દર્શાવતી લૂપ લખો. 5 દર્શાવવાનું ટાળો (સૂચન : continue નો ઉપયોગ કરો) અને જો સંખ્યા 8 થાય તો લૂપ બંધ કરો (સૂચન : break નો ઉપયોગ કરો).

ઑબ્જેક્ટ ઓરિએન્ટેડ અભિગમ અને પ્રોગ્રામિંગ

પરિચય

આપણે અગાઉ અલ્ગોરિધમનો ઉપયોગ કરીને સમસ્યાનું નિરાકરણ કેવી રીતે કરવું તે વિશે શીખ્યા અને સમજ્યા કે અલ્ગોરિધમ કમ્પ્યૂટર પ્રોગ્રામ્સ જેવા હોય છે. સમસ્યાનું નિરાકરણ શીખ્યા પછી, આપણે C પ્રોગ્રામિંગ વિષે અભ્યાસ કર્યો, જેમાં આપણે C ભાષાના સિન્ટેક્સનો ઉપયોગ કરીને પ્રોગ્રામ લખ્યા. C એક પ્રોસિજરલ ભાષા (procedural language) છે જે પ્રોગ્રામ લખવા માટે એક પછી એક સૂચનાઓનો ઉપયોગ કરે છે. પ્રોસિજરલ પ્રોગ્રામિંગ ભાષા પ્રોગ્રામને ક્રમબદ્ધ પગલામાં આપેલ સૂચનાઓના સંદર્ભમાં ગોઠવે છે, જેના કારણે પ્રોગ્રામ મોટા અને જટિલ બનતા જાય છે. તેનું વ્યવસ્થાપન કરવું અને તેને ફરીથી ઉપયોગમાં લેવા મુશ્કેલ બની જાય છે. તે વાસ્તવિક-દુનિયાની એન્ટિટી (વસ્તુઓ)ની આસપાસ પ્રોગ્રામનો વિકાસ કે ગોઠવણ કરતું નથી અને તેના એક ભાગમાં કરવામાં આવેલો ફેરફાર અન્ય ભાગોને અનપેક્ષિત રીતે અસર કરે છે.

ઑબ્જેક્ટ ઓરિએન્ટેડ પ્રોગ્રામિંગ (Object Oriented Programming-OOP) પ્રોગ્રામને ઑબ્જેક્ટ નામની વાસ્તવિક-દુનિયાની એન્ટિટીની આસપાસ ગોઠવે છે. ઉદાહરણ તરીકે, કાર (car) એક ઑબ્જેક્ટ છે જેમાં રંગ, મોડેલ વગેરે જેવા એટ્રિબ્યુટ્સ (attributes) હોય છે અને તે ચાલી (move) શકે છે અને અટકી (stop) શકે છે. ઑબ્જેક્ટ ક્લાસ (class)માંથી બનાવવામાં આવે છે, જે ઑબ્જેક્ટ માટે ટેમ્પ્લેટ અથવા બ્લુપ્રિન્ટ તરીકે કાર્ય કરે છે. ઉદાહરણ તરીકે, કારની એક ડિઝાઇનમાંથી, આપણે ઘણી સમાન કારનું ઉત્પાદન કરી શકીએ છીએ. તમે કમ્પ્યૂટર અથવા મોબાઇલ પર જે રમતો રમી હશે, તે ઑબ્જેક્ટ ઓરિએન્ટેડ પ્રોગ્રામિંગનો ઉપયોગ કરીને બનાવવામાં આવેલી હોય છે. ઑબ્જેક્ટ ઓરિએન્ટેડ પ્રોગ્રામિંગ કોડને ખૂબ જ સ્પષ્ટ, સમજવામાં સરળ અને સુધારવા તથા વ્યવસ્થાપન કરવા માટે સરળ રાખવાની મંજૂરી આપે છે. તે ઑબ્જેક્ટ, ક્લાસ, એન્કેપ્સ્યુલેશન (encapsulation), એબ્સ્ટ્રેક્શન (abstraction), ઈન્હેરિટન્સ (inheritance) અને પોલિમોર્ફિઝમ (polymorphism) જેવા ખ્યાલોની આસપાસ કાર્ય કરે છે. આ પ્રકરણમાં આ તમામ ઑબ્જેક્ટ ઓરિએન્ટેડ પ્રોગ્રામિંગના ખ્યાલોને વાસ્તવિક-દુનિયાના ઉદાહરણો સાથે શીખીશું. આ પ્રકરણના અંતમાં ખૂબ જ લોકપ્રિય અને વ્યાપકપણે ઉપયોગમાં લેવાતી ઑબ્જેક્ટ ઓરિએન્ટેડ પ્રોગ્રામિંગ ભાષાઓનો સંક્ષિપ્ત પરિચય પણ મેળવીશું.

પ્રોસિજરલ પ્રોગ્રામિંગની મર્યાદાઓ (Limitations of Procedural Programming)

1990 પહેલાં લોકો પ્રોગ્રામ લખવા માટે પ્રોસિજરલ ભાષાઓ (procedural languages) જેમ કે BASIC, FORTRAN, COBOL, Pascal વગેરેનો ઉપયોગ કરતા હતા. પ્રોસિજરલ પ્રોગ્રામમાં ક્રમબદ્ધ સૂચનાઓ લખવામાં આવે છે. અગાઉના પ્રકરણોમાં આપણે C ભાષામાં પ્રોગ્રામ લખ્યા છે, જ્યાં આપણે આખો પ્રોગ્રામ *main()* ફંક્શનમાં સ્ટેટમેન્ટના સ્વરૂપમાં લખીએ છીએ. પ્રોસિજરલ પ્રોગ્રામિંગ નાના અથવા મધ્યમ કદના પ્રોગ્રામ માટે યોગ્ય છે. પરંતુ જ્યારે મોટા પ્રોગ્રામની વાત આવે છે, ત્યારે તેમાં ઘણા પ્રશ્નો ઉભા થાય છે. ચાલો, આપણે તેમને સંક્ષિપ્તમાં સમજીએ:

- મનુષ્યો ઑબ્જેક્ટના સ્વરૂપમાં વિશ્વને સરળતાથી સમજી અને તેની સાથે ક્રિયા-પ્રતિક્રિયા કરી શકે છે, જે સહજ રીતે જટિલ સમસ્યાઓનું નિરાકરણ લાવે છે. પ્રોસિજરલ પ્રોગ્રામિંગ પ્રોગ્રામને ક્રમબદ્ધ પગલામાં આપેલ સૂચનાઓ તરીકે લખવાની અનુમતિ આપે છે, જે મુખ્યત્વે પ્રક્રિયા (process) એટલે કે ફંક્શન/પ્રોસિજર (Functions/Procedures) પર ધ્યાન કેન્દ્રિત કરે છે, ઑબ્જેક્ટ્સ પર નહીં. અહીં માનવ વિશ્વને જે રીતે સમજે છે અને તેની સાથે ક્રિયા-પ્રતિક્રિયા કરે છે તેને પ્રતિબિંબિત કરતું નથી. તેથી, ક્યારેક

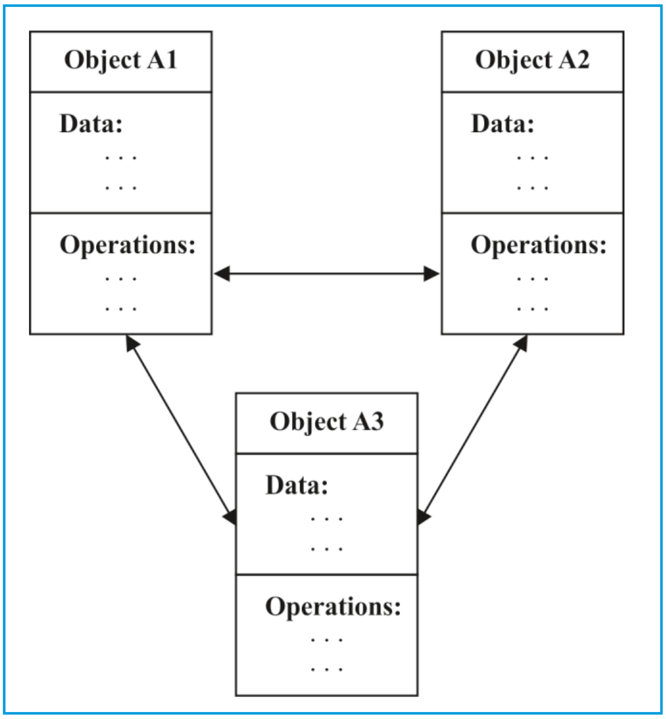
પ્રોગ્રામ મોટો અને જટિલ બની જાય છે, ત્યારે પ્રોસિજરલ પ્રોગ્રામિંગમાં સમસ્યાઓનું વ્યવસ્થાપન, સમજણ અને નિરાકરણ કરવું મુશ્કેલ બની જાય છે.

- પ્રોસિજરલ પ્રોગ્રામિંગમાં ડેટા અને ફંક્શન (ફંક્શન એ ચોક્કસ કાર્ય રજૂ કરતી એક નાની પ્રક્રિયા છે) અલગ હોય છે અને તેઓ કોઈપણ પ્રતિબંધો વિના એકબીજા સાથે ભળી જાય છે. આનાથી ગૂંચવણ થાય છે અને પ્રોગ્રામમાં ફેરફાર કરવો અથવા સમસ્યા હલ કરવી મુશ્કેલ બની જાય છે. એવું પણ શક્ય છે કે કોડ ઘણી વખત પુનરાવર્તિત થાય, જેનાથી પ્રોગ્રામ બિનજરૂરી રીતે લાંબો બને છે. ડેટાનો અનિયંત્રિત ઍક્સેસ, ઉદાહરણ તરીકે C માં ગ્લોબલ વેરીએબલ (global variable) સુરક્ષાની ચિંતાઓ પણ ઊભી કરે છે. તેનાથી પ્રોગ્રામનો કોઈપણ ભાગ અકસ્માતે મહત્ત્વપૂર્ણ ડેટા બદલી શકે છે.
- વર્તમાન એપ્લિકેશન ખૂબ જ મોટી અને જટિલ હોય છે, કારણકે તે મોટા પ્રમાણમાં ડેટા સાથે કામ કરે છે અને વિવિધ પ્રકારની સેવાઓ પ્રદાન કરે છે. આવા સંજોગોમાં, વ્યવસાયને કોડના સંગઠનની એવી જરૂર છે જે ભૂલોને સુધારવા, સેવાઓ ઉમેરવા અથવા બદલવા વગેરે સહિત ફેરફારોને ત્વરિત સમર્થન આપે. પ્રોસિજરલ પ્રોગ્રામિંગ સારી રીતે સંરચિત છે, પરંતુ તે પ્રોગ્રામ અથવા સૉફ્ટવેરનું આયોજન જે રીતે કરે છે તેના કારણે મોટી અને જટિલ એપ્લિકેશન્સ માટે ઝડપથી અપડેટ અને ફેરફારોની સરળતા પ્રદાન કરતું નથી. ઑબ્જેક્ટ ઓરિએન્ટેડ પ્રોગ્રામિંગ પ્રોગ્રામને ઑબ્જેક્ટના સ્વરૂપમાં ગોઠવીને આ ફાયદો આપે છે, જેની સાથે વ્યવહાર કરવો સરળ બને છે કારણ કે મનુષ્યો સહજ રીતે રોજિંદા જીવનમાં ઑબ્જેક્ટ સાથે કામ કરવા સક્ષમ છે.

ઉપરોક્ત તથ્યોને જોતાં, આધુનિક સોફ્ટવેર ડેવલપમેન્ટ માટે ઉદ્યોગો ઑબ્જેક્ટ ઓરિએન્ટેડ પ્રોગ્રામિંગ ભાષાઓનો ઉપયોગ કરે છે. જે ખ્યાલ પર ઑબ્જેક્ટ ઓરિએન્ટેડ પ્રોગ્રામિંગ ભાષાઓ કાર્ય કરે છે તેને સમજવું ખૂબ જ જરૂરી છે. તેને ઑબ્જેક્ટ ઓરિએન્ટેડ ખ્યાલ અથવા સિદ્ધાંતો કહેવામાં આવે છે. આ પ્રકરણમાં આપણે આધુનિક ઑબ્જેક્ટ ઓરિએન્ટેડ પ્રોગ્રામિંગ ભાષાઓ દ્વારા સમર્થિત ઑબ્જેક્ટ ઓરિએન્ટેડ ખ્યાલની વિગતવાર ચર્ચા કરીશું.

