પારુલ ઇન્સ્ટિટ્યુટ ઑફ એન્જિનિયરિંગ અને
ટેક્નોલોજી

ડિપ્લોમા સ્ટડીઝ કોમ્પ્યુટર

વિભાગ

1st સત્ર

ઉકેલ સાથે પ્રશ્ન બેંક

કોમ્પ્યુટર પ્રોગ્રામીંગ

(03606103)

પ્રશ્નો

UNIT-1- ફ્લો ચાર્ટ અને અલ્ગોરિધમ 1. ફ્લોચાર્ટની વ્યાખ્યા.
(2 M)
2. અલ્ગોરિધમની વ્યાખ્યા? બે નંબર ઉમેરવા માટે અલ્ગોરિધમ લખો.(3 m)

4. ફ્લોચાર્ટના ફાયદા અને ગેરફાયદા સમજાવો.(2 m)
5. ફ્લોચાર્ટના પ્રતીકો વ્યાખ્યાયિત કરો(2 m)
6. વર્તુળનું ક્ષેત્રફળ શોધવા માટે અલ્ગોરિધમ લખો.(4 M)
7. આપેલ સંખ્યાના ફેક્ટોરિયલ શોધવા માટે ફ્લોચાર્ટ દોરો. (4 M)
8. ફિબોનાકી શ્રેણી છાપવા માટે ફ્લોચાર્ટ દોરો. (4 M)
9. ત્રણમાંથી મહત્તમ સંખ્યાઓ શોધવા માટે ફ્લોચાર્ટ દોરો. (4 M)

10. આપેલ સંખ્યા એકી કે બેકી છે તે દર્શાવવા માટે ફ્લોચાર્ટ દોરો. (4 M)

11. આપેલ સંખ્યાનો વર્ગ શોધવા માટે અલ્ગોરિધમ લખો. (4 M)
12. લંબચોરસનું ક્ષેત્રફળ શોધવા માટે અલ્ગોરિધમ લખો.(4 M)

UNIT-2- 'C' ની મૂળભૂત બાબતો
1. C પ્રોગ્રામની મૂળભૂત રચના સમજાવો. (3 M)
2. C ભાષાના ફાયદા લખો. (2 M)
3. C ભાષાના ટોકન્સ સમજાવો. (2 M)
4. વેરીએબલની ઘોષણા અને પ્રારંભ શું છે? ઉદાહરણ આપો.(4 M)

5. ચલ વ્યાખ્યાયિત કરવા માટેના નિયમો લખો. (2 M)
6.તમારી સ્ક્રીન પર "હેલો વર્લ્ડ" પ્રિન્ટ કરવા માટે C પ્રોગ્રામ લખો.(3 M)
7. ઉદાહરણ સાથે પ્રકાર રૂપાંતરણ સમજાવો. (4 M)
8. Constant નો ઉપયોગ કરીને વર્તુળનું ક્ષેત્રફળ શોધવા માટે C પ્રોગ્રામ લખો. (4 M)
9.C. (2 M) ના કીવર્ડ્સ સમજાવો
10. આપેલ નંબર સકારાત્મક કે નકારાત્મક છે તે તપાસવા માટે પ્રોગ્રામ પર લખો. (4M)

11. આપેલ સંખ્યા વિષમ કે બેકી છે તે શોધવા માટે પ્રોગ્રામ લખો. (4 M)
12. ઓળખકર્તા શું છે? (2 M)
13.અચલના પ્રકારો સમજાવો.(2 M)

14. ચલોની ગતિશીલ શરૂઆતના ઉદાહરણ સાથે સમજાવો. (4 M)

15. C. (2 M) માં સંશોધકોને સમજાવો

યુનિટ-3 ઓપરેટર અને એક્સપ્રેશન્સ 1. ઓપરેટર શું છે? સીમાં
ઉપલબ્ધ વિવિધ ઓપરેટરોની યાદી બનાવો. (2 M)
2. અંકગણિત ઓપરેટરોને ઉદાહરણો સાથે સમજાવો. (4 M)
3. રિલેશનલ ઓપરેટરોને ઉદાહરણો સાથે સમજાવો. (4 M)
4. લોજિકલ ઓપરેટરોને ઉદાહરણ સાથે સમજાવો. (4 M)
5. અસાઇનમેન્ટ ઓપરેટરોને ઉદાહરણ સાથે સમજાવો. (4 M)
6. શરતી ઓપરેટરોને ઉદાહરણ સાથે સમજાવો. (4 M)
7. ઉદાહરણ સાથે બીટવાઇઝ ઓપરેટરો સમજાવો. (4 M)
8. અંકગણિત ઓપરેટરનો ઉપયોગ કરીને કેલ્ક્યુલેટર ડિઝાઇન કરવા માટે એક પ્રોગ્રામ
લખો. (4 M)
9. Bitwise Operator નો ઉપયોગ કરીને બે નંબરો સ્વેપ કરવા માટે એક પ્રોગ્રામ લખો. (4 M)

યુનિટ-4- નિર્ણય નિવેદન

1. સરળ જો નિવેદનો સમજાવો. (2 M)
2. જો....અન્ય નિવેદનો સમજાવો. (3 M)
3. ઉદાહરણ સાથે Nested if-else સ્ટેટમેન્ટ સમજાવો. (4 M)
4. ઉદાહરણ સાથે જો અન્ય-જો નિસરણી સમજાવો. (4 M)
5. ઉદાહરણ સાથે સ્વિચ ...કેસ સ્ટેટમેન્ટ સમજાવો. (4 M)
6. ઉદાહરણ સાથે બ્રેક સ્ટેટમેન્ટ સમજાવો. (4 M)
7. If...Else અને સ્વિચ કેસ વચ્ચે શું તફાવત છે. (4M)
8. બ્રેક, ચાલુ રાખો વચ્ચેનો તફાવત આપો. (4 M)
9. આપેલ કોઈ વિષમ અથવા બેકી છે તે શોધવા માટે પ્રોગ્રામ લખો. (4 M)
10. સ્વિચ કેસનો ઉપયોગ કરીને દિવસો દર્શાવવા માટે એક પ્રોગ્રામ લખો. (4 M)
11. આપેલ નંબર ધન છે કે નકારાત્મક તે શોધવા માટે પ્રોગ્રામ લખો. (4M)

એકમ- 5 લૂપ કંટ્રોલ સ્ટેટમેન્ટ

1. લૂપ નિયંત્રણ નિવેદનો વ્યાખ્યાયિત કરો અને તેના પ્રકારોની સૂચિ બનાવો. (3 M)
2. પ્રથમ 10 નંગનો સરવાળો પ્રિન્ટ કરવા માટે એસી પ્રોગ્રામ લખો. લૂપ માટે ઉપયોગ કરીને. (4 M)
3. લૂપ માટે નેસ્ટેડનો ઉપયોગ કરવાના રાજ્ય ફાયદા. (3 M)
4. જ્યારે લૂપ કરો ત્યારે તફાવત કરો. (3 M)
5. WHILE LOOP ઉદાહરણ સાથે સમજાવો. (4 M)
6. ઉદાહરણ સાથે DO...WHILE LOOP સમજાવો. (4 M)
7. 2 થી 100 ની વચ્ચે અવિભાજ્ય સંખ્યાઓ શોધવા માટે નેસ્ટેડ ફોર લૂપ્સનો ઉપયોગ કરીને AC પ્રોગ્રામ લખો. (4 M)

જવાબ

એકમ-1- ફ્લો ચાર્ટ અને અલ્ગોરિધમ

1. ફ્લોચાર્ટની વ્યાખ્યા. (2 મીટર)
વર્ષ:

. ફ્લોચાર્ટ એ અલ્ગોરિધમનું ડાયાગ્રામમેટિક પ્રતિનિધિત્વ છે. ફ્લોચાર્ટ પ્રોગ્રામ લખવામાં અને અન્ય લોકોને પ્રોગ્રામ સમજાવવામાં ખૂબ મદદરૂપ છે.