ઑબ્જેક્ટ ઓરિએન્ટેડ અભિગમ (Object-Oriented Approach)

આપણે જાણીએ છીએ કે વાસ્તવિક દુનિયા ઑબ્જેક્ટની બનેલી છે. વાસ્તવિક દુનિયાની કોઈપણ સમસ્યા કેટલાક ઑબ્જેક્ટ, તેમના સંબંધો અને તેમની વચ્ચેના સંચાર (communication)ની બનેલી હોય છે. ચાલો, એક ઘરનું ઉદાહરણ લઈએ જેમાં એક નાનો પરિવાર રહે છે. ઘરના બેડરૂમમાં પલંગ, ટેબલ, કબાટ વગેરે હોય છે. રસોડું રસોઈ માટે વપરાય છે, જેમાં ગેસ સ્ટવ, ડીશ, કપ, મિક્સર, ફ્રિજ અને અન્ય ઘણી વસ્તુઓનો સમાવેશ થાય છે. હૉલમાં સોફા, ટેલિવિઝન સેટ અને અન્ય ઉપયોગી ફર્નિચર ગોઠવાયેલા હોય છે. ઘરમાં પંખા, એસી અને અન્ય ઈલેક્ટ્રોનિક ઉપકરણો પણ છે. ઘરમાં ઘણી પ્રવૃત્તિઓ થાય છે જેમ કે રસોઈ કરવી, સાથે બેસીને વાતો કરવી, એસી ચાલુ-બંધ કરવું વગેરે. પરિવારમાં માતા-પિતા-બાળક, પતિ-પત્ની, પિતા-પુત્ર જેવા સંબંધો છે. પરિવારના સભ્યો



આકૃતિ 11.1 : ઑબ્જેક્ટ ઓરિએન્ટેડ પ્રોગ્રામનું માળખું

વિવિધ સુવિધાઓનો ઉપયોગ તેમની સાથે ક્રિયા-પ્રતિક્રિયા (interacting) કરીને કરે છે, ઉદાહરણ તરીકે, શાકભાજી કાઢવા માટે ફ્રિજ ખોલવું. ઘરનું આ વર્ણન વાસ્તવિક દુનિયામાં એક સિસ્ટમ અથવા સમસ્યાનું ઉદાહરણ છે. તેની રચના શેના પર આધારિત છે? વિવિધ ઑબ્જેક્ટ. પરિવારનો દરેક સભ્ય, બેડરૂમ, ગેસ સ્ટવ, ડીશ, કપ, પંખા, સોફા, એસી વગેરે એ ઑબ્જેક્ટના ઉદાહરણો છે. ઘરમાં રહેલી વસ્તુઓનો ઉપયોગ પરિવારના સભ્યો દ્વારા થાય છે અને પરિવારના દરેક સભ્ય એકબીજા સાથે સંબંધિત છે. આથી, ઘરને મનુષ્યો અને વિવિધ નિર્જીવ વસ્તુઓ સહિત, ચોક્કસ રીતે ગોઠવાયેલા વિવિધ ઑબ્જેક્ટનું સંગઠન કહી શકાય.

ઑબ્જેક્ટ શેનાથી બને છે? ચાલો આપણે એક પંખાને ધ્યાનમાં લઈએ. પંખામાં રંગ, કિંમત, મોડેલ, તેની બનાવટનું વર્ષ વગેરે જેવા કેટલાક એટ્રિબ્યુટ્સ હોય છે. આપણે પંખા પર ચાલુ (ON) કે બંધ (OFF) કરવો, સ્પીડ વધારવા કે ઘટાડવા જેવા ઓપરેશન કરી શકીએ છીએ. તેથી, ઑબ્જેક્ટને ગુણધર્મો (properties) અને વર્તન (behaviour) (તેના પર કરવામાં આવતી ક્રિયાઓ જેને ફંક્શન અથવા મેથડ પણ કહેવાય છે) તરીકે વર્ણવવામાં આવે છે. ઑબ્જેક્ટ ઓરિએન્ટેડ પ્રોગ્રામિંગ ઑબ્જેક્ટ પર એક બિલ્ડિંગ બ્લોક (building block) તરીકે ધ્યાન કેન્દ્રિત કરે છે અને પ્રોગ્રામ બનાવે છે. આપણે કહી શકીએ કે ઑબ્જેક્ટ ઓરિએન્ટેડ પ્રોગ્રામિંગનું કેન્દ્ર ઑબ્જેક્ટ છે, જેમાં ડેટા (ગુણધર્મો) અને ઓપરેશન્સ (ફંક્શન્સ અથવા મેથડ્સ)નો સમાવેશ થાય છે. એક ઑબ્જેક્ટ ઓરિએન્ટેડ પ્રોગ્રામમાં ઘણા વિવિધ પ્રકારના ઑબ્જેક્ટ હોઈ શકે છે જે અમુક રીતે એકબીજા સાથે સંબંધિત હોય છે અને તેઓ વિવિધ પ્રવૃત્તિઓ અથવા કાર્યો કરવા માટે એકબીજા સાથે સંચાર (communicate) કરે છે. આકૃતિ 11.1 ઑબ્જેક્ટ ઓરિએન્ટેડ પ્રોગ્રામનું માળખું દર્શાવે છે.

આકૃતિમાં ત્રણ ઑબ્જેક્ટ દર્શાવ્યા છે, જે દરેક પોતાનો ડેટા (data) અને મેથડ (method) ધરાવે છે. ઑબ્જેક્ટને જોડતા તીર (arrow) દર્શાવે છે કે ઑબ્જેક્ટ્સ એકબીજાના ફંક્શનને કૉલ કરીને એકબીજા સાથે સંચાર (communicate) કરે છે. મોટા પ્રોગ્રામમાં, ઑબ્જેક્ટનો એક સમૂહ કોઈક રીતે એકબીજા સાથે સંબંધિત હોઈ શકે છે.

પિઝાના નાના રેસ્ટોરન્ટનું નીચેનું ઉદાહરણ પ્રક્રિયાગત (procedural) અને ઑબ્જેક્ટ ઓરિએન્ટેડ (object-oriented) એમ બંને પ્રકારના અભિગમને સ્પષ્ટ કરશે.

પ્રક્રિયાગત અભિગમ (Procedural Way) : આપણે કણક (dough) કેવી રીતે બનાવવો, કેવી રીતે બેક કરવો, ઑર્ડર કેવી રીતે લેવો અને ટેબલ કેવી રીતે સાફ કરવા તેની પ્રક્રિયાના પગલાંની એક યાદી તૈયાર કરીએ છીએ. જ્યારે ગ્રાહક પિઝાનો ઑર્ડર આપે છે, ત્યારે આપણે દર વખતે મેન્યુઅલમાં આપેલી યાદીને ઉપરથી નીચે સુધી અનુસરીએ છીએ. હવે ધારો કે આપણે સેન્ડવીચ (sandwich) ઉમેરી રહ્યા છીએ, તો તમે છેલ્લે એક પગલું ઉમેરી રહ્યા છો, ભલે કેટલાક પગલાં સામાન્ય હોય. કોઈપણ ફેરફાર માટે સમગ્ર મેન્યુઅલનો સંદર્ભ લેવો પડે છે અને જેમ જેમ વધુ વસ્તુઓ ઉમેરાતી જાય છે તેમ તેમ તે ગૂંચવણભર્યું બનતું જાય છે.

ઑબ્જેક્ટ અભિગમ (Object Way) : આપણી પાસે મેનૂ, પિઝા મેકર, વેઈટર, કેશિયર વગેરે જેવા ઑબ્જેક્ટની સ્પષ્ટ યાદી છે. આ બધા પોતપોતાનું કાર્ય ખૂબ સારી રીતે જાણે છે અને જ્યારે સોંપવામાં આવે ત્યારે તે કાર્ય ખૂબ સારી રીતે કરે છે. જો આપણે નવી વસ્તુ ઉમેરીએ, જેમ કે વિવિધ પ્રકારના પિઝા, તો આપણે માત્ર પિઝામેકરની સૂચનાઓ જ અપડેટ કરવાની જરૂર છે. કેશિયર માત્ર રોકડનું સંચાલન કરે છે અને અન્ય વિગતોની દેખરેખ રાખતો નથી. વેઈટર માત્ર ડિલિવરી પર ધ્યાન કેન્દ્રિત કરે છે. આથી, તેમાંથી કોઈ એકની સૂચનાઓમાં ફેરફાર કરવાથી અન્યને અસર થતી નથી.

ઑબ્જેક્ટ ઓરિએન્ટેડ અભિગમ (Object Oriented Concepts)

ઑબ્જેક્ટ એ ઑબ્જેક્ટ ઓરિએન્ટેડ પ્રોગ્રામિંગનો પાયાનો ભાગ છે. વાસ્તવિક દુનિયાની સિસ્ટમ વિવિધ ઑબ્જેક્ટ અને તેમના સંબંધોની બનેલી હોય છે. અગાઉ ચર્ચા કરેલ નાના પરિવારવાળા ઘરના ઉદાહરણને યાદ

રાખો. જોકે ઑબ્જેક્ટ એ ઑબ્જેક્ટ ઓરિએન્ટેડ પ્રોગ્રામિંગના કેન્દ્રમાં છે, પરંતુ સિસ્ટમમાં ઑબ્જેક્ટ અને અન્ય ઑબ્જેક્ટ સાથેના તેમના સંબંધોને લાક્ષણિક બનાવવા માટે આપણે વિવિધ ખ્યાલોને સમજવા પડશે. આનાથી આપણે વાસ્તવિક દુનિયાની સિસ્ટમને યોગ્ય રીતે અને ચોક્કસાઈથી સમજી શકીશું, જેને પછી ઑબ્જેક્ટ ઓરિએન્ટેડ પ્રોગ્રામિંગનો ઉપયોગ કરીને અમલમાં મૂકી શકાય છે.

ઑબ્જેક્ટ ઓરિએન્ટેડ અભિગમમાં નીચેના ખ્યાલોનો સમાવેશ થાય છેઃ

- ઑબ્જેક્ટ (Object)
- ક્લાસ (Class)
- એબ્સ્ટ્રેક્શન (Abstraction)
- એન્કેપ્સ્યુલેશન (Encapsulation) અને ડેટા હાઈડિંગ (Data hiding)
- મેસેજ પાસિંગ (Message passing)
- કમ્પોઝિશન (Composition) અને એગ્રિગેશન (Aggregation)
- ઈન્હેરીટન્સ (Inheritance)
- પોલિમોર્ફિઝમ (Polymorphism)

ચાલો, આપણે એક પછી એક ઉદાહરણો સાથે આ ખ્યાલોની ચર્ચા કરીએ.

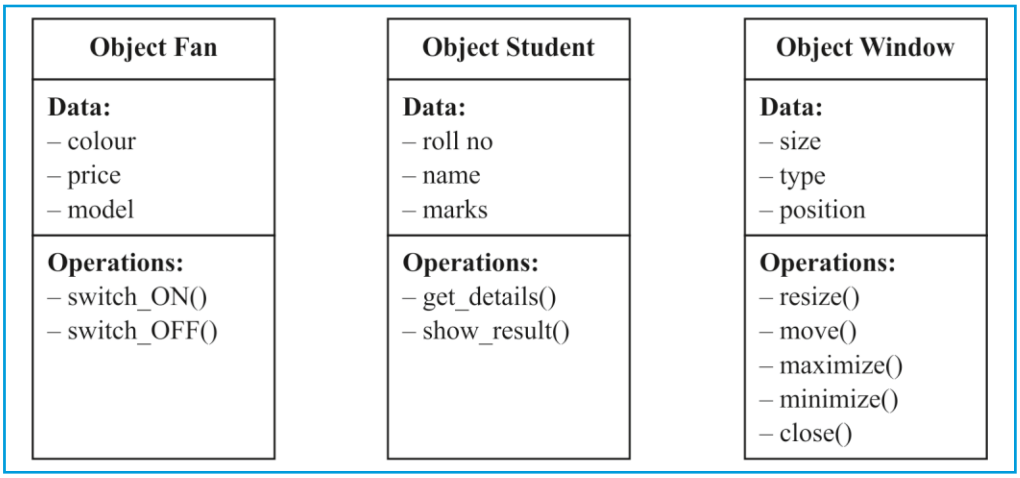
ઑબ્જેક્ટ (Objects)

ઑબ્જેક્ટને વાસ્તવિક દુનિયાની એન્ટીટી તરીકે વ્યાખ્યાયિત કરવામાં આવે છે. દૈનિક જીવનમાં શિક્ષક, સ્કૂલ બેગ, સાયકલ, બ્લેકબોર્ડ, મોબાઈલ ફોન અને અન્ય અનેક એન્ટીટી સાથે કાર્ય કરીએ છીએ. આ બધા ઑબ્જેક્ટનાં ઉદાહરણો છે. વાસ્તવિક દુનિયાના ઑબ્જેક્ટ પ્રોપર્ટી (properties) અને બિહેવિયર (behaviours)ના બનેલા હોય છે.

પ્રોપર્ટી (Properties) : આને ઑબ્જેક્ટ્સની લાક્ષણિકતાઓ (characteristics) અથવા એટ્રિબ્યુટ્સ (attributes) પણ કહેવામાં આવે છે. ઉદાહરણ તરીકે, પંખાના ગુણધર્મો રંગ (colour), કિંમત (price) અને મૉડેલ (model) છે. પ્રોગ્રામિંગમાં ઑબ્જેક્ટની પ્રોપર્ટીને ડેટા (data) તરીકે રજૂ કરવામાં આવે છે. ચાલો વિદ્યાર્થીને એક ઑબ્જેક્ટ તરીકે લઈએ. પ્રોગ્રામમાં તેની પ્રોપર્ટી રોલ નંબરને પૂર્ણાંક (integer) ડેટા પ્રકાર તરીકે, નામને સ્ટ્રિંગ (string) તરીકે અને તેના માર્ક્સ ને ફ્લોટિંગ-પોઈન્ટ (floating-point) નંબર તરીકે રજૂ કરી શકીએ.

બિહેવિયર (Behaviour) : બિહેવિયર એટલે કે ઑબ્જેક્ટને જ્યારે કહેવામાં આવે ત્યારે ઑબ્જેક્ટ દ્વારા કરવામાં આવતી ક્રિયા (action). બિહેવિયરને ઑબ્જેક્ટ પર કરવામાં આવતા ઑપરેશન તરીકે પણ વ્યાખ્યાયિત કરવામાં આવે છે. ઉદાહરણ તરીકે, જ્યારે પંખો ચાલુ થાય છે, ત્યારે તે કાર્ય કરવાનું શરૂ કરશે. તેવી જ રીતે, વિદ્યાર્થી પરીક્ષા આપી શકે છે. પ્રોગ્રામમાં ઑપરેશનને ફંક્શનનો ઉપયોગ કરીને વ્યાખ્યાયિત કરવામાં આવે છે (જેને મેથડ (methods) પણ કહેવામાં આવે છે). જ્યારે આપણને આપેલ ઑબ્જેક્ટ પર કોઈ ચોક્કસ ઑપરેશન કરવાની જરૂર હોય, ત્યારે ફંક્શન અથવા મેથડને કૉલ કરવામાં આવે છે.

ઑબ્જેક્ટને લંબચોરસના રૂપમાં ઑબ્જેક્ટ ડાયાગ્રામનો ઉપયોગ કરીને રજૂ કરી શકાય છે, જેના ત્રણ ભાગ હોય છેઃ ઑબ્જેક્ટનું નામ, પ્રોપર્ટી અને ઑપરેશન. આકૃતિ 11.2 ઑબ્જેક્ટનાં ત્રણ અલગ-અલગ ઉદાહરણો દર્શાવે છે.



આકૃતિ 11.2 : ઑબ્જેક્ટ ડાયાગ્રામ (ઉદાહરણ : પંખો, વિદ્યાર્થી, વિન્ડો)

આકૃતિમાં દર્શાવ્યા પ્રમાણે પંખાને રંગ, કિંમત અને મૉડેલ જેવી તેની પ્રોપર્ટી સાથે વ્યાખ્યાયિત કરવામાં આવ્યો છે, અને તેના ઑપરેશનમાં પંખાને ચાલુ (switch_ON()) અને બંધ (switch_OFF()) કરવાનો સમાવેશ થાય છે. વિદ્યાર્થીને તેની પ્રોપર્ટી રોલ નંબર, નામ અને માર્ક્સ સાથે વ્યાખ્યાયિત કરવામાં આવેલ છે, અને વિદ્યાર્થીની વિગતો મેળવવા (get_details()) અને પરિણામ બતાવવા (show_result()) તે તેના ઑપરેશન છે. ત્રીજું ઉદાહરણ એ કમ્પ્યૂટર સૉફ્ટવેરમાં આવેલી વિન્ડો (Window) ઑબ્જેક્ટનું છે. તેને તેના ગુણધર્મો જેમ કે તેનું કદ, પ્રકાર અને સ્ક્રીન પર તેનું સ્થાન સાથે વ્યાખ્યાયિત કરવામાં આવેલ છે. તેના પર ઑપરેશન જેમ કે, તેનું કદ બદલવું (resize()), અલગ સ્થાન પર ખસેડવું (move()), વિન્ડો મોટી કરવી (maximize()), વિન્ડો નાની કરવી (minimize()), વિન્ડો બંધ કરવી (close()) વગેરે કરી શકાય છે.

જુદી જુદી શ્રેણીઓના ઑબ્જેક્ટના ઉદાહરણો નીચે આપેલા છે:

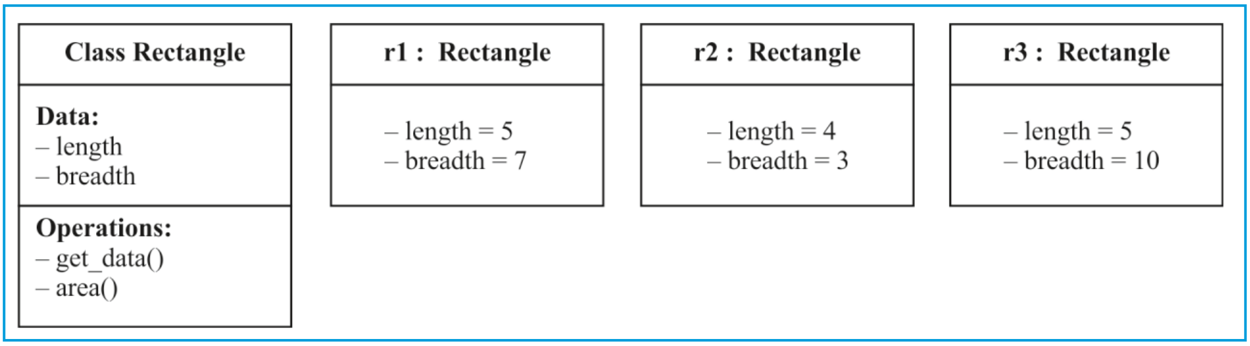
મનુષ્ય	:	વિદ્યાર્થી, શિક્ષક, કર્મચારી
ફર્નિચર	:	પલંગ, સોફા, ટેબલ, ખુરશી, બેન્ચ
ઇલેક્ટ્રોનિક વસ્તુઓ	:	મોબાઇલ, કમ્પ્યૂટર, ટેલિવિઝન, હાર્ડ ડિસ્ક
ભૌમિતિક આકાર	:	બિંદુ, ચોરસ, લંબચોરસ, ત્રિકોણ, સમઘન, ગોળો
GUI ઘટક	:	વિન્ડો, મેનૂ, લિસ્ટ, ચેકબોક્સ, રેડિયો બટન
યુઝર દ્વારા નિર્ધારિત ડેટા	:	સમય, બિંદુ, રેલવે ટિકિટ, વિદ્યાર્થીનું પરિણામ
ઈમેજ ફાઇલ	:	JPG, PNG, GIF
ડોક્યુમેન્ટ ફાઇલ	:	વર્ડ ફાઇલ, સ્પ્રેડશીટ, PDF, ટેક્સ્ટ ફાઇલ, C ફાઇલ

ઑબ્જેક્ટ ઓરિએન્ટેડ પ્રોગ્રામિંગ હંમેશાં એ કેન્દ્રમાં રાખે છે કે આપેલ સમસ્યાને ઑબ્જેક્ટમાં કેવી રીતે વહેંચવામાં આવે (જ્યારે પ્રોસીજરલ પ્રોગ્રામિંગમાં એ કેન્દ્રમાં હોય છે કે સમસ્યાને પ્રક્રિયાઓ એટલે કે ફંક્શન્સમાં કેવી રીતે વહેંચવામાં આવે). ઑબ્જેક્ટની દ્રષ્ટિએ વિચારવું મનુષ્યો માટે સરળ બને છે, કારણ કે આપણે હંમેશાં વાસ્તવિક દુનિયાની સિસ્ટમને એવા ઑબ્જેક્ટ તરીકે સમજીએ છીએ જેની સાથે આપણે સરળતાથી સંવાદ કરી શકીએ છીએ.

ક્લાસ (Class)

ચાલો, એક ઑબ્જેક્ટ - Student પર વિચાર કરીએ. એક વર્ગમાં ઘણા વિદ્યાર્થીઓ હોય છે જે પ્રોપર્ટી અને બિહેવીયર જેવી સમાન વિગતો ધરાવે છે. સમાન પ્રોપર્ટી અને બિહેવીયર ધરાવતા ઑબ્જેક્ટ એક જૂથ અથવા

ક્લાસ બનાવે છે જેને ઑબ્જેક્ટ ક્લાસ (Object Class) તરીકે ઓળખવામાં આવે છે. તેને સામાન્ય રીતે ફક્ત ક્લાસ કહેવાય છે. ક્લાસ તેનામાંથી તારવવામાં આવેલા તમામ ઑબ્જેક્ટ દ્વારા વહેંચાયેલી પ્રોપર્ટી અને ઓપરેશનનું વર્ણન કરે છે. તેથી જ ક્લાસ એક ડિઝાઇન ડોક્યુમેન્ટ અથવા ટેમ્પલેટ તરીકે કાર્ય કરે છે. જેમકે, કારની ડિઝાઇન સમાન હોય તો તેમાંથી ઘણી સમાન કારનું ઉત્પાદન કરી શકાય છે. એકવાર ક્લાસ વ્યાખ્યાયિત થઈ જાય, પછી તેમાંથી ગમે તેટલા ઑબ્જેક્ટ બનાવી શકાય છે. સમાન ક્લાસ ટેમ્પ્લેટમાંથી ઉતરી આવેલા દરેક ઑબ્જેક્ટને તે ક્લાસનો ઇન્સ્ટન્સ (instance) કહેવામાં આવે છે. તેથી, દરેક વ્યક્તિગત ઑબ્જેક્ટ તે ક્લાસનો ઇન્સ્ટન્સ છે જેમાંથી તે ઉતરી આવ્યો છે. ઘણીવાર ક્લાસ અને ઑબ્જેક્ટ શબ્દોનો ઉપયોગ એકબીજાના બદલે થાય છે, પરંતુ સંદર્ભ અર્થ સ્પષ્ટ કરી શકે છે. ક્લાસને ક્લાસ ડાયાગ્રામનો ઉપયોગ કરીને દર્શાવવામાં આવે છે. આકૃતિ 11.3 ક્લાસ અને ઇન્સ્ટન્સનું ઉદાહરણ દર્શાવે છે.



આકૃતિ 11.3 : લંબચોરસ (Rectangle) ક્લાસ અને તેના ઇન્સ્ટન્સ

આકૃતિમાં Rectangle નામનો એક ક્લાસ વ્યાખ્યાયિત કરેલો છે જેમાં length (લંબાઈ) અને breadth (પહોળાઈ) પ્રોપર્ટી અને get_data તથા area ઓપરેશન આપેલાં છે. આકૃતિમાં આ ક્લાસના ત્રણ ઇન્સ્ટન્સ r1, r2, અને r3 દર્શાવેલા છે. જોઈ શકાય છે કે આ ત્રણેય ઇન્સ્ટન્સ સમાન માળખું ધરાવે છે, પરંતુ તેમની લંબાઈ અને પહોળાઈના મૂલ્યોના કારણે તેઓ એકબીજાથી અલગ પડે છે. એક ઇન્સ્ટન્સના બધા મૂલ્યોના સમૂહને તે ઑબ્જેક્ટની સ્થિતિ (state) કહેવામાં આવે છે. તે પણ શક્ય છે કે બે ઇન્સ્ટન્સ માટે બંને ગુણધર્મોના મૂલ્યો સમાન હોય, તેમ છતાં પણ તેમને અલગ ગણી શકાય, કારણ કે કમ્પ્યૂટરમાં બંનેને અલગ મેમરી ફાળવવામાં આવે છે. ચાલો, હવે આપણે C++ માં Rectangle વર્ગને વ્યાખ્યાયિત કરીએ.

```
class Rectangle {
    private:
        /* properties */
        int length;
        int breadth;
    public:
        /* Operations */
        void getData() {
                …
        }
        int area() {
                …
        }
};
```

ગૂંચવણ ટાળવા માટે અહીં અમલની વિગતો આપવામાં આવી નથી. અહીં આપેલો ક્લાસ માત્ર એક ટેમ્પ્લેટની વ્યાખ્યા કરે છે અને તેને કોઈ મેમરી ફાળવવામાં આવી નથી. નીચે આપેલું main() ફંક્શન દર્શાવે છે કે કેવી રીતે ક્લાસમાંથી ઑબ્જેક્ટ બનાવવામાં (instantiate કરવામાં) આવે છે:

```
void main( )
{
    Rectangle r1, r2, r3;
    …
}
```

આ ડેટાના પ્રકાર મુજબ ચલ બનાવ્યા જેવું છે, એટલે કે Rectangle પ્રકારના ઑબ્જેક્ટ બનાવવા. એકવાર ઈન્સ્ટન્સ બનાવીએ, પછી તેને મેમરી ફાળવવામાં આવે છે. તેનો અર્થ એ છે કે ઉપરોક્ત કિસ્સામાં, ત્રણ ઈન્સ્ટન્સ r1, r2 અને r3 માટે અલગ-અલગ મેમરી ફાળવવામાં આવે છે, અને તે બધા સમાન કાર્યો (getData() અને area())ને વહેંચે છે. getData() ઓપરેશનનો ઉપયોગ ઑબ્જેક્ટ માટેની લંબાઈ અને પહોળાઈ મેળવવા માટે થાય છે અને area() નો ઉપયોગ તેની મેમરીમાં સંગ્રહિત લંબાઈ અને પહોળાઈના મૂલ્યોનો ઉપયોગ કરીને આપેલ ઈન્સ્ટન્સનું ક્ષેત્રફળ ગણવા માટે થાય છે.

રોજિંદા જીવનમાં આપણે ઘણી એવી સિસ્ટમનો ઉપયોગ કરીએ છીએ જ્યાં આપણે સમાન ઑબ્જેક્ટના સમૂહનો ઉપયોગ કરીએ છીએ. તમે જે શાળામાં અભ્યાસ કરો છો તે તેનું એક ઉદાહરણ છે. શાળામાં ઘણા શિક્ષકો અને વિદ્યાર્થીઓ હોય છે. શિક્ષકો વિવિધ ધોરણોમાં અલગ-અલગ વિષયો ભણાવે છે. આપણે Teacher, Student, Subjects વગેરે માટે ક્લાસ બનાવી શકીએ છીએ, જેથી તેમના ટેમ્પ્લેટ (એટલે કે તેમની પ્રોપર્ટી અને ઓપરેશન) વ્યાખ્યાયિત કરી શકાય. એકવાર આ ક્લાસ વ્યાખ્યાયિત થઈ જાય, પછી જરૂરિયાત મુજબ આ દરેક ક્લાસના જેટલા જોઈએ તેટલા ઈન્સ્ટન્સ (objects) બનાવી શકીએ છીએ. વિદ્યાર્થીઓને તેમના વર્ગો અને ધોરણો પ્રમાણે જૂથબદ્ધ પણ કરી શકાય છે. શિક્ષક દ્વારા ચોક્કસ વિભાગને ભણાવવામાં આવતા વિષયને સંગ્રહિત કરવા માટે સિસ્ટમમાં teacher-subject-division મેપિંગ પણ બનાવી શકાય છે. સંપૂર્ણ સિસ્ટમને, સિસ્ટમની જરૂરિયાત પ્રમાણે ઉપર જણાવેલ ઑબ્જેક્ટ ઉપરાંત વધારાના ઑબ્જેક્ટની પણ જરૂર પડી શકે છે.