. લંબચોરસ, હીરા, અંડાકાર અને નાના વર્તુળો જેવા ચોક્કસ હેતુના પ્રતીકોનો ઉપયોગ કરીને ફ્લો ચાર્ટ દોરવામાં આવે છે. આ પ્રતીકો ફ્લો લાઇન તરીકે ઓળખાતા તીરો દ્વારા જોડાયેલા છે.

2. અલ્ગોરિધમની વ્યાખ્યા? બે નંબર ઉમેરવા માટે અલ્ગોરિધમ લખો. (3 મીટર)
વર્ષ:

. એલ્ગોરિધમ એ તાર્કિક અને ગાણિતિક સમસ્યાઓ ઉકેલવા માટેની સૂચનાઓનો સમૂહ છે. આપેલ સમસ્યાને ઉકેલવા માટે તે પગલું-દર-પગલાંનો અભિગમ તૈયાર કરે છે. તે ઇનપુટ્સ લે છે અને આઉટપુટ ઉત્પન્ન કરે છે.

. સારા અલ્ગોરિધમના લક્ષણો છે: ○ ચોકસાઇ – પગલાં ચોક્કસ રીતે
જણાવવામાં આવ્યા છે (વ્યાખ્યાયિત).

○ વિશિષ્ટતા - દરેક પગલાના પરિણામો અનન્ય રીતે વ્યાખ્યાયિત કરવામાં આવે છે અને ફક્ત તેના પર આધાર રાખે છે ઇનપુટ અને અગાઉના પગલાંનું પરિણામ.

○ મર્યાદિતતા - મર્યાદિત સંખ્યામાં સૂચનાઓ અમલમાં મૂક્યા પછી અલ્ગોરિધમ અટકે છે.

○ ઇનપુટ - અલ્ગોરિધમ ઇનપુટ મેળવે છે.

○ આઉટપુટ - અલ્ગોરિધમ આઉટપુટ ઉત્પન્ન કરે છે.

○ સામાન્યતા - અલ્ગોરિધમ ઇનપુટ્સના સમૂહને લાગુ પડે છે.

 બે સંખ્યા ઉમેરવા માટે અલ્ગોરિધમ લખો.
પગલું 1: શરૂ કરો
પગલું 2: $_{a,b}$ માં બે નંબરો વાંચો
પગલું 3: $_{c=a+b}$
પગલું 4: લખો/પ્રિન્ટ કરો
$_{c}$ પગલું 5: રોકો.

3. ફ્લોચાર્ટના ફાયદા અને ગેરફાયદા સમજાવો.(2 $_{m}$)
વર્ષ:

 ફ્લોચાર્ટના ફાયદા

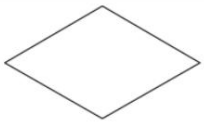
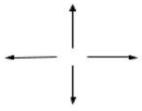
. તે સંચારની અનુકૂળ પદ્ધતિ છે.

. તે ખૂબ જ સ્પષ્ટ રીતે સૂચવે છે કે શું કરવામાં આવી રહ્યું છે, જ્યાં પ્રોગ્રામમાં તાર્કિક છે જટિલતાઓ

. પ્રોગ્રામિંગને ઠીક કરવાની ચાવી.
. નવી સિસ્ટમની યોજના બનાવવા અને ડિઝાઇન કરવા માટે તે એક મહત્વપૂર્ણ સાધન છે.
. તે દરેક સ્તરે ભજવેલ ભૂમિકાને સ્પષ્ટપણે દર્શાવે છે.
. તે ભવિષ્યમાં અસુવિધાઓને બચાવે છે અને દસ્તાવેજીકરણના હેતુને પૂર્ણ કરે છે
એક સિસ્ટમ.
. તે તાર્કિક ચોકસાઈને પ્રોત્સાહન આપે છે.
. તે સુનિશ્ચિત કરે છે કે કોઈપણ તાર્કિક માર્ગ કોઈપણ પગલાં લીધા વિના અધૂરો ન રહે.

 ફ્લોચાર્ટના ગેરફાયદા
. ફ્લોચાર્ટ એ સમયનો વ્યય છે અને સોફ્ટવેરની પ્રક્રિયાને ધીમી કરે છે
વિકાસ
. ફ્લોચાર્ટ બનાવવા માટે ખૂબ ખર્ચાળ છે અને તેનો ઉપયોગ અને સંચાલન કરવું મુશ્કેલ છે.
. ફ્લોચાર્ટ એ માણસથી કમ્પ્યુટર કોમ્યુનિકેશન માટે નથી.
. જો તમારે પ્રક્રિયામાં ફેરફાર અથવા વૈકલ્પિક કરવાની જરૂર હોય તો તે ફ્લોચાર્ટમાં કરવું ખૂબ જ મુશ્કેલ હશે. કારણ કે
કાં તો તમારે ફ્લોચાર્ટનો અંત ભૂંસી નાખવો પડશે અથવા તો શરૂ કરવું પડશે.'

4. ફ્લોચાર્ટના પ્રતીકોને વ્યાખ્યાયિત કરો. (2 મીટર)
વર્ષ:

Symbol	Description
	Start / Stop
	Input / Output (Read / Print)
	Process
	Decision Making
	Subroutine
	Direction

5. વર્તુળનું ક્ષેત્રફળ શોધવા માટે અલ્ગોરિધમ લખો. (4 M)

જવાબ:

અલ્ગોરિધમના પગલાં: 1. START.

2. પૂર્ણાંક વિસ્તાર, ત્રિજ્યા.

3. પ્રિન્ટ કરો "વર્તુળની ત્રિજ્યા દાખલ કરો - 4. વિસ્તાર=3.14*ત્રિજ્યા*ત્રિજ્યા.

"

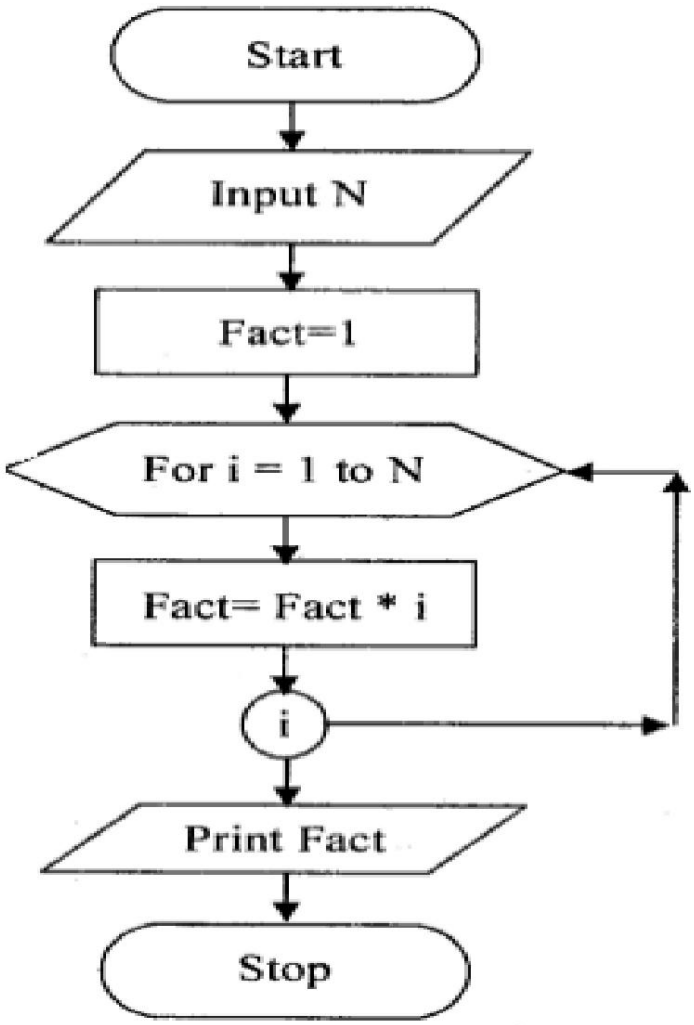
"