એબ્સ્ટ્રેક્શન (Abstraction)

ઑબ્જેક્ટ ઓરિએન્ટેડ પ્રોગ્રામિંગમાં એબ્સ્ટ્રેક્શન એક ખૂબ જ મહત્વપૂર્ણ ખ્યાલ છે. એબ્સ્ટ્રેક્શનનો અર્થ એ છે કે બિનજરૂરી વિગતો છુપાવવી અને માત્ર જરૂરી વિગતોને જ પ્રકાશિત કરવી. આપણે જાણીએ છીએ કે ઑબ્જેક્ટ તેની પ્રોપર્ટી અને ઓપરેશન દ્વારા વ્યાખ્યાયિત કરવામાં આવે છે. વાસ્તવિક દુનિયામાં એક ઑબ્જેક્ટમાં મોટી સંખ્યામાં પ્રોપર્ટી હોય છે. શું આપણે તે બધાને પ્રોગ્રામમાં ધ્યાનમાં લેવા જોઈએ કે તેમાંથી ફક્ત અમુકને જ? અહીં એબ્સ્ટ્રેક્શન મદદરૂપ થાય છે. આપણે ફક્ત તે જ પ્રોપર્ટીને ધ્યાનમાં લેવાની હોય છે જે આપેલ વાસ્તવિક-વિશ્વની સમસ્યા માટે સોફ્ટવેર સિસ્ટમને અમલમાં મૂકવા માટે જરૂરી હોય. આને ડેટા એબ્સ્ટ્રેક્શન (data abstraction) તરીકે પણ ઓળખી શકાય છે. આ જ બાબત ઓપરેશનને પણ લાગુ પડે છે.

ચાલો, આને એક ઉદાહરણ દ્વારા સમજીએ. આપણે અગાઉના વિભાગમાં Rectangle (લંબચોરસ) વર્ગને ફક્ત length (લંબાઈ) અને breadth (પહોળાઈ) સાથે વ્યાખ્યાયિત કર્યો હતો. જોકે, જો આપણે ગ્રાફિકલ સિસ્ટમ વિકસાવી રહ્યા હોઈએ, તો લંબચોરસમાં અન્ય પ્રોપર્ટી પણ હોઈ શકે છે, જેમ કે તેને રંગથી ભરવા માટે વપરાતો fill-color, position (સ્થાન), thickness of broder (બોર્ડરની જાડાઈ), fill-style (રંગ ભરવાની સ્ટાઈલ) અને અન્ય ઘણા ગુણધર્મો પણ તેમાં હોઈ શકે છે. આ બધા ગુણધર્મો ગ્રાફિકલ સિસ્ટમ માટે આવશ્યક છે, પરંતુ આપણા કિસ્સામાં આપણે ફક્ત ક્ષેત્રફળની ગણતરી કરવા માટે લંબાઈ અને પહોળાઈ મેળવવા માંગતા હતા. તેથી fill-color,

position, thickness, fill-style જેવી પ્રોપર્ટીને છુપાવવામાં આવ્યા હતા. ગ્રાફિકલ સિસ્ટમ માટે, આપણને changeColor (રંગ બદલવો), move (સ્થાન બદલવું) જેવા ઓપરેશનની પણ જરૂર પડે, જે અગાઉના વિભાગમાં છુપાવેલા રાખવામાં આવ્યા હતા.

એબ્સ્ટ્રેક્શન જટિલ સિસ્ટમ્સને સરળ બનાવે છે. તે કોડિંગ સ્તર પર ખૂબ જ સ્પષ્ટતા પૂરી પાડે છે. તે માત્ર જરૂરી વિગતો પર ધ્યાન કેન્દ્રિત કરીને વધુ સારી ડિઝાઈન બનાવવામાં સક્ષમ બનાવે છે. ઑબ્જેક્ટ ઓરિએન્ટેડ પ્રોગ્રામિંગમાં ક્લાસની વ્યાખ્યા પોતે જ એબ્સ્ટ્રેક્શનનું એક ઉદાહરણ છે, કારણ કે તે વાસ્તવિક-વિશ્વની એન્ટિટીને માત્ર આવશ્યક વિગતો સાથે સંક્ષિપ્ત કરે છે.

એન્કેપ્સ્યુલેશન અને ડેટા હાઈડિંગ (Encapsulation and Data Hiding)

આપણે શીખ્યા તે મુજબ ઑબ્જેક્ટ ડેટા (માહિતી) અને ઑપરેશન (મેથડ)નો બનેલો હોય છે. ક્લાસ એક જ યુનિટમાં પ્રોપર્ટી (ડેટા) અને ઑપરેશન (મેથડ) બંનેને વ્યાખ્યાયિત કરે છે, જેનું નામ વાસ્તવિક દુનિયાની એન્ટીટી જેવું જ હોય છે. ડેટા અને ઑપરેશનને એક જ યુનિટમાં જોડવાની પ્રક્રિયાને એન્કેપ્સ્યુલેશન કહેવામાં આવે છે. ક્લાસ ડેટા અને મેથડ બંનેને એક જ નામ હેઠળ એન્કેપ્સ્યુલેટ કરે છે, જેમ કે અગાઉ ચર્ચા કરેલ Rectangle ક્લાસ. એન્કેપ્સ્યુલેશન બે સૌથી મહત્વપૂર્ણ પાસાંઓ રજૂ કરે છે:

- વાસ્તવિક દુનિયાની એન્ટીટીને વ્યાખ્યાયિત કરવા માટે ડેટા અને મેથડને એકસાથે મૂકે છે. આ સિસ્ટમ ડિઝાઈનને ખૂબ જ સ્પષ્ટ બનાવે છે, જે અન્ય કોઈ પણ ઑબ્જેક્ટને અસર કર્યા વિના સિસ્ટમમાં ઑબ્જેક્ટની આંતરિક વિગતો બદલવાની મંજૂરી આપે છે.
- ઑબ્જેક્ટના અમલ જેવા આંતરિક પાસાંને બાહ્ય પાસાં એટલે કે, સિસ્ટમના બાકીના ભાગથી અલગ પાડે છે. તે બાહ્ય દુનિયાથી ડેટા છુપાવે છે અને ઑબ્જેક્ટની મેથડનો ઉપયોગ કરીને જ તેને સુરક્ષિત રીતે ઍક્સેસ કરવાની મંજૂરી આપે છે. ડેટાને બાહ્ય દુનિયાથી છુપાવવાની આ પ્રક્રિયાને ડેટા હાઈડિંગ (Data Hiding) કહેવામાં આવે છે. ઑબ્જેક્ટના ડેટાનો ઉપયોગ કરવા માટે, બાહ્ય દુનિયા, એટલે કે સિસ્ટમના અન્ય ઑબ્જેક્ટ, ટાર્ગેટ ઑબ્જેક્ટની મેથડને કૉલ કરે છે અને ઑબ્જેક્ટના છુપાયેલા ડેટા પર કોઈપણ ઑપરેશન કરવા માટે તેમની સાથે સંવાદ કરે છે.

મેસેજ પાસિંગ (Message Passing)

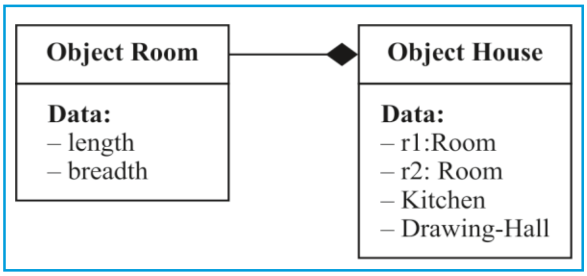
વાસ્તવિક દુનિયાના ઑબ્જેક્ટ કાર્યો કરવા માટે એકબીજાને સંદેશા મોકલીને અને પ્રાપ્ત કરીને સંવાદ કરે છે. આપણે કહી શકીએ કે મેસેજ પાસિંગ વાસ્તવિક દુનિયામાં સંદેશાવ્યવહારની પદ્ધતિનું મોડેલિંગ કરે છે, જે વિવિધ પ્રવૃત્તિઓનું સંચાલન કરે છે. ઑબ્જેક્ટ ઓરિએન્ટેડ પ્રોગ્રામિંગનો ઉપયોગ કરીને વિકસાવવામાં આવેલ સોફ્ટવેર અથવા પ્રોગ્રામ વાસ્તવિક દુનિયાની સમસ્યાનો ઉકેલ છે. તેથી વપરાશકર્તાઓને કાર્યો કરીને સેવાઓ પ્રદાન કરવા માટે પ્રોગ્રામમાં વ્યાખ્યાયિત ઑબ્જેક્ટ્સ વચ્ચે સંદેશાવ્યવહાર માટે તે સમાન પદ્ધતિનો ઉપયોગ કરે છે. એકવાર ક્લાસ વ્યાખ્યાયિત થઈ જાય, પછી આપણે તે ક્લાસમાંથી ઑબ્જેક્ટ્સના ઈન્સ્ટન્સ બનાવી શકીએ છીએ અને પછી ઑબ્જેક્ટ્સ સંદેશા મોકલી અથવા પ્રાપ્ત કરી શકે છે. અગાઉ વ્યાખ્યાયિત કરેલ Rectangle ક્લાસને ધ્યાનમાં લઈએ, જ્યાં *main()* ફંક્શનમાં, આપણે ત્રણ ઈન્સ્ટન્સ r1, r2 અને r3 બનાવ્યા હતા. નીચેનું વિધાન એક મેસેજ પાસિંગનું ઉદાહરણ છે:

```
a = r1.area( );
```

અહીં, એપ્લિકેશન દ્વારા મોકલાયેલો સંદેશ ઑબ્જેક્ટ r1 ક્ષેત્રફળની ગણતરી કરવા માટે પ્રાપ્ત કરે છે. જવાબમાં, એપ્લિકેશન તેના વેરિયેબલ a માં ક્ષેત્રફળનું મૂલ્ય પ્રાપ્ત કરે છે. એક ઑબ્જેક્ટ ઓરિએન્ટેડ પ્રોગ્રામ જરૂરી ક્લાસને વ્યાખ્યાયિત કરે છે, ક્લાસને ઈન્સ્ટન્સિએટ કરીને ઑબ્જેક્ટ્સ બનાવે છે, અને અંતે આ ઑબ્જેક્ટ્સ તેમનું કાર્ય પૂર્ણ કરવા માટેની પ્રવૃત્તિઓ સિદ્ધ કરવા માટે સંદેશા મોકલી અને પ્રાપ્ત કરીને વાતચીત કરે છે.

કમ્પોઝીશન અને એગ્રીગેશન (Composition and Aggression)

ઘણી વખત, વાસ્તવિક દુનિયાનો એક ઑબ્જેક્ટ બીજા ઘણા ઑબ્જેક્ટનો, એટલે કે ભાગનો, બનેલો હોય છે. આવા સંજોગોમાં, ભાગ (part) અને મુખ્ય ઑબ્જેક્ટ (main object) વચ્ચેનો સંબંધ પાર્ટ-ઓફ (part-of) અથવા હેઝ-એ (has-a) પ્રકારનો હોય છે. જો ભાગ અને મુખ્ય ઑબ્જેક્ટ વચ્ચેનો સંબંધ પાર્ટ-ઓફ હોય, તો તેને કમ્પોઝિશન કહેવામાં આવે છે. અને જો ભાગ અને મુખ્ય ઑબ્જેક્ટ વચ્ચેનો સંબંધ હેઝ-એ હોય, તો તેને ઍગ્રિગેશન કહેવામાં આવે છે. ચાલો આપણે વાસ્તવિક દુનિયાના ઉદાહરણો દ્વારા કમ્પોઝિશન અને ઍગ્રિગેશનને સમજીએ.

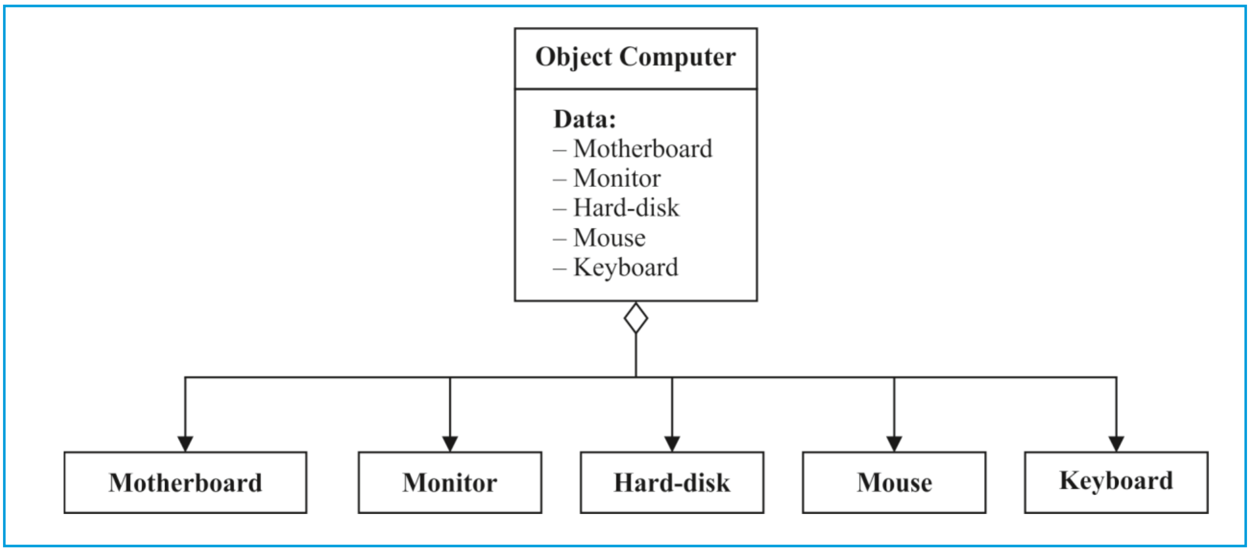


આકૃતિ 11.4 : કમ્પોઝીશનનું ઉદાહરણ

કમ્પોઝિશન (Composition) : આપણે જાણીએ છીએ કે કોઈપણ ઘર એક અથવા વધુ રૂમ, રસોડું, અને ડ્રોઈંગ હૉલનું બનેલું હોય છે. આપણે કહી શકીએ કે ઘર એ રૂમ, રસોડું અને ડ્રોઈંગ હૉલ દ્વારા બનેલું છે. એનો અર્થ એ થયો કે ઘર એક ઑબ્જેક્ટ છે જે અન્ય ઑબ્જેક્ટ (ભાગ)નું બનેલું છે. ઘર એક સંપૂર્ણ ઑબ્જેક્ટ છે, જ્યારે રૂમ, રસોડું અને ડ્રોઈંગ હૉલ તેના ભાગ છે, જે પોતે પણ ઑબ્જેક્ટ છે. આ અભિગમને આકૃતિ 11.4 માં બતાવ્યા પ્રમાણે કમ્પોઝિશનનો ઉપયોગ કરીને મોડેલ કરી શકીએ છીએ. સરળતા માટે અહીં ફક્ત ઘર અને રૂમ બતાવ્યા છે.

આ એક કન્ટેનરશિપ (containership) જેવું છે, જેમ કે ઘર રૂમને સમાવે છે. જો ઘરમાં બે રૂમ હોય, તો આકૃતિમાં બતાવ્યા પ્રમાણે આપણી પાસે ઘર ઑબ્જેક્ટના સભ્ય તરીકે બે રૂમ ઑબ્જેક્ટ હશે. કમ્પોઝિશન દર્શાવવા માટે રંગ ભરેલા ડાયમંડ (solid diamond)નો ઉપયોગ થાય છે. કમ્પોઝિશન એક મજબૂત અથવા વિશિષ્ટ સંબંધ (exclusive relationship) છે, જ્યાં સુધી સંપૂર્ણ ઑબ્જેક્ટ અથવા પેરન્ટ ઑબ્જેક્ટ અસ્તિત્વમાં હોય ત્યાં સુધી જ તેનો ભાગ અસ્તિત્વમાં રહે છે. એનો અર્થ એ છે કે, રૂમ, રસોડું અને ડ્રોઈંગ હૉલ સ્વતંત્ર રીતે અસ્તિત્વ ધરાવતા નથી અને જો ઘર દૂર કરવામાં આવે (નાશ કરવામાં આવે) તો તેઓ પણ દૂર થઈ જાય છે.

ઍગ્રિગેશન (Aggregation) : કમ્પ્યૂટર સિસ્ટમ (computer system) મધરબોર્ડ, મોનિટર, હાર્ડ ડિસ્ક, માઉસ અને કીબોર્ડ જેવા વિવિધ ભાગોનું બનેલું હોય છે. આપણે કહી શકીએ કે કમ્પ્યૂટરની પાસે મધરબોર્ડ છે, મોનિટર છે, હાર્ડ ડિસ્ક છે, માઉસ છે અને કીબોર્ડ છે. આ સંબંધ આકૃતિ 11.5 માં બતાવ્યા પ્રમાણે પોલા ડાયમન્ડ સાથે દર્શાવવામાં આવે છે. સરળતા માટે, ઘટક ઑબ્જેક્ટની પ્રોપર્ટી અહીં દર્શાવવામાં આવી નથી.

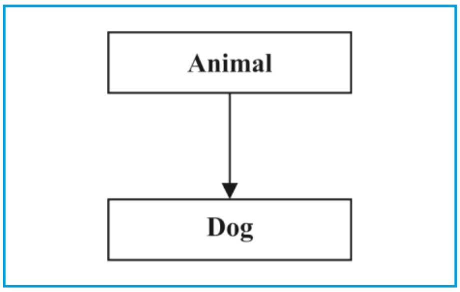


આકૃતિ 11.5 : એગ્રીગેશનનું ઉદાહરણ

એગ્રિગેશન (Aggregation) એ નિર્બળ અથવા નોન-એક્સક્લુઝીવ સંબંધ (non-exclusive relationship) છે, કારણકે તેના ભાગ સંપૂર્ણ ઑબ્જેક્ટ અથવા પેરન્ટ ઑબ્જેક્ટ વિના પણ સ્વતંત્ર રીતે અસ્તિત્વ ધરાવી શકે છે. ઉદાહરણ તરીકે, માઉસ કમ્પ્યૂટર વિના પણ અસ્તિત્વ ધરાવી શકે છે.

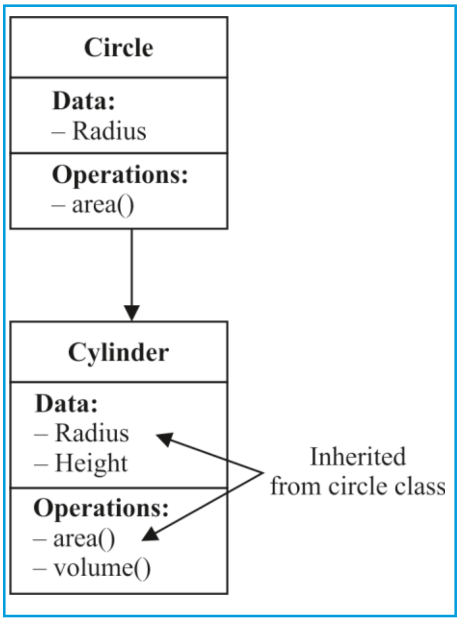
ઈન્હેરીટન્સ (Inheritance)

ઈન્હેરીટન્સ એક એવી પ્રક્રિયા છે જેના દ્વારા એક ક્લાસ બીજા ક્લાસની લાક્ષણિકતાઓ વારસામાં મેળવે છે. જે ક્લાસમાંથી લાક્ષણિકતાઓ વારસામાં મેળવવામાં આવે છે તેને બેઝ ક્લાસ (Base Class) અથવા પેરન્ટ ક્લાસ (Parent Class) કહેવામાં આવે છે. અને જે ક્લાસ લાક્ષણિકતાઓ વારસામાં મેળવે છે તેને ડિરાઈવ્ડ ક્લાસ (Derived Class) અથવા ચાઈલ્ડ ક્લાસ (Child Class) કહેવામાં આવે છે. આકૃતિ 11.6 ઈન્હેરીટન્સ દર્શાવે છે જ્યાં ડોગ (Dog) એ પેરન્ટ ક્લાસ એનિમલ (Animal)નો ચાઈલ્ડ ક્લાસ છે. સ્વાભાવિક છે કે શ્વાન એ પ્રાણીની ઘણી લાક્ષણિકતાઓ વારસામાં મેળવે છે, કારણ કે શ્વાન પણ એક પ્રાણી જ છે. ઈન્હેરીટન્સ "is-a" સંબંધનું પ્રતિનિધિત્વ કરે છે.



આકૃતિ 11.6 : ઈન્હેરીટન્સનું ઉદાહરણ

ઈન્હેરીટન્સ સંબંધિત ઑબ્જેક્ટને એકસાથે જૂથબદ્ધ કરવાની મંજૂરી આપે છે, જે કેટલીક સામાન્ય લાક્ષણિકતાઓ વહેંચે છે. તે આપણને ઑબ્જેક્ટ ઓરિએન્ટેડ પ્રોગ્રામિંગમાં હાલના કોડને વિસ્તૃત કરવા (extend), પુનઃઉપયોગ કરવા (reuse) અને સુધારવા (improve) ની મંજૂરી આપે છે. આકૃતિ 11.7માં બતાવ્યા પ્રમાણે વર્તુળ/સર્કલ (Circle) અને નળાકાર/સિલિન્ડર (Cylinder) ક્લાસનું નીચેનું ઉદાહરણ ધ્યાનમાં લો.

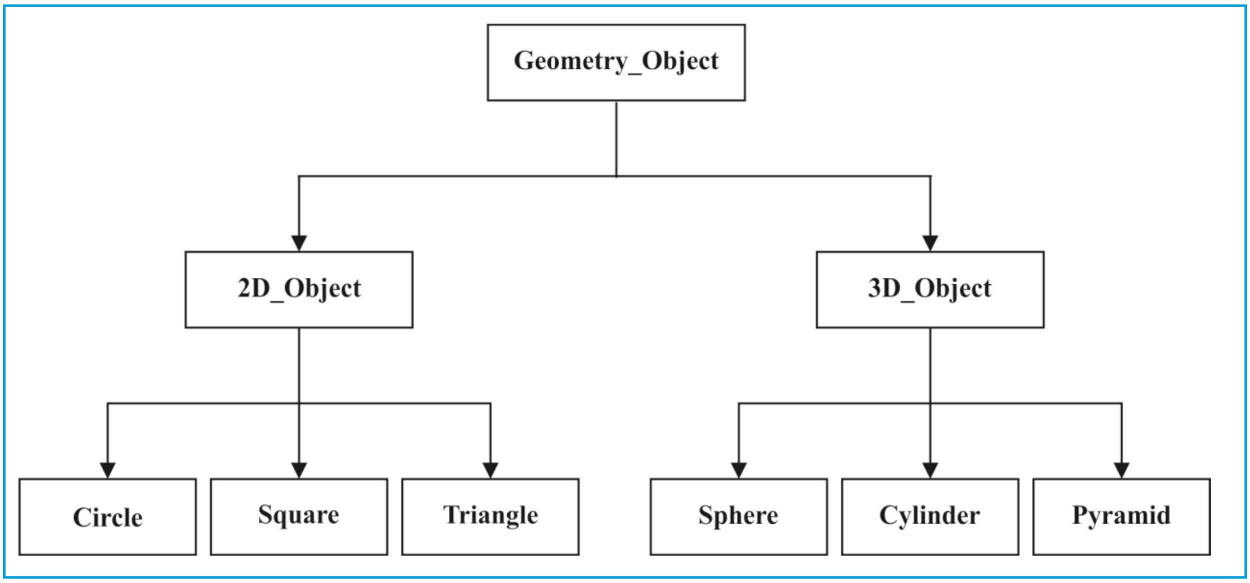


આકૃતિ 11.7 : ઈન્હેરીટન્સ દ્વારા વિસ્તૃતીકરણ અને પુનઃઉપયોગ

Circle ક્લાસ Radius પ્રોપર્ટી અને area() ઑપરેશન સાથે વ્યાખ્યાયિત થયેલ છે. તેને સિલિન્ડર ક્લાસ (Cylinder class) સુધી વિસ્તૃત કરવામાં આવે છે, જેમાં પ્રોપર્ટી તરીકે Radius (જે સર્કલ ક્લાસમાંથી વારસામાં મળે છે) અને Height ઉમેરવામાં આવે છે. આ એક વિસ્તરણ (extension)નું ઉદાહરણ છે, કારણ કે સિલિન્ડર ક્લાસ સર્કલમાં એક વધુ પ્રોપર્ટી ઉમેરીને તેને વિસ્તૃત કરે છે. area() ઑપરેશન સર્કલ ક્લાસમાંથી વારસામાં મળે છે, જેનો ઉપયોગ સિલિન્ડર ક્લાસમાં ઉમેરવામાં આવેલા volume() ઑપરેશનને વ્યાખ્યાયિત કરવા માટે થઈ શકે છે, કારણ કે સિલિન્ડરનું ઘનફળ $\pi r^2 h$ છે, જ્યાં πr^2 ક્ષેત્રફળ દર્શાવે છે. સર્કલ પહેલેથી જ ક્ષેત્રફળ ઑપરેશનને વ્યાખ્યાયિત કરે છે, જેનો ઘનફળની ગણતરી કરવા માટે પુનઃઉપયોગ કરવામાં આવ્યો છે.

ઈન્હેરીટન્સ એ વાસ્તવિક જીવનમાં વ્યાપકપણે ઉપયોગમાં લેવાતો ખ્યાલ છે, જ્યાં આપણે એક મેઈન ક્લાસને સબક્લાસમાં વિભાજિત કરીએ છીએ. બધા સબક્લાસ કેટલીક સામાન્ય લાક્ષણિકતાઓ (પ્રોપર્ટી અને ઑપરેશન્સ) વહેંચે છે, જ્યારે તેમની પોતાની અનન્ય લાક્ષણિકતાઓ પણ ધરાવે છે. સામાન્ય લાક્ષણિકતાઓ ટોચના ક્લાસ (top class)માં વ્યાખ્યાયિત કરવામાં આવે છે, જે બધા સબક્લાસ દ્વારા વહેંચાય છે. સબક્લાસને સમાન રીતે તેમના પણ સબક્લાસમાં આગળ વિભાજિત કરી શકાય છે અને તે આગલા સ્તર માટે બેઝ ક્લાસ

તરીકે કાર્ય કરે છે. આકૃતિ 11.8 ભૌમિતિક ઑબ્જેક્ટ (geomatry objects)નું ઉદાહરણ દર્શાવે છે, જેને 2D ઑબ્જેક્ટ અને 3D ઑબ્જેક્ટમાં વિભાજિત કરવામાં આવે છે. 2D ઑબ્જેક્ટને આગળ વર્તુળ (Circle), ચોરસ (Square) અને ત્રિકોણ (Triangle) જેવા આકારોમાં વિભાજિત કરવામાં આવે છે, અને 3D ઑબ્જેક્ટને ગોળો (Sphere), નળાકાર (Cylinder) અને પિરામિડ (Pyramid)માં વિભાજિત કરવામાં આવે છે.