5. "વર્તુળનો વિસ્તાર = છાપો

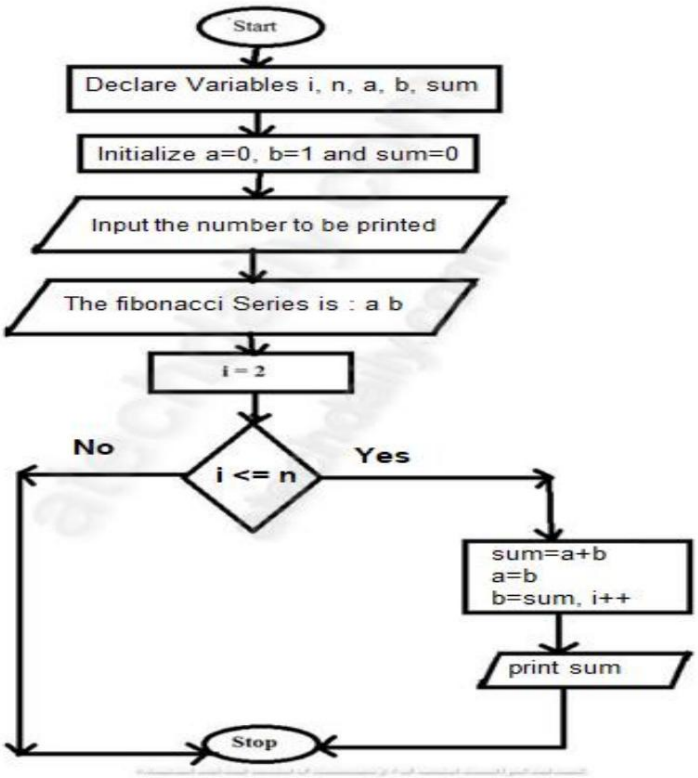
6. પ્રિન્ટ એરિયા.

7. બહાર નીકળો.

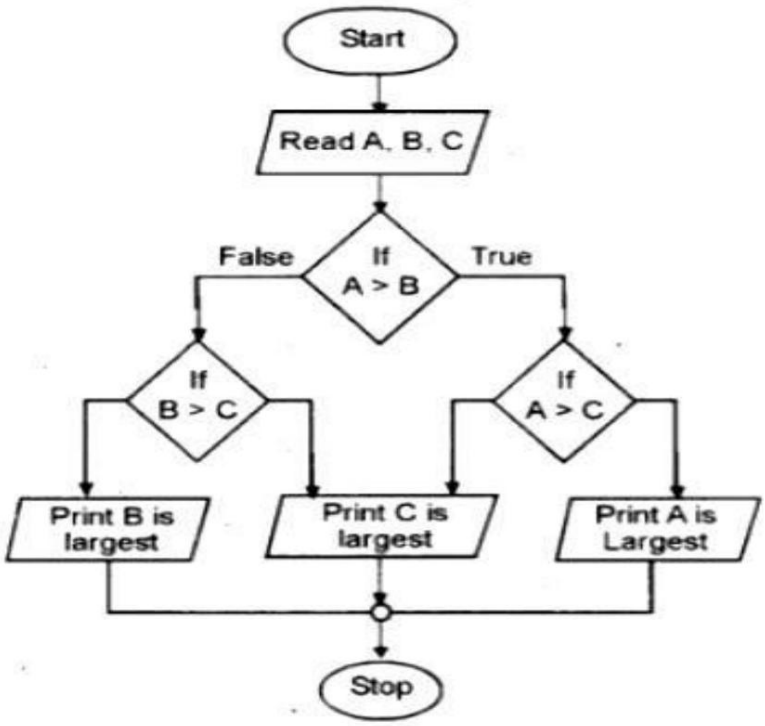
6. આપેલ સંખ્યાના ફેક્ટોરિયલ શોધવા માટે ફ્લોચાર્ટ દોરો.(4 M)



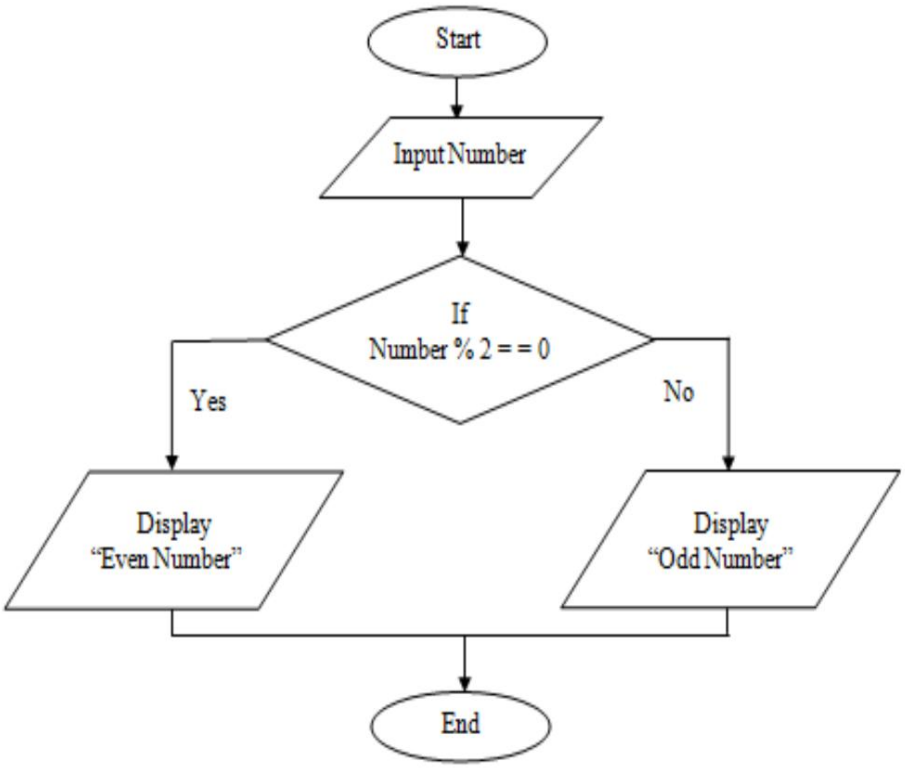
8. ફિબોનાકી શ્રેણી છાપવા માટે ફ્લોચાર્ટ દોરો. (4 M)
વર્ષ:



8. ત્રણમાંથી મહત્તમ સંખ્યાઓ શોધવા માટે ફ્લોચાર્ટ દોરો. (4 M)
વર્ષ:



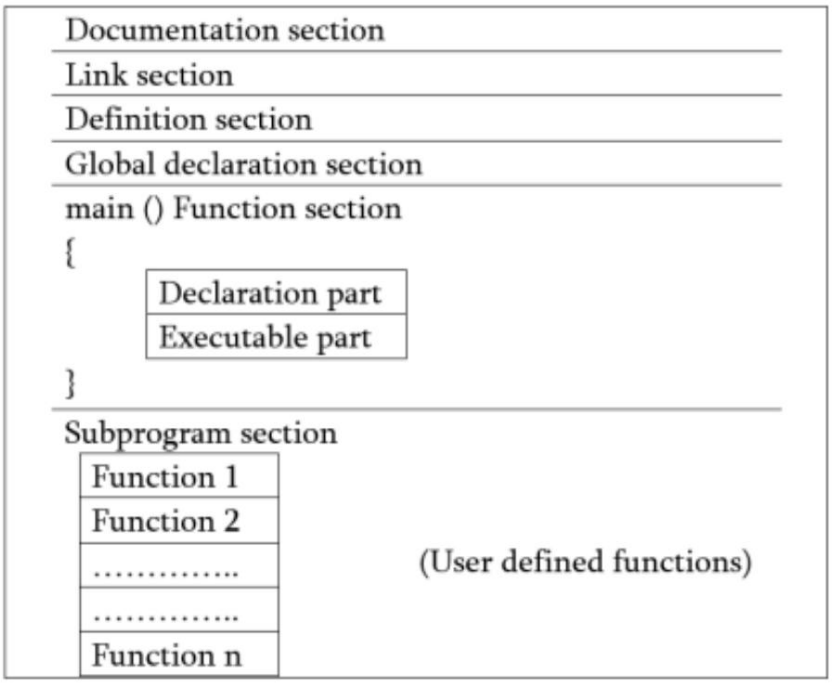
9. આપેલ સંખ્યા એકી કે બેકી છે તે દર્શાવવા માટે ફ્લોચાર્ટ દોરો. (4 M)
વર્ષ:



10. આપેલ સંખ્યાનો વર્ગ શોધવા માટે અલ્ગોરિધમ લખો. (4 M)
જવાબ: અલ્ગોરિધમ આ રીતે લખી શકાય છે: પગલું
1 - પ્રક્રિયા શરૂ કરો પગલું 2 - ઇનપુટ મેળવો x પગલું
3 - ઇનપુટ મૂલ્યનો ગુણાકાર કરીને વર્ગની ગણતરી
કરો એટલે કે., ચોરસ □ x* x પગલું 4 - પરિણામ ચોરસ દર્શાવો પગલું 5 - બંધ

11. લંબચોરસનું ક્ષેત્રફળ શોધવા માટે અલ્ગોરિધમ લખો. (4 M)
જવાબ: અલ્ગોરિધમ આ રીતે લખી શકાય છે:
પગલું 1: પ્રારંભ કરો
પગલું 2: ઇનપુટ લંબાઈ અને પહોળાઈ
પગલું 3: વિસ્તાર = લંબાઈ * પહોળાઈ
પગલું 4: પ્રિન્ટ વિસ્તાર
પગલું 5: રોકો.

1. C પ્રોગ્રામની મૂળભૂત રચના સમજાવો. (3 M)
જવાબ: સી પ્રોગ્રામની મૂળભૂત રચના:



1. દસ્તાવેજીકરણ વિભાગ: દસ્તાવેજીકરણ વિભાગમાં પ્રોગ્રામનું નામ, લેખક અને અન્ય વિગતો આપતી ટિપ્પણી રેખાઓનો સમૂહ હોય છે, જેનો પ્રોગ્રામર પછીથી ઉપયોગ કરવા માંગે છે.

2. લિંક વિભાગ: લિંક વિભાગ કમ્પાઇલરને સિસ્ટમ લાઇબ્રેરીમાંથી ફંક્શનને લિંક કરવા માટે સૂચનાઓ પ્રદાન કરે છે જેમ કે #include ડાયરેક્ટિવનો ઉપયોગ.
3. વ્યાખ્યા વિભાગ: વ્યાખ્યા વિભાગ તમામ સાંકેતિક સ્થિરાંકોને વ્યાખ્યાયિત કરે છે જેમ કે #define નિર્દેશનો ઉપયોગ કરીને.
4. વૈશ્વિક ઘોષણા વિભાગ: કેટલાક ચલો છે જેનો ઉપયોગ એક કરતા વધુ કાર્યમાં થાય છે. આવા ચલોને વૈશ્વિક ચલ કહેવામાં આવે છે અને વૈશ્વિક ઘોષણા વિભાગમાં જાહેર કરવામાં આવે છે જે તમામ કાર્યોની બહાર છે. આ વિભાગ વપરાશકર્તા દ્વારા નિર્ધારિત તમામ કાર્યોને પણ જાહેર કરે છે.

5. મુખ્ય કાર્ય વિભાગ: તે કમ્પાઈલરને કહે છે કે મુખ્ય() થી એક્ઝેક્યુશન ક્યાંથી શરૂ કરવું.

{ એક્ઝેક્યુશન શરૂ થવાથી બિંદુ }
 મુખ્ય કાર્યમાં બે વિભાગ છે

1. ઘોષણા વિભાગ : આમાં ચલો અને તેમના ડેટા પ્રકારો જાહેર કરવામાં આવે છે.
2. એક્ઝિક્યુટેબલ વિભાગ : આમાં પ્રોગ્રામનો ભાગ છે જે ખરેખર આપણને જોઈતું કાર્ય કરે છે.

2. C ભાષાના ફાયદા લખો. (2 M)

જવાબ: c ભાષાના ફાયદાઓની સૂચિ બનાવો: 1. તે સમજવામાં

સરળ છે 2. ઘણી લાઇબ્રેરીઓની હાજરી 3. લખવામાં સરળ

છે

4. ઓછી કિંમત

5. ઝડપી અમલ ઝડપ

6. પોર્ટેબલ

7. સરળ ડીબગીંગ 8.

પ્રક્રિયાલક્ષી ભાષા 9. સંકલનની ઝડપ 10.

અલ્ગોરિધમ્સ અને ડેટા સ્ટ્રક્ચર્સનું અમલીકરણ

11. ડાયનેમિક મેમરી ફાળવણી.

3. c ભાષાના ટોકન્સ સમજાવો. (2 m)

જવાબ: c ભાષામાં ટોકન્સને નીચેની શ્રેણીઓમાં વિભાજિત કરી શકાય છે:

. c માં કીવર્ડ્સ . c માં
ઓળખકર્તાઓ . c માં
સ્ટ્રીંગ્સ . c માં ઓપરેટર્સ
. c માં સતત . c માં વિશિષ્ટ
અક્ષરો

c માં કીવર્ડ્સ: c માં

કીવર્ડ્સને પૂર્વ-વ્યાખ્યાયિત અથવા આરક્ષિત શબ્દો તરીકે વ્યાખ્યાયિત કરી શકાય છે જેનું પોતાનું મહત્વ છે, અને દરેક કીવર્ડ તેની પોતાની કાર્યક્ષમતા ધરાવે છે.

c માં ઓળખકર્તા

c માં આઇડેન્ટિફાયર: c

માં ઓળખકર્તાઓનો ઉપયોગ ચલ, ફંક્શન્સ, એરે, સ્ટ્રક્ચર્સ વગેરેના નામકરણ માટે થાય છે.

c માં ઓળખકર્તા એ વપરાશકર્તા દ્વારા નિર્ધારિત શબ્દો છે.

c માં સ્ટ્રીંગ્સ: c

માં સ્ટ્રીંગ્સ હંમેશા સ્ટ્રીંગના અંતમાં નલ અક્ષર '\0' ધરાવતા અક્ષરોની એરે તરીકે રજૂ થાય છે.

c માં ઑપરેટર્સ: c માં

ઑપરેટર્સ એ કાર્યો કરવા માટે વપરાતું વિશિષ્ટ પ્રતીક છે. ડેટા વસ્તુઓ કે જેના પર ઓપરેટરો લાગુ કરવામાં આવે છે તે ઓપરેન્ડ તરીકે ઓળખાય છે. ઓપરેટરો ઓપરેન્ડ વચ્ચે લાગુ કરવામાં આવે છે.

યુનરી ઓપરેટર: યુનરી

ઓપરેટર એ એક ઓપરેન્ડ પર લાગુ ઓપરેટર છે. ઉદાહરણ તરીકે: ઇન્ક્રીમેન્ટ ઓપરેટર (++), ડીક્રીમેન્ટ ઓપરેટર (--), sizeof, (પ્રકાર)*.