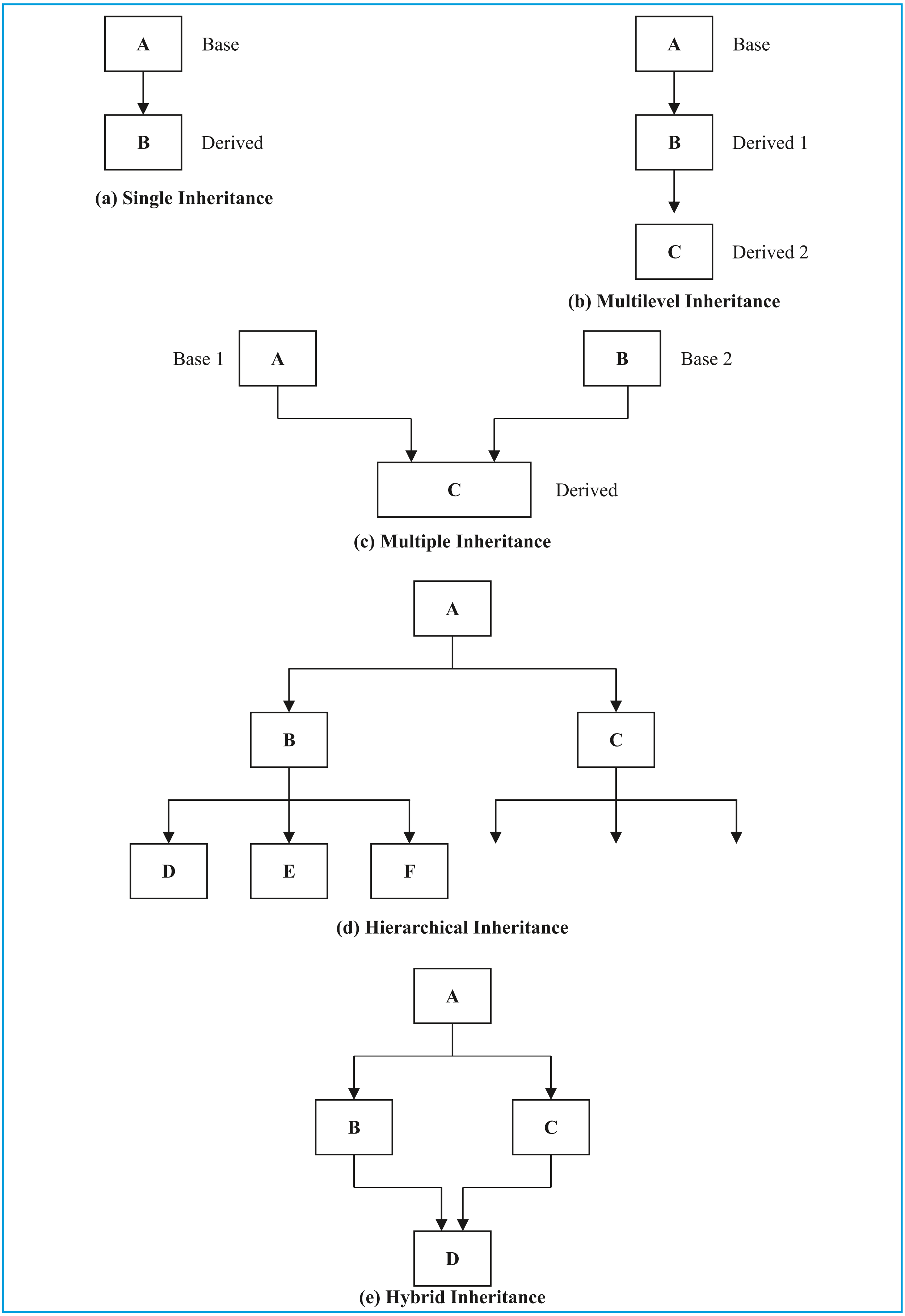
આકૃતિ 11.8 : ભૌમિતિક ઑબ્જેક્ટનો વંશવેલો

આકૃતિમાં દર્શાવ્યા મુજબ, Geometry_Object ક્લાસમાં એવી વિશેષતાઓ સામેલ છે જે 2D_Object અને 3D_Object બંને ક્લાસ દ્વારા વહેંચાયેલી છે. 2D_Object ક્લાસ દ્વારા વ્યાખ્યાયિત વિશેષતાઓ Circle, Square અને Triangle માટે સામાન્ય છે, અને તેમની પોતાની વિશેષતાઓ પણ છે.

ઑબ્જેક્ટ વચ્ચેના વિવિધ સંબંધોને ધ્યાનમાં લેતા, આપણે પાંચ જુદા જુદા પ્રકારના ઈન્હેરીટન્સને વ્યાખ્યાયિત કરી શકીએ છીએ:

- સિંગલ ઈન્હેરીટન્સ (Single Inheritance)
- મલ્ટીલેવલ ઈન્હેરીટન્સ (Multilevel Inheritance)
- મલ્ટીપલ ઈન્હેરીટન્સ (Multiple Inheritance)
- હાઈરાર્કીકલ ઈન્હેરીટન્સ (Hierarchical Inheritance)
- હાઈબ્રિડ ઈન્હેરીટન્સ (Hybrid Inheritance)

આકૃતિ 11.9 આ પાંચેય પ્રકારના વારસાગત સંબંધો દર્શાવે છે. જ્યારે એક ક્લાસ બીજા ક્લાસમાંથી તારવવામાં (derived) આવે છે, ત્યારે તેને સિંગલ ઈન્હેરીટન્સ કહેવામાં આવે છે. તે પિતા અને બાળક (parent-child) વચ્ચેના સંબંધને રજૂ કરે છે. મલ્ટીલેવલ ઈન્હેરીટન્સમાં એક ક્લાસ એવા અન્ય ક્લાસમાંથી ઉત્પન્ન થાય છે જે પોતે પણ મૂળભૂત ક્લાસ (base class) માંથી ઉત્પન્ન થયો હોય. તે દાદા, પિતા અને બાળક (grandfather-parent-child) વચ્ચેના સંબંધને રજૂ કરે છે. મલ્ટીપલ ઈન્હેરીટન્સ બે કે તેથી વધુ મૂળભૂત ક્લાસનો ઉપયોગ કરીને એક નવો ક્લાસ ઉત્પન્ન કરે છે, જેમાં બંનેની વિશેષતાઓનું સંયોજન હોય છે. ઉદાહરણ તરીકે, આપણે કાર (car) અને વિમાન (plane) ની વિશેષતાઓને જોડીને એર-ટેક્સી (air_taxi) બનાવી શકીએ છીએ. આપણે ઘણા કુદરતી પદાનુક્રમો તેમજ સંસ્થાઓ અને વ્યવસાયોમાં પદાનુક્રમોનો ઉપયોગ કરીએ છીએ. તેને હાઈરાર્કીકલ સંબંધનો ઉપયોગ કરીને રજૂ કરવામાં આવે છે. જ્યારે કોઈ સંબંધ રજૂ કરવા માટે એક કરતા વધુ પ્રકારના વારસાગત સંબંધનો ઉપયોગ કરવામાં આવે છે, ત્યારે તેને હાઈબ્રિડ વારસાગત સંબંધ કહેવામાં આવે છે.

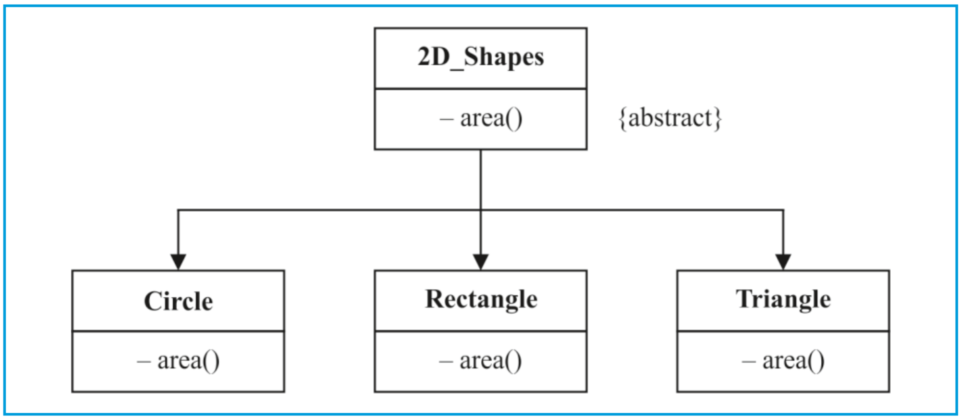


આકૃતિ 11.9 : ઈન્હેરીટન્સના પ્રકાર

પૉલિમૉર્ફિઝમ (Polymorphism)

પૉલિમૉર્ફિઝમનો અર્થ છે અનેક સ્વરૂપો (many forms). તેને એક જ વસ્તુ દ્વારા બહુવિધ વર્તન (multiple behaviour) તરીકે પણ વર્ણવી શકાય છે. આપણે C પ્રોગ્રામિંગમાં + ગણિતીય ઑપરેટરનો ઉપયોગ integer, float, double વગેરે જેવા વિવિધ ડેટા પ્રકાર સાથે કર્યો છે. ઑબ્જેક્ટ-ઑરિએન્ટેડ પ્રોગ્રામિંગ આ + ઑપરેટરને યુઝર દ્વારા વ્યાખ્યાયિત ઑબ્જેક્ટ સાથે પણ કાર્ય કરવા માટે વિસ્તારે છે. આને ઑપરેટર ઓવરલોડિંગ (operator overloading) કહેવામાં આવે છે, કારણ કે ઑપરેટરને યુઝર દ્વારા વ્યાખ્યાયિત ઑબ્જેક્ટ સાથે કામ કરવા સક્ષમ બનાવવામાં આવે છે. તેનો અર્થ એ છે કે, આ જ ઑપરેટર જુદા જુદા ઑબ્જેક્ટ માટે અલગ રીતે વર્તે છે, કારણકે સરવાળાની પ્રક્રિયા દરેક ઑબ્જેક્ટ પ્રમાણે બદલાય છે. આ રીતે, ઑપરેટર + ના ઘણા સ્વરૂપો છે. આથી, ઑપરેટર ઓવરલોડિંગ એ પૉલિમૉર્ફિઝમનું એક ઉદાહરણ છે. ઑબ્જેક્ટ-ઑરિએન્ટેડ પ્રોગ્રામિંગમાં બીજું ઉદાહરણ ફંક્શન ઓવરલોડિંગ (function overloading) છે, જે સમાન નામવાળા પરંતુ જુદા જુદા કાર્યો કરતા બે અથવા વધુ ફંક્શનને મંજૂરી આપે છે. ઑપરેટર ઓવરલોડિંગ અને ફંક્શન ઓવરલોડિંગ બંને કમ્પાઇલ ટાઇમ પૉલિમૉર્ફિઝમ (compile time polymorphism)ના ઉદાહરણો છે, કારણ કે ઘણા બિહેવિયરમાંથી કોઈ એકને પસંદ કરવાનો અંતિમ નિર્ણય પ્રોગ્રામનું કમ્પાઇલિંગ કરતી વખતે એટલે કે કમ્પાઇલ ટાઇમ પર લેવામાં આવે છે.

ઑબ્જેક્ટ-ઑરિએન્ટેડ પ્રોગ્રામિંગમાં ઇન્હેરીટન્સ અને પૉલિમૉર્ફિઝમ બંને સાથે મળીને એક મહત્વપૂર્ણ ભૂમિકા ભજવે છે. આકૃતિ 11.10માં આપેલ વારસાગત સંબંધના પદાનુક્રમને ધ્યાનમાં લો, જે 2D_Shapes નામના બેઝ ક્લાસને તેના સબક્લાસ Circle, Rectangle અને Triangle સાથે રજૂ કરવામાં આવે છે.



આકૃતિ 11.10 : રન ટાઈમ પૉલિમૉર્ફિઝમ

આપણે જાણીએ છીએ કે દરેક આકારનું ક્ષેત્રફળ હોય છે, જે વિવિધ આકારો માટે અલગ અલગ રીતે ગણવામાં આવે છે. બધા આકારો આ સામાન્ય ઑપરેશનને વહેંચે છે, તેથી તેનો ઉલ્લેખ 2D_Shapes નામના બેઝ ક્લાસમાં કરવામાં આવ્યો છે. જોકે, તેની વ્યાખ્યા દરેક ઑબ્જેક્ટ માટે અલગ હોય છે. આ કારણોસર, area() ઑપરેશન બેઝ ક્લાસમાં એબ્સ્ટ્રેક્ટ (abstract) છે; એટલે કે, માત્ર તેનું ઇન્ટરફેસ (interface) પૂરું પાડવામાં આવે છે, પરંતુ તે દરેક સબક્લાસ દ્વારા તેની પોતાની વ્યાખ્યા સાથે અમલમાં મૂકવામાં આવે છે. ઉદાહરણ તરીકે, Circle ક્લાસ πr^2 નો ઉપયોગ કરીને *area()*નો અમલ કરે છે, Rectangle ક્લાસ length*breadth નો ઉપયોગ કરીને અમલ કરે છે અને Triangle ક્લાસ (height*base)/2 નો ઉપયોગ કરીને અમલ કરે છે. આ એક પ્રશ્ન ઊભો કરે છે કે જ્યારે આપણે 2D_shape ઑબ્જેક્ટ પર ક્ષેત્રફળની ગણતરી કરવા માંગીએ છીએ, ત્યારે ત્રણ અમલીકરણોમાંથી શેનો ઉપયોગ કરવો? જ્યારે પ્રોગ્રામ રન કરવામાં આવે છે ત્યારે કે આ આકાર વર્તુળ (circle), લંબચોરસ (rectangle) કે ત્રિકોણ (triangle)માંથી કયો છે એ માહિતી ઉપલબ્ધ થશે. આથી,

આને રન-ટાઇમ પૉલિમૉર્ફિઝમ (run-time polymorphism) તરીકે ઓળખવામાં આવે છે. આ એક ખૂબ જ શક્તિશાળી પદ્ધતિ છે જે ડાયનેમિઝમ (dynamism) અને ફ્લેક્સીબિલિટી (flexibility) પ્રદાન કરે છે.

ઑબ્જેક્ટ ઓરિએન્ટેડ પ્રોગ્રામિંગના લાભ (Benefits of Object Oriented Programming)

ઑબ્જેક્ટ-ઑરિએન્ટેડ પ્રોગ્રામિંગ (OOP) જાળવણીમાં સરળતા અને ખર્ચ-અસરકારકતા (cost-effectiveness) સાથે ગુણવત્તાયુક્ત સોફ્ટવેર બનાવવા માટે ઘણા ફાયદાઓ પ્રદાન કરે છે. તેના મુખ્ય લાભ નીચે મુજબ છે:

- તે વાસ્તવિક દુનિયાની વસ્તુઓનું મૉડલ બનાવે છે, જે સિસ્ટમમાં સ્પષ્ટ ડિઝાઇન અને સીમાઓને કારણે જટિલ સિસ્ટમને કુદરતી રીતે અને સરળતાથી સમજવાની મંજૂરી આપે છે.
- તે સંબંધિત ડેટા અને ક્રિયાઓને ક્લાસ નામના એક યુનિટમાં એકસાથે જોડે છે, જેનાથી કોડ વાંચવામાં અને તેનું સંચાલન કરવામાં સરળતા રહે છે.
- એક જ ક્લાસનો ઉપયોગ અન્ય પ્રોગ્રામમાં અથવા તે જ પ્રોગ્રામના જુદા જુદા ભાગમાં થઈ શકતો હોવાથી પુનઃઉપયોગિતા વધે છે.
- એક ક્લાસને પ્રોગ્રામના અન્ય ભાગોને અસર કર્યા વિના અપડેટ કરી અથવા બદલી શકાય છે, જેનાથી ફેરફારો સરળ બને છે.
- એન્કેપ્સ્યુલેશન ડેટાને સુરક્ષિત બનાવે છે અને ડેટા સંરક્ષણમાં વધારો કરીને માત્ર સુરક્ષિત રીતે ઉપયોગ કરવાની મંજૂરી આપે છે.
- ઇન્હેરીટન્સ વાસ્તવિક દુનિયાના સંબંધોને સરળતાથી રજૂ કરે છે અને કોડનું પુનરાવર્તન ટાળે છે, કારણ કે તે એક ક્લાસને બીજા ક્લાસની વિશેષતાઓનો ઉપયોગ કરવાની મંજૂરી આપે છે.
- પૉલિમૉર્ફિઝમ જુદા જુદા ઑબ્જેક્ટ્સ માટે સમાન ફંક્શનને અલગ રીતે અમલમાં મૂકવાની મંજૂરી આપે છે, જે કોડને વધુ ડાયનેમિક અને શક્તિશાળી બનાવે છે.
- તે વધુ ક્લાસ ઉમેરીને અથવા વારસાગત સંબંધોનો ઉપયોગ કરીને હાલના ક્લાસનું વિસ્તરણ કરીને સ્કેલેબિલિટીને સમર્થન આપે છે.
- ટીમ સરળતાથી સહયોગ કરી શકે છે, કારણ કે વિવિધ સભ્યો કોઈપણ મૂંઝવણ વિના જુદા જુદા ક્લાસ વિકસાવી શકે છે.
- તમામ આધુનિક ભાષાઓ OOPને સમર્થન આપે છે, જે ડેવલપર માટે વધુ વિકલ્પો પ્રદાન કરે છે.

ઑબ્જેક્ટ ઓરિએન્ટેડ ભાષાઓ (Object Oriented Languages)

ઑબ્જેક્ટ-ઑરિએન્ટેડ ભાષાઓ એ એવી પ્રોગ્રામિંગ ભાષાઓ છે જે સમગ્ર પ્રકરણમાં ચર્ચા કરાયેલા ઑબ્જેક્ટ-ઑરિએન્ટેડ ખ્યાલોને સમર્થન આપે છે. ઑબ્જેક્ટ-ઑરિએન્ટેડ ભાષાઓ યુઝરને વાસ્તવિક દુનિયાની સિસ્ટમનું મૉડલ બનાવવા માટે ઇન્હેરીટન્સ અને પૉલિમૉર્ફિઝમ સાથે ઑબ્જેક્ટ અને ક્લાસના સંદર્ભમાં મોટી અને જટિલ એપ્લિકેશનનું આયોજન કરવાની મંજૂરી આપે છે. તેના કારણે વાસ્તવિક દુનિયાની ઘટનાઓ સાથેની તેની સમાનતાને કારણે તેની સમજણ અને જાળવણી સરળ બને છે. કોઈપણ ભાષાને ઑબ્જેક્ટ-ઑરિએન્ટેડ તરીકે લાયક ઠરવા માટે, તેણે એન્કેપ્સ્યુલેશન, ઇન્હેરીટન્સ અને પૉલિમૉર્ફિઝમને સમર્થન આપવું આવશ્યક છે. ચાલો, લોકપ્રિય ઑબ્જેક્ટ-ઑરિએન્ટેડ ભાષાઓ વિષે જાણકારી મેળવીએ.

C++

C++ એ ખૂબ જ જૂની અને લોકપ્રિય ઑબ્જેક્ટ-ઑરિએન્ટેડ ભાષા છે, જેની રચના 1980ના દાયકાની શરૂઆતમાં યુએસએ (USA) માં આવેલી AT&T બેલ લેબોરેટરીઝમાં બજાર્ને સ્ટ્રોસ્ટૂપ (Bjarne Stroustrup) દ્વારા કરવામાં આવી હતી. Simula67 એ પ્રથમ ઑબ્જેક્ટ-ઑરિએન્ટેડ ભાષા હતી, જે ઓલે-જોહાન ડાહલ (Ole-Johan Dahl) અને ક્રિસ્ટન નાયગાર્ડ (Kristen Nygaard) દ્વારા નોર્વેમાં ટ્રાફિક અથવા ફેક્ટરી પ્રક્રિયાઓ જેવી વાસ્તવિક દુનિયાની સિસ્ટમનું અનુકરણ કરવા માટે વિકસાવવામાં આવી હતી. તે જ સમયે, C તેની સરળતા, ફ્લેક્સીબિલિટી અને કાર્યક્ષમતાને કારણે સિસ્ટમ પ્રોગ્રામિંગમાં વ્યાપકપણે ઉપયોગમાં લેવાતી સૌથી લોકપ્રિય ભાષા હતી. સ્ટ્રોસ્ટૂપ Simula67 અને C ભાષા બંનેની વિશેષતાઓને જોડવા માંગતા હતા. તેમણે C ભાષાનું વિસ્તરણ Simula67 માંથી લેવાયેલા ક્લાસ સાથે કર્યું અને તેનું નામ 'C with classes' રાખ્યું. પાછળથી તેનું નામ બદલીને C++ રાખવામાં આવ્યું, જેમાં ઇંક્રિમેન્ટ ઑપરેટર ++ નો વિચાર કરીને સૂચવવામાં આવ્યું કે C++ એ C + 1 છે, એટલે કે તે પ્રોસિજરલ અને ઑબ્જેક્ટ-ઑરિએન્ટેડ બંને અભિગમોને સમર્થન પૂરાં પાડે છે. C++ એ ગેમ્સ, ઑપરેટિંગ સિસ્ટમ અને ટૂલ્સ વિકસાવવા માટે વિદ્યાર્થીઓ, એન્જિનિયરો અને ઉદ્યોગમાં ખૂબ જ લોકપ્રિય રહી છે.

C++ ને C ભાષાની લોકપ્રિયતા અને વ્યાપક ઉપયોગનો ફાયદો મળ્યો છે, કારણ કે C++ OOP વિશેષતાઓ સાથે C સિન્ટેક્સને અનુસરે છે, જે શીખવાની પ્રક્રિયાને ઝડપી બનાવે છે. આજે, સિન્ટેક્સની દ્રષ્ટિએ મોટા ભાગની આધુનિક ઑબ્જેક્ટ-ઑરિએન્ટેડ પ્રોગ્રામિંગ ભાષાઓ C ભાષામાંથી ઉતરી આવેલી છે, અને તે C++ જેવી જ ઑબ્જેક્ટ-ઑરિએન્ટેડ સુવિધાઓને સમર્થન આપે છે. આ જ કારણ છે કે આજે પણ C++ ઑબ્જેક્ટ-ઑરિએન્ટેડ ખ્યાલ અને પ્રોગ્રામિંગ શીખવા માટે પ્રથમ પસંદગી રહી છે.

જાવા (Java)

જાવા એ અન્ય એક લોકપ્રિય ઑબ્જેક્ટ-ઑરિએન્ટેડ પ્રોગ્રામિંગ ભાષા છે, જે 1990ના દાયકાના મધ્યમાં સન માઇક્રોસિસ્ટમ્સ (Sun Microsystems)માં જેમ્સ ગોસલિંગ (James Gosling) અને તેમની ટીમ દ્વારા વિકસાવવામાં આવી હતી. જાવા સંપૂર્ણપણે ઑબ્જેક્ટ-ઑરિએન્ટેડ છે અને દરેક વસ્તુ ઑબ્જેક્ટની આસપાસ રહે છે. જાવાની સૌથી આકર્ષક વિશેષતા છે ''કોડ એકવાર લખો અને ગમે ત્યાં ચલાવો'' (write code once and run it anywhere). એટલે કે, એક સિસ્ટમ પર લખેલો કોડ ફરીથી કમ્પાઇલ કર્યા વિના કે તેમાં ફેરફાર કર્યા વિના કોઈપણ અન્ય સિસ્ટમ પર વાપરી શકાય છે. તેને પ્લેટફોર્મ-સ્વતંત્ર (platform independent) પણ કહેવામાં આવે છે, કારણ કે તે કોઈપણ હાર્ડવેર અને ઑપરેટિંગ સિસ્ટમ સંયોજન પર ફેરફાર વિના કામ કરે છે. જાવાની મુખ્ય વિશેષતાઓમાં ઑબ્જેક્ટ-ઑરિએન્ટેડ, પ્લેટફોર્મ-સ્વતંત્ર, સુરક્ષિત (secure), મજબૂત (robust), ડાયનેમિક, વિતરિત (distributed) અને મલ્ટિથ્રેડેડ (multithreaded)નો સમાવેશ થાય છે. જાવાનો સૌથી મોટો ફાયદો એ છે કે તે નાના અને સરળથી લઈને મોટા અને જટિલ વિવિધ પ્રકારના એપ્લિકેશન્સ માટે વિવિધ લાઇબ્રેરીઓના સંદર્ભમાં મોટો સપોર્ટ પૂરો પાડે છે. આનાથી વિકાસ ઝડપી બને છે, કારણ કે ડેવલપરે ફક્ત એપ્લિકેશનના મુખ્ય તર્ક પર જ ધ્યાન કેન્દ્રિત કરવું પડે છે. તે સ્ટાન્ડર્ડ, એન્ટરપ્રાઇઝ અને મોબાઇલ એપ્લિકેશન ડેવલપમેન્ટને સપોર્ટ કરે છે, જે તેને ઉદ્યોગ માટે આકર્ષક બનાવે છે.

પાયથોન (Python)

પાયથોન (Python) ખૂબ જ સરળતાથી શીખી શકાય તેવી આધુનિક પ્રોગ્રામિંગ ભાષા છે, જે ઑબ્જેક્ટ-ઑરિએન્ટેડ પ્રોગ્રામિંગને સંપૂર્ણપણે સમર્થન આપે છે. તે 1990ના દાયકાની શરૂઆતમાં એમ્સ્ટરડેમમાં ગાઇડો વેન રોસમ (Guido van Rossum) દ્વારા એક સરળ ધ્યેય સાથે વિકસાવવામાં આવી હતીઃ પ્રોગ્રામિંગને વધુ સરળ અને વધુ વાંચનક્ષમ બનાવવું. તે અંગ્રેજી જેવી લાગે છે, જે તેને શાળાના વિદ્યાર્થીઓ માટે પણ યોગ્ય બનાવે છે. પાયથોન ડેટા સાયન્સ (Data Science) અને આર્ટિફિશિયલ ઇન્ટેલિજન્સ/મશીન લર્નિંગ (AI/

Machine Learning) માટે સર્વસંમત પ્રોગ્રામિંગ ભાષા તરીકે સ્વીકારવામાં આવી છે, કારણ કે તેને તેના માટે સમૃદ્ધ લાઇબ્રેરીઓનું સમર્થન મળે છે. તેમાંની કેટલીક નીચે આપવામાં આવી છેઃ

- NumPy : સંખ્યાઓ, એરે (arrays) સાથે કામ કરવા અને ગણિતની પ્રક્રિયા ઝડપથી કરવા માટે.
- Pandas : ટેબલનું સંચાલન કરવા અને અવ્યવસ્થિત (messy) ડેટાને દૂર કરવા માટે.
- Matplotlib : ચાર્ટ્સ અને ગ્રાફ્સ એટલે કે વિઝ્યુલાઇઝેશન માટે.
- Scikit-learn : આગાહી અને વર્ગીકરણ જેવા મશીન લર્નિંગ મૉડલ્સ બનાવવા માટે.
- TensorFlow : ડીપ લર્નિંગ અને એડવાન્સ્ડ AI માટે.

આવા વ્યાપક લાઇબ્રેરી સપોર્ટને કારણે પાયથોનનો ઉપયોગ ફેસ રેકગ્નેશન, રેકોમેન્ડેશન સિસ્ટમ્સ, ટ્યુમર ડિટેક્શન જેવી હેલ્થકેર એપ્લિકેશન્સ સહિત તમામ પ્રકારના AI/ML એપ્લિકેશન ડેવલપમેન્ટ માટે થાય છે. જનરેટિવ AI ખૂબ જ શક્તિશાળી LLMs (Large Language Models), જેમ કે ChatGPT, Claude, Gemini, LLaMA વગેરેની ઉપલબ્ધતાને કારણે વધુને વધુ લોકપ્રિય બની રહ્યું છે. LLMs બનાવવા માટે પણ પાયથોન સૌથી યોગ્ય પ્રોગ્રામિંગ ભાષા છે.