બાઈનરી ઓપરેટર:

દ્વિસંગી ઓપરેટર એ બે ઓપરેન્ડ વચ્ચે લાગુ થયેલ ઓપરેટર છે.

c માં સ્થિરાંકો:

અચળ એ ચલને અસાઇન કરેલ મૂલ્ય છે જે સમગ્ર પ્રોગ્રામ દરમિયાન સમાન રહેશે, એટલે કે, સ્થિર મૂલ્ય બદલી શકાતું નથી.

c માં વિશિષ્ટ પાત્રો:

c માં કેટલાક વિશિષ્ટ અક્ષરોનો ઉપયોગ કરવામાં આવે છે, અને તેનો વિશેષ અર્થ છે જેનો ઉપયોગ અન્ય હેતુ માટે કરી શકાતો નથી.

4. વેરીએબલની ઘોષણા અને પ્રારંભ શું છે? ઉદાહરણ આપો.
(4 M)

વર્ષ:

. કોમ્પ્યુટર પ્રોગ્રામિંગ ભાષામાં વેરીએબલની ઘોષણા એ વેરીએબલનું નામ અને તેના ડેટા પ્રકારને સ્પષ્ટ કરવા માટે વપરાતું નિવેદન છે. ઘોષણા કમ્પાઈલરને પ્રોગ્રામમાં એન્ટિટીના અસ્તિત્વ અને તેના સ્થાન વિશે જણાવે છે. જ્યારે તમે ચલ જાહેર કરો છો, ત્યારે તમારે તેને પ્રારંભ પણ કરવો જોઈએ.

. આરંભ એ વેરીએબલને મૂલ્ય અસાઇન કરવાની પ્રક્રિયા છે. દરેક પ્રોગ્રામિંગ લેંગ્વેજ પાસે વેરીએબલ શરૂ કરવાની પોતાની પદ્ધતિ છે. જો મૂલ્ય વેરીએબલને સોંપવામાં આવ્યું નથી, તો પ્રક્રિયાને માત્ર ઘોષણા કહેવામાં આવે છે.

ચલ જાહેર કરવાનું મૂળભૂત સ્વરૂપ છે:

પ્રકાર ઓળખકર્તા [= મૂલ્ય] [, ઓળખકર્તા [= મૂલ્ય]]...];

અથવા

data_typevariable_name = મૂલ્ય;

5. ચલ વ્યાખ્યાયિત કરવા માટેના નિયમો લખો. (2 M)

જવાબ: ચલોને વ્યાખ્યાયિત કરવા માટેના નિયમો

. ચલમાં મૂળાક્ષરો, અંકો અને અન્ડરસ્કોર હોઈ શકે છે.

. ચલ નામ મૂળાક્ષરોથી શરૂ થઈ શકે છે, અને માત્ર અન્ડરસ્કોર. તે સાથે શરૂ કરી શકાતી નથી એક અંક.

. ચલ નામની અંદર કોઈ વ્હાઇટસ્પેસની મંજૂરી નથી.

. ચલ નામ કોઈપણ આરક્ષિત શબ્દ અથવા કીવર્ડ ન હોવું જોઈએ, દા.ત. int, goto , વગેરે

6.તમારી સ્ક્રીન પર "હેલો વર્લ્ડ" પ્રિન્ટ કરવા માટે c પ્રોગ્રામ લખો. (3 M)

જવાબ:

```
#include<stdio.h>
પૂર્ણાંક મુખ્ય()
{
printf("હેલો વર્લ્ડ");
પરત 0;
}
```

આઉટપુટ:

હેલોવર્લ્ડ

7. ઉદાહરણ સાથે પ્રકાર રૂપાંતરણ સમજાવો. (4 M)

જવાબ: ટાઇપ કાસ્ટ એ મૂળભૂત રીતે એક પ્રકારમાંથી બીજામાં રૂપાંતર છે.

પ્રકાર રૂપાંતરણના બે પ્રકાર છે:

1. ગર્ભિત પ્રકાર રૂપાંતરણ 2. સ્પષ્ટ પ્રકાર
રૂપાંતરણ

ગર્ભિત પ્રકાર રૂપાંતરણ: . તેને
'સ્વચાલિત પ્રકાર રૂપાંતર' તરીકે પણ ઓળખવામાં આવે છે. . વપરાશકર્તાના
કોઈપણ બાહ્ય ટ્રિગર વિના, કમ્પાઈલર દ્વારા તેની જાતે જ કરવામાં આવે છે. . સામાન્ય રીતે ત્યારે થાય છે જ્યારે
અભિવ્યક્તિમાં એક કરતા વધુ ડેટા પ્રકાર હાજર હોય. . આવી સ્થિતિમાં નુકસાન ટાળવા માટે પ્રકારનું રૂપાંતર (પ્રકાર
પ્રમોશન) થાય છે
ડેટા

bool -> char -> short int -> int -> unsigned int -> long

-> unsigned -> longlong -> float -> double -> long double

ગર્ભિત રૂપાંતરણો માટે માહિતી ગુમાવવી શક્ય છે, ચિહ્નો ખોવાઈ શકે છે.

પ્રકાર ગર્ભિત રૂપાંતરણનું ઉદાહરણ:

```
#include<stdio.h> int main()
{ int x = 10; // પૂર્ણાંક x
char y = 'a'; // અક્ષર c


   // y ગર્ભિત રીતે int માં રૂપાંતરિત. ASCII // 'a' નું મૂલ્ય
   97 છે
   x = x + y; //
   x અસ્પષ્ટપણે ફ્લોટ ફ્લોટ z = x + 1.0 માં
રૂપાંતરિત થાય છે; printf("x = %d, z = %f", x, z); પરત
0; }
```

આઉટપુટ:
x = 107, z = 108.000000

સ્પષ્ટ પ્રકાર રૂપાંતરણ:
. આ પ્રક્રિયાને ટાઇપ કાસ્ટિંગ પણ કહેવામાં આવે છે અને તે વપરાશકર્તા દ્વારા નિર્ધારિત છે. અહીં વપરાશકર્તા કરી શકે છે
તેને ચોક્કસ ડેટા પ્રકાર બનાવવા માટે પરિણામને કાસ્ટ કરો.

c માં વાક્યરચના:

(પ્રકાર) અભિવ્યક્તિ

ઉદાહરણ:

```
#include<stdio.h>
int main(){
ડબલ x = 1.2;
int sum = (int)x + 1;
printf("સમ = %d", સરવાળો); પરત
કરો 0;}
```

આઉટપુટ:
સરવાળો = 2

8. Constant નો ઉપયોગ કરીને વર્તુળનું ક્ષેત્રફળ શોધવા માટે c પ્રોગ્રામ લખો. (4 M)
જવાબ: #include<stdio.h>

```
પૂર્ણાંક મુખ્ય()
{
const float pi=3.14;
int ત્રિજ્યા;
ફ્લોટ વિસ્તાર;
printf("ત્રિજ્યા દાખલ કરો:");
scanf("%d",&ત્રિજ્યા);
વિસ્તાર = pi * (ત્રિજ્યા * ત્રિજ્યા);
printf("વર્તુળનો વિસ્તાર: %.2f", વિસ્તાર);
પરત 0;
}
```

9.c. (2 M) ના કીવર્ડ્સ સમજાવો

જવાબ: કીવર્ડ એ આરક્ષિત શબ્દ છે. તમે તેનો ઉપયોગ ચલ નામ, સતત નામ વગેરે તરીકે કરી શકતા નથી. c ભાષામાં ફક્ત 32 અનામત શબ્દો (કીવર્ડ્સ) છે.

c ભાષામાં 32 કીવર્ડ્સની સૂચિ નીચે આપેલ છે:

ઓટો	વિરામ	કેસ	ચાર	const	ડિફોલ્ટ કરવાનું ચાલુ રાખો		
બીજું બમણું		enum બાહ્ય ફ્લોટ			માટે	પર જાઓ	જો
int	લાંબી	રીટર્ન ટુંકમાં રજીસ્ટર કરો			હસ્તાક્ષર કર્યા	માપ	સ્થિર
રચના	સ્વિચ typedef	યુનિયન સહી વિનાનું રદબાતલ				અસ્થિર જ્યારે	

10. આપેલ નંબર સકારાત્મક કે નકારાત્મક છે તે તપાસવા માટે પ્રોગ્રામ પર લખો.
(4 M)
જવાબ: #include <stdio.h> void main()
{ intnum; printf("એક નંબર દાખલ
કરો: \n"); scanf("%d", &num); જો
(num> 0) printf("%d એ ધન સંખ્યા
છે \n", num); અન્યથા જો (સંખ્યા< 0)
printf("%d એ નકારાત્મક સંખ્યા છે \n",
num); else printf("0 ન તો સકારાત્મક કે
નકારાત્મક નથી"); }

આઉટપુટ:

સંખ્યા દાખલ કરો: 0
0 ન તો સકારાત્મક છે
કે ન તો નકારાત્મક

11. આપેલ સંખ્યા વિષમ અથવા બેકી છે તે શોધવા માટે પ્રોગ્રામ લખો (4 M)
જવાબ: #include <stdio.h> int main()
{ intnum; printf("પૂર્ણાંક દાખલ
કરો:"); scanf("%d", &num); if(num %
2 == 0) printf("%d બરાબર છે.", num);
else printf("%d વિચિત્ર છે.", num); પરત
0; }

આઉટપુટ:

પૂર્ણાંક દાખલ કરો: -7 -7
વિચિત્ર છે.

12. ઓળખકર્તા શું છે? (2 M)
જવાબ: C IDENTIFIERS Identifiers નો ઉપયોગ ચલ, કાર્યો અને એરેના નામ માટે સામાન્ય પરિભાષા તરીકે થાય છે.

સી ઓળખકર્તાઓને નામ આપતી વખતે અમુક નિયમોનું પાલન કરવું જોઈએ: . તેઓ અક્ષર અથવા અન્ડરસ્કોર
(_) થી શરૂ થવું જોઈએ. . તેમાં માત્ર અક્ષરો, અંકો અથવા અન્ડરસ્કોર હોવા જોઈએ.

. અન્ય કોઈ વિશિષ્ટ પાત્રની મંજૂરી નથી. . તે કીવર્ડ ન હોવો જોઈએ. .

તેમાં સફેદ જગ્યા ન હોવી જોઈએ. . તે 31 અક્ષરો સુધીનું હોવું જોઈએ

કારણ કે માત્ર પ્રથમ 31 અક્ષરો જ નોંધપાત્ર છે.

13.અચલના પ્રકારો સમજાવો. (2 M)

જવાબ: કોન્સ્ટન્ટ બે પ્રકારના હોય છે:

1. સંખ્યાત્મક સ્થિરાંકો:

2. અક્ષર સ્થિરાંકો:

. સંખ્યાત્મક સ્થિરાંકો:

1. પૂર્ણાંક સ્થિરાંક 786,-127 છે 2. લાંબા
સ્થિરાંક ટર્મિનલ 'l' અથવા 'L' સાથે લખવામાં આવે છે, ઉદાહરણ તરીકે 1234567899L એ a છે
લાંબા સતત.
3. સહી ન કરેલ સ્થિરાંકો ટર્મિનલ 'u' અથવા 'U' સાથે લખવામાં આવે છે, અને પ્રત્યય 'ul' અને
'UL' એ સહી વગરની લાંબી સૂચવે છે.

4. ફ્લોટિંગ પોઈન્ટ સ્થિરાંકોમાં દશાંશ બિંદુ અથવા ઘાતાંક અથવા બંને હોય છે.

. અક્ષર સ્થિરાંકો:

એક અક્ષર સ્થિરાંક એક અક્ષર તરીકે લખવામાં આવે છે જેમ કે 'a'.
અક્ષર સ્થિરતાનું મૂલ્ય એ મશીનના અક્ષર સમૂહમાંના અક્ષરનું સંખ્યાત્મક મૂલ્ય છે.

એસ્કેપ સિક્વન્સના કેટલાક ઉદાહરણો નીચે મુજબ છે: એસ્કેપ સિક્વન્સનું વર્ણન \a ચેતવણી

\b બેકસ્પેસ
\f ફોર્મ ફીડ \n નવી લાઈન \r કેરેજ રીટર્ન

\t આડું ટેબ
\v વર્ટિકલ ટેબ

શબ્દમાળા સ્થિરાંકો: બેવડા અવતરણથી ઘેરાયેલા શૂન્ય અથવા વધુ અક્ષરોનો ક્રમ છે.

ગણતરી સતત : તે સતત પૂર્ણાંક મૂલ્યોની સૂચિ છે.
ઉદા.: enum રંગ { લાલ, લીલો, વાદળી } enum માં પ્રથમ નામની કિંમત 0 છે અને પછીનું 1 અને તેથી વધુ જ્યાં સુધી સ્પષ્ટ મૂલ્યો ઉલ્લેખિત ન હોય. જો તમામ મૂલ્યો ઉલ્લેખિત ન હોય તો, અસ્પષ્ટ મૂલ્યો છેલ્લા ઉલ્લેખિત મૂલ્યથી પ્રગતિ ચાલુ રાખે છે.

14. ચલોની ગતિશીલ શરૂઆતના ઉદાહરણ સાથે સમજાવો. (4 M)

જવાબ: ઑબ્જેક્ટનું ડાયનેમિક ઇનિશિયલાઇઝેશન એ ઑબ્જેક્ટને રન ટાઇમ પર ઇનિશિયલાઇઝ કરવાનો સંદર્ભ આપે છે એટલે કે ઑબ્જેક્ટનું પ્રારંભિક મૂલ્ય રન ટાઇમ દરમિયાન પૂરું પાડવામાં આવે છે. કન્સ્ટ્રક્ટરનો ઉપયોગ કરીને અને કન્સ્ટ્રક્ટરને પેરામીટર મૂલ્યો પસાર કરીને ડાયનેમિક આરંભ પ્રાપ્ત કરી શકાય છે. રન ટાઈમ દરમિયાન ક્લાસ વેરીએબલ્સને ઈનિશિએલાઈઝ કરવા માટે આ પ્રકારનું ઈનિશિએલાઈઝેશન જરૂરી છે.

ઑબ્જેક્ટ્સના ગતિશીલ પ્રારંભની જરૂર છે:

1. તે મેમરીનો અસરકારક રીતે ઉપયોગ કરે છે.
2. ઓવરલોડેડ કન્સ્ટ્રક્ટરનો ઉપયોગ કરીને વિવિધ પ્રારંભિક ફોર્મેટ પ્રદાન કરી શકાય છે.
3. તેને ધ્યાનમાં રાખીને રન ટાઇમ પર ડેટાના વિવિધ ફોર્મેટનો ઉપયોગ કરવાની લવચીકતા છે
   પરિસ્થિતિ

15. c. (2 m) માં સંશોધકોને સમજાવો
જવાબ: તે વેરીએબલ માટે ફાળવવામાં આવનારી મેમરી સ્પેસની માત્રાનો ઉલ્લેખ કરે છે.
ચલ માટે ફાળવેલ મેમરીને સંશોધિત કરવા માટે મોડિફાયર્સ મૂળભૂત ડેટા પ્રકારો સાથે પ્રીફિક્સ કરવામાં આવે છે.

c પ્રોગ્રામિંગ લેંગ્વેજમાં પાંચ ડેટા પ્રકાર સંશોધકો છે:

- લાંબી
- ટૂંકું
- હસ્તાક્ષર કર્યા
- સહી ન કરેલ
- લાંબા લાંબા

કેરેક્ટર : કેરેક્ટર ડેટા ટાઈપનો ઉપયોગ કેરેક્ટર સ્ટોર કરવા માટે થાય છે. કેરેક્ટર ડેટા પ્રકારનું ચલ માત્ર એક બાઈટ મેમરી ફાળવે છે અને માત્ર એક જ અક્ષર સ્ટોર કરી શકે છે. કીવર્ડ char નો ઉપયોગ અક્ષર પ્રકારના ચલોને જાહેર કરવા માટે થાય છે. ઉદાહરણ તરીકે: char ch = 'A';

પૂર્ણાંક : પૂર્ણાંક ડેટા પ્રકારનો ઉપયોગ આંકડાકીય પ્રકારનું મૂલ્ય સંગ્રહવા માટે થાય છે. કીવર્ડ int નો ઉપયોગ પૂર્ણાંક પ્રકારના ચલોને જાહેર કરવા માટે થાય છે. પૂર્ણાંક ડેટા પ્રકારના વેરીએબલની મેમરીનું કદ ઓપરેટિંગ સિસ્ટમ પર આધારિત છે. ઉદાહરણ તરીકે: પૂર્ણાંક સંખ્યા = 10;

ફ્લોટ : ફ્લોટિંગ પોઈન્ટ ડેટા પ્રકારનો ઉપયોગ દશાંશ મૂલ્યોના મૂલ્યને સંગ્રહિત કરવા માટે થાય છે.
કીવર્ડ ફ્લોટનો ઉપયોગ ફ્લોટિંગ ડેટા પ્રકારના ચલોને જાહેર કરવા માટે થાય છે. ઉદાહરણ તરીકે: ફ્લોટ રેટ = 5.6;

ડબલ : ડબલ ડેટા પ્રકાર ફ્લોટિંગ ડેટા પ્રકાર જેવો જ છે સિવાય કે તે દસ અંક સુધીની ચોકસાઇ પ્રદાન કરે છે અને આઠ બાઇટ્સ મેમરી ધરાવે છે.
ઉદાહરણ તરીકે: ડબલ ડી = 11676.2435676542;

યુનિટ-3 ઓપરેટર અને અભિવ્યક્તિઓ

1. ઓપરેટર શું છે? સીમાં ઉપલબ્ધ વિવિધ ઓપરેટરોની યાદી બનાવો. (2 M)

જવાબ: ઓપરેટર એ એક પ્રતીક છે જે કમ્પાઈલરને ચોક્કસ ગાણિતિક અથવા તાર્કિક કાર્યો કરવા કહે છે.

ઓપરેટરોના પ્રકાર - . અંકગણિત
ઓપરેટર્સ . રિલેશનલ ઓપરેટર્સ . લોજિકલ
ઓપરેટર્સ . બિટવાઈસ ઓપરેટર્સ .
એસાઈનમેન્ટ ઓપરેટર્સ . મિસ ઓપરેટર્સ

2. અંકગણિત ઓપરેટરોને ઉદાહરણો સાથે સમજાવો. (4 M)

જવાબ: અંકગણિત ઓપરેટર ગાણિતિક કામગીરી કરે છે જેમ કે સરવાળો, બાદબાકી, ગુણાકાર, ભાગાકાર, ઇન્ક્રીમેન્ટ ઓપરેટર, ડીક્રમેન્ટ ઓપરેટર.

ઉદાહરણ તરીકે: #include
<stdio.h> main() { int a = 21; int b
= 10; int c ; c = a + b; printf("લાઇન
1 - c નું મૂલ્ય %d\n", c); c = a -
b; printf("લાઇન 2 - c નું મૂલ્ય
%d\n", c); * બી; printf("લાઇન 3
- c નું મૂલ્ય %d\n", c); c = a /
b; printf("લાઈન 4 - c નું મૂલ્ય %d\n", c); c = a % b; printf("લાઇન 5 - c
નું મૂલ્ય %d\n", c); c = a++; printf("લાઇન 6 - c નું મૂલ્ય %d\n", c);

c = a

c = a--;
printf("લાઇન 7 - c નું મૂલ્ય %d\n", c); }

આઉટપુટ:
લાઇન 1 - c નું મૂલ્ય 31 છે
લાઇન 2 - c નું મૂલ્ય 11 છે
લાઇન 3 - c નું મૂલ્ય 210 છે
પંક્તિ 4 - c નું મૂલ્ય 2 છે
પંક્તિ 5 - c નું મૂલ્ય 1 છે
લાઇન 6 - c નું મૂલ્ય 21 છે
લાઇન 7 - c નું મૂલ્ય 22 છે

3. રિલેશનલ ઓપરેટરોને ઉદાહરણો સાથે સમજાવો. (4 M)

જવાબ: સી ભાષા દ્વારા આધારભૂત રીલેશનલ ઓપરેટરો. ધારો કે ચલ A 10 ધરાવે છે અને ચલ B 20 પછી – ધરાવે છે

. == તપાસે છે કે બે ઓપરેન્ડની કિંમતો સમાન છે કે નહીં. જો હા, તો શરત સાચી બને છે.

(A == B) સાચું નથી.

. != બે ઓપરેન્ડની કિંમતો સમાન છે કે નહીં તે તપાસે છે. જો મૂલ્યો સમાન ન હોય, તો સ્થિતિ સાચી બને છે. (A ! = B) સાચું છે.

. > ડાબા ઓપરેન્ડનું મૂલ્ય જમણા મૂલ્ય કરતાં વધારે છે કે કેમ તે તપાસે છે ઓપરેન્ડ જો હા, તો શરત સાચી બને છે. (A > B) સાચું નથી.

. < ડાબા ઓપરેન્ડનું મૂલ્ય જમણા મૂલ્ય કરતાં ઓછું છે કે કેમ તે તપાસે છે ઓપરેન્ડ જો હા, તો શરત સાચી બને છે. (A < B) સાચું છે.

. >= ડાબા ઓપરેન્ડનું મૂલ્ય જમણા ઓપરેન્ડના મૂલ્ય કરતાં વધારે કે બરાબર છે કે કેમ તે તપાસે છે. જો હા, તો શરત સાચી બને છે. (A >= B) સાચું નથી.

. <= તપાસે છે કે ડાબા ઓપરેન્ડનું મૂલ્ય (A <= B) ના મૂલ્ય કરતાં ઓછું અથવા બરાબર છે કે નહીં. યોગ્ય ઓપરેન્ડ. જો હા, તો શરત સાચી બને છે.

દાખ્લા તરીકે:

```
# સમાવેશ થાય છે <stdio.h>
મુખ્ય() {
Int a = 21;
પૂર્ણાંક b = 10;
પૂર્ણાંક c;
જો ( a == b ) {
Printf("લાઇન 1 - a બરાબર b\n");
   } બીજું {
Printf("લાઇન 1 - a એ b\n" ની બરાબર નથી); }
જો ( a< b ) {
Printf("લાઇન 2 - a b\n" કરતા ઓછી છે);
   } બીજું {
Printf("લાઇન 2 - a b\n" કરતા ઓછી નથી);
   }=
જો ( a > b ) {
Printf("લાઇન 3 - a b\n" કરતાં મોટી છે);
   } બીજું {
Printf("લાઇન 3 - a b\n" કરતા મોટી નથી);}
   /* ચાલો a અને b ની કિંમત બદલીએ */
A = 5;
બી = 20;
જો ( a<= b ) {
Printf("લાઈન 4 - a કાં તો b\n કરતા ઓછી અથવા બરાબર છે");
   }
જો ( b> = a ) {
Printf("રેખા 5 - b કાં તો b\n" થી મોટી અથવા બરાબર છે);
```

```
} }
```

આઉટપુટ:

લીટી 1 - a એ b ની બરાબર નથી
લીટી 2 - a એ b કરતા ઓછી નથી
લીટી 3 - a એ b કરતા મોટી છે
પંક્તિ 4 - a કાં તો b કરતા ઓછી અથવા બરાબર છે
પંક્તિ 5 - b કાં તો b કરતાં મોટી અથવા બરાબર છે

4. લોજિકલ ઓપરેટરોને ઉદાહરણ સાથે સમજાવો. (4 M)

જવાબ: C ભાષા દ્વારા સમર્થિત લોજિકલ ઓપરેટરો. ધારો કે ચલ A 1 અને ધરાવે છે
ચલ B 0 ધરાવે છે.