ઉપર જણાવેલ લોકપ્રિય ઑબ્જેક્ટ-ઑરિએન્ટેડ પ્રોગ્રામિંગ ભાષાઓ ઉપરાંત, સૉફ્ટવેર ઉદ્યોગ નીચેની OOPને સમર્થન આપતી ભાષાઓનો પણ વ્યાપકપણે ઉપયોગ કરે છેઃ

જાવાસ્ક્રિપ્ટ (JavaScript) : વેબ ડેવલપમેન્ટ માટે વપરાય છે. તે ઑબ્જેક્ટ આધારિત છે, બ્રાઉઝરમાં ચાલે છે અને ડાયનેમિક કન્ટેન્ટ પૂરૂ પાડે છે.

C # (સી શાર્પ) : માઇક્રોસોફ્ટ દ્વારા વિન્ડોઝ એપ્લિકેશન ડેવલપમેન્ટ અને ટૂલ્સ માટે વિકસાવવામાં આવી છે.

પીએચપી (PHP) : વેબ ડેવલપમેન્ટ માટે વપરાય છે. સર્વર બાજુની સ્ક્રિપ્ટિંગ માટે સારી છે.

સારાંશ

પ્રોસિજરલ પ્રોગ્રામિંગ કોડને ક્રમબદ્ધ પગલાની સૂચનાઓ તરીકે ગોઠવે છે, ડેટા અને ફંક્શનને એકબીજા સાથે ભેળવી દે છે અને પ્રતિબંધો વિના ડેટાને ઍક્સેસ કરવાની મંજૂરી આપે છે. તે પ્રોગ્રામને મોટો અને જટિલ બન્યા પછી કોડનું સંચાલન, અપડેટ અને તેમાં રહેલી સમસ્યાઓના નિવારણ જેવા કાર્યો મુશ્કેલ બનાવે છે. ઑબ્જેક્ટ-ઑરિએન્ટેડ પ્રોગ્રામિંગ કોડને વાસ્તવિક દુનિયાની વસ્તુઓની જેમ ઑબ્જેક્ટ તરીકે ગોઠવીને આ તમામ સમસ્યાઓનું નિરાકરણ લાવે છે. કોડને ઑબ્જેક્ટ્સના સંદર્ભમાં ગોઠવવાથી તે વાસ્તવિક-દુનિયાની સિસ્ટમ સાથે ગાઢ સામ્યતાને કારણે અને ખાસ કરીને સિસ્ટમ મોટી અને જટિલ હોય ત્યારે, તેનું સંચાલન અને સુધારા સરળ બને છે. ઑબ્જેક્ટમાં પ્રોપર્ટી (ડેટા) અને બિહેવીયર (ફંક્શન અથવા મેથડ) હોય છે, અને સમાન ડેટા અને મેથડ વહેંચતા ઑબ્જેક્ટનો સમૂહ ઑબ્જેક્ટ ક્લાસ બનાવે છે. ક્લાસ એ ઑબ્જેક્ટ માટેની એક બ્લુપ્રિન્ટ છે, અને દરેક ઑબ્જેક્ટ તે ક્લાસનો ઈન્સ્ટન્સ છે. ઑબ્જેક્ટ અને ક્લાસ ઉપરાંત, ઑબ્જેક્ટ-ઑરિએન્ટેડ પ્રોગ્રામિંગ એન્કેપ્સ્યુલેશન, ડેટા હાઇડિંગ અને એબ્સ્ટ્રેક્શન જેવા ખ્યાલોને સમર્થન આપે છે. એન્કેપ્સ્યુલેશન ડેટા અને મેથડને એકસાથે મૂકે છે અને આકસ્મિક ફેરફારોથી ડેટાનું રક્ષણ કરે છે. કમ્પોઝિશન અને એગ્રિગેશનનો ઉપયોગ અનુક્રમે પાર્ટ-ઓફ (part-of) અને હેઝ-એ (has-a) સંબંધોને દર્શાવવા માટે થાય છે. ઈન્હેરીટન્સ અને પૉલિમૉર્ફિઝમ ઑબ્જેક્ટ-ઑરિએન્ટેડ પ્રોગ્રામિંગમાં મહત્વની ભૂમિકા ભજવે છે. વારસાગત સંબંધો એક ક્લાસને બીજા ક્લાસની વિશેષતાઓનો ઉપયોગ કરવાની તેમજ ક્લાસને સબક્લાસમાં વિભાજીત કરવાની મંજૂરી આપે છે. આપણે પાંચ જુદા જુદા પ્રકારના વારસાગત સંબંધો જોયા : સિંગલ, મલ્ટીપલ, મલ્ટીલેવલ, હાઇરાર્કીકલ અને હાઇબ્રિડ. પૉલિમૉર્ફિઝમ એક જ મેથડના બહુવિધ સ્વરૂપોને મંજૂરી આપે છે, જે ડાઈનેમિઝમ અને ફ્લેક્સીબિલિટીને સમર્થન આપે છે. આ પ્રકરણની સમાપ્તિ લોકપ્રિય અને વ્યાપકપણે ઉપયોગમાં લેવાતી ઑબ્જેક્ટ-ઑરિએન્ટેડ ભાષાઓ C++, Java અને Python ની ચર્ચા સાથે કરવામાં આવી.

સ્વાધ્યાય

1. પ્રોસિજરલ પ્રોગ્રામિંગની મર્યાદાઓ શું છે?
2. ઑબ્જેક્ટ-ઑરિએન્ટેડ પ્રોગ્રામિંગનો બિલ્ડિંગ બ્લોક શું છે? ઑબ્જેક્ટ-ઑરિએન્ટેડ પ્રોગ્રામનું માળખું જણાવો.
3. ઑબ્જેક્ટ શું છે? તમે જાણતા હો તેવા ઓછામાં ઓછા 5 ઑબ્જેક્ટની સૂચિ બનાવો.
4. ઑબ્જેક્ટ શેનાથી બને છે? એક ઉદાહરણ આપો.
5. તમે ક્લાસ વિશે શું સમજો છો? તે ઑબ્જેક્ટથી કેવી રીતે અલગ છે?
6. નીચેના ઑબ્જેક્ટની પ્રોપર્ટી અને ઓપરેશન જણાવો : વિદ્યાર્થી, બસની ટિકિટ, વર્તુળ, સાયકલ.
7. ઈન્હેરીટન્સ શું છે? વારસાગત સંબંધોના વિવિધ પ્રકારોની સૂચિ બનાવો.
8. પૉલિમૉર્ફિઝમ શું છે? તેનો ઉપયોગ શું છે?
9. ઑબ્જેક્ટ-ઑરિએન્ટેડ પ્રોગ્રામિંગમાં એન્કેપ્સ્યુલેશન શા માટે મહત્વનું છે?
10. એન્કેપ્સ્યુલેશન અને એબ્સ્ટ્રેક્શનનો અર્થ સમજાવો.
11. **સાચું કે ખોટું જણાવો.**
    (1) ઑબ્જેક્ટ એ ક્લાસનો એક ઈન્સ્ટન્સ છે.
    (2) એબ્સ્ટ્રેક્શન ડેટાને છુપાવીને સુરક્ષિત બનાવે છે.
    (3) એન્કેપ્સ્યુલેશન પ્રોપર્ટીને છુપાવે છે અને ઓપરેશનને દર્શાવે છે.
    (4) ઈન્હેરીટન્સ એક ક્લાસની પ્રોપર્ટી અને મેથડનો બીજા ક્લાસમાં ઉપયોગ કરવાની મંજૂરી આપે છે.
    (5) પક્ષી એક સંકલ્પનાત્મક ક્લાસ (conceptual class) છે.
12. **ખાલી જગ્યા પૂરો.**
    (1) ____________ એક ક્લાસની લાક્ષણિકતાઓનો અન્ય ક્લાસમાં ઉપયોગ કરવાની મંજૂરી આપે છે.
    (2) ડેટા અને મેથડને એકજૂથ કરવાની પ્રક્રિયાને ____________ કહે છે.
    (3) ઑબ્જેક્ટ બનાવવા માટેની બ્લ્યુ પ્રિન્ટને ____________ કહે છે.
    (4) બિનજરૂરી વિગતોને છુપાવવાની પ્રક્રિયાને ____________ કહે છે.
    (5) ____________ દ્વારા એકથી વધુ ફંક્શનને એક સમાન નામ આપી શકાય છે.
13. **બહુ વિકલ્પી પ્રશ્નો. સૌથી યોગ્ય જવાબ પસંદ કરો.**
    (1) OOP (ઑબ્જેક્ટ-ઑરિએન્ટેડ પ્રોગ્રામિંગ) માં ઑબ્જેક્ટ શું છે?
        (a) ક્લાસ બનાવવા માટેની બ્લુપ્રિન્ટ (b) ડેટા ટાઈપ
        (c) વાસ્તવિક દુનિયાની એન્ટિટી (d) ટાસ્ક
    (2) OOP માં ક્લાસ શું છે?
        (a) ઑબ્જેક્ટ્સનો સમૂહ (b) ઑબ્જેક્ટ્સ બનાવવા માટેનું એક ટેમ્પલેટ
        (c) ખાસ પ્રકારનો ઑબ્જેક્ટ (d) મેથડનો સમૂહ
    (3) નીચેનામાંથી શેના દ્વારા બાહ્ય હસ્તક્ષેપ સામે ડેટા અને મેથડને સુરક્ષિત બનાવવામાં આવે છે?
        (a) Inheritance (b) Polymorphism (c) Encapsulation (d) Abstraction

(4) નીચેનામાંથી કયું બિનજરૂરી વિગતો છુપાવે છે અને માત્ર જરૂરી વિગતો જ પ્રસ્તુત કરે છે?

(a) Inheritance (b) Polymorphism (c) Encapsulation (d) Abstraction

(5) નીચેનામાંથી શેના દ્વારા એક ક્લાસની વિશેષતાઓનો ઉપયોગ બીજા ક્લાસમાં કરી શકાય છે?

(a) Inheritance (b) Polymorphism (c) Encapsulation (d) Abstraction

(6) ઑબ્જેક્ટ વિશે નીચેનામાંથી કયું વિધાન સાચું નથી?

(a) ઑબ્જેક્ટમાં એટ્રિબ્યુટ્સ અને મેથડનો સમાવેશ થાય છે

(b) ઑબ્જેક્ટ ક્લાસનું એક ઇન્સ્ટન્સ છે

(c) ઑબ્જેક્ટ એક વાસ્તવિક દુનિયાની એન્ટિટી છે

(d) ઑબ્જેક્ટ ક્લાસની બ્લુપ્રિન્ટ છે

(7) પૉલિમૉર્ફિઝમનો અર્થ શું છે?

(a) એક જ ફંક્શનના વિવિધ સ્વરૂપો (b) એક જ ક્લાસના વિવિધ ઇન્સ્ટન્સ

(c) બેઝ ક્લાસ માટેનો સબક્લાસ (d) ઑબ્જેક્ટનું એગ્રિગેશન

(8) નીચેનામાંથી કયો ઑબ્જેક્ટ નથી?

(a) લંબચોરસ (Rectangle) (b) ત્રિજ્યા (Radius)

(c) વર્તુળ (Circle) (d) ત્રિકોણ (Triangle)

(9) ઇનહેરીટન્સના કયા પ્રકારમાં બે કે તેથી વધુ ક્લાસનો બેઝ ક્લાસ તરીકે ઉપયોગ થાય છે?

(a) સિંગલ (b) મલ્ટીપલ (c) મલ્ટીલેવલ (d) હાઈરાર્કીકલ

(10) નીચેનામાંથી કઈ ઑબ્જેક્ટ-ઑરિએન્ટેડ ભાષા નથી?

(a) C (b) C++ (c) Java (d) Python

પ્રાયોગિક સ્વાધ્યાય

1. તમારી શાળાની લાઇબ્રેરીની મુલાકાત લો અને ત્યાં જોવા મળતા ઓછામાં ઓછા 5 ઑબ્જેક્ટની યાદી બનાવો.

2. લાઇબ્રેરીમાં પુસ્તક (Book) સૌથી મહત્વપૂર્ણ ઑબ્જેક્ટમાંથી એક છે. તેના એટ્રીબ્યુટ અને બિહેવીયરની યાદી બનાવો.

3. નીચેના ઑબ્જેક્ટને તેમની સામાન્ય વિશેષતાઓના આધારે વિવિધ ક્લાસમાં વર્ગીકૃત કરો:

કાર (Car), કેળું (Banana), મોર (Peacock), સફરજન (Apple), સ્કૂટર (Scooter), ચકલી (Sparrow), દ્રાક્ષ (Grape), ટ્રક (Truck), કબૂતર (Dove), બસ (Bus)

4. શોપિંગ મૉલના વિવિધ વિભાગોમાં વેચાતી વસ્તુઓ માટે એક ઇનહેરીટન્સ હાઈરાર્કી તૈયાર કરો.

ઉદાહરણ તરીકે, કાપડ (cloth) એક વસ્તુ છે જે આગળ પુરુષો, સ્ત્રીઓ અને બાળકોના કપડાંમાં વિભાજિત થાય છે. પુરુષોના કપડાં આગળ શર્ટ અને પેન્ટ વગેરેમાં વિભાજિત થઈ શકે છે.

5. વિવિધ પ્રકારના વાહનો (vehicles) માટે હાઈરાર્કી તૈયાર કરો.

ટોચના સ્તરે એવા સામાન્ય કાર્યોને ઓળખો, જેને નીચલા સ્તરે અલગ અલગ અમલની જરૂર હોય.