1. &&કહેવાતા લોજિકલ અને ઓપરેટર. જો બંને ઓપરેન્ડ બિન-શૂન્ય હોય, તો સ્થિતિ સાચી બને છે.

(A&&B) ખોટું છે.

સાચી બને છે. (A લોજિકલ ઓપરેટર કહેવાય છે. જો બે ઓપરેન્ડમાંથી કોઈપણ બિન-શૂન્ય હોય, તો 2. || સ્થિતિ

Logical NOT Operator કહેવાય છે. તેનો ઉપયોગ તેના ઓપરેન્ડની તાર્કિક સ્થિતિને વિપરીત કરવા માટે થાય છે. જો
કોઈ શરત સાચી હોય, તો Logical NOT ઓપરેટર તેને ખોટી બનાવશે.

!(A&&B) સાચું છે.

દાખ્લા તરીકે:

```
#include <stdio.h> Main() { Int
a = 5; ઈન્ટ બી = 20;
પૂર્ણાંક c; જો ( a&&b )
{ Printf("લાઇન 1 - શરત
સાચી છે\n" );}


જો ( a || b )
{ Printf("લાઇન 2 - શરત સાચી છે\n"); }
A = 0;
બી = 10;
જો ( a&&b )
{ Printf("લાઇન 3 - શરત સાચી છે\n"); } અન્ય
   { Printf("લાઇન 3 - શરત સાચી નથી\n");

   }
જો ( !(a && b) )
{ Printf("લાઇન 4 - શરત સાચી છે\n");

} }
```

આઉટપુટ:

લાઇન 1 - શરત સાચી છે
લાઇન 2 - શરત સાચી છે
લાઇન 3 - સ્થિતિ સાચી નથી

લાઇન 4 - શરત સાચી છે

5. અસાઇનમેન્ટ ઓપરેટરોને ઉદાહરણ સાથે સમજાવો. (4 M)
જવાબ: નીચેનું કોષ્ટક C ભાષા દ્વારા સમર્થિત અસાઇનમેન્ટ ઓપરેટરોની યાદી આપે છે

1. = સરળ સોંપણી ઓપરેટર. ને જમણી બાજુના ઓપરેન્ડથી મૂલ્યો અસાઇન કરે છે
ડાબી બાજુ ઓપરેન્ડ, ઉદાહરણ તરીકે:C = A + B એ A + B ની કિંમત C ને સોંપશે
2. += ઉમેરો અને સોંપણી ઓપરેટર. તે ડાબા ઓપરેન્ડમાં જમણો ઓપરેન્ડ ઉમેરે છે અને પરિણામ ડાબા ઓપરેન્ડને સોંપે છે.
ઉદાહરણ તરીકે:C += A એ C = C + A ની સમકક્ષ છે

3. -= બાદબાકી અને સોંપણી ઓપરેટર. તે ડાબા ઓપરેન્ડમાંથી જમણા ઓપરેન્ડને બાદ કરે છે અને પરિણામ ડાબા ઓપરેન્ડને સોંપે છે. ઉદાહરણ તરીકે:C -= A એ C = C - A ની સમકક્ષ છે

4. *= ગુણાકાર અને સોંપણી ઓપરેટર. તે ડાબા ઓપરેન્ડ સાથે જમણા ઓપરેન્ડનો ગુણાકાર કરે છે અને પરિણામ ડાબા ઓપરેન્ડને સોંપે છે. ઉદાહરણ તરીકે:C *= A એ C = C * A ની સમકક્ષ છે

5. /= વિભાજન અને સોંપણી ઓપરેટર. તે ડાબા ઓપરેન્ડને જમણા ઓપરેન્ડ સાથે વિભાજીત કરે છે અને પરિણામ ડાબા ઓપરેન્ડને સોંપે છે. ઉદાહરણ તરીકે /= A એ C = C/A ની સમકક્ષ છે

%= મોડ્યુલસ અને અસાઇનમેન્ટ ઓપરેટર. તે બે ઓપરેન્ડનો ઉપયોગ કરીને મોડ્યુલસ લે છે અને ડાબા ઓપરેન્ડને પરિણામ સોંપે છે. ઉદાહરણ તરીકે:C %= A એ C = C % A ની સમકક્ષ છે

<<=લેફ્ટ શિફ્ટ અને અસાઇનમેન્ટ ઓપરેટર . ઉદાહરણ તરીકે:C <<= 2 એ C = C << 2 સમાન છે

>>=રાઇટ શિફ્ટ અને અસાઇનમેન્ટ ઓપરેટર . ઉદાહરણ તરીકે:C >>= 2 એ C = C >> 2 જેવું જ છે

&= bitwise AND assignment operator . ઉદાહરણ તરીકે: C &= 2 એ C = C & 2 સમાન છે

^= બીટવાઇઝ એક્સક્લુઝિવ અથવા અને અસાઇનમેન્ટ ઓપરેટર. ઉદાહરણ તરીકે:C ^= 2 એ C = C ^ 2 સમાન છે

|= બિટવાઇઝ સમાવિષ્ટ અથવા અને અસાઇનમેન્ટ ઓપરેટર. ઉદાહરણ તરીકે:C |= 2 એ C = C | સમાન છે 2

ઉદાહરણ:

```
# સમાવેશ થાય છે <stdio.h>
મુખ્ય() {
int a = 21;
int c ;
   c = a;
printf("લાઇન 1 -        = ઓપરેટર ઉદાહરણ, c નું મૂલ્ય = %d\n", c );
   c += a;
printf("લાઇન 2 - += ઓપરેટર ઉદાહરણ, c નું મૂલ્ય = %d\n", c);
   c -= a;
printf("લાઇન 3 - -= ઓપરેટરનું ઉદાહરણ, c નું મૂલ્ય = %d\n", c);
```

```
c  *=  a;
printf("લાઇન 4 - c /=  *=  ઓપરેટરનું ઉદાહરણ, c નું મૂલ્ય = %d\n", c );
a;
printf("લાઇન 5 - /= ઓપરેટર ઉદાહરણ, c = %d\n" નું મૂલ્ય, c);
c = 200;
c % = a;
printf("લાઇન 6 - %= ઓપરેટરનું ઉદાહરણ, c નું મૂલ્ય = %d\n", c);
   c <<= 2;
printf("લાઇન 7 - <<= ઓપરેટર ઉદાહરણ, c = %d\n", c નું મૂલ્ય);
   c >>= 2;
printf("લાઈન 8 - >>= ઓપરેટરનું ઉદાહરણ, c = %d\n", c નું મૂલ્ય);
c&= 2;
printf("લાઇન 9 - &= ઓપરેટર ઉદાહરણ, c = %d\n" નું મૂલ્ય, c);
c  ^= 2;
printf("લાઇન 10 -        ^= ઓપરેટરનું ઉદાહરણ, c નું મૂલ્ય = %d\n", c );
c |= 2;
printf("લાઇન 11 - |= ઓપરેટર ઉદાહરણ, c નું મૂલ્ય = %d\n", c);
}
```

આઉટપુટ:

```
લાઇન 1 -  = ઓપરેટરનું ઉદાહરણ, c = 21 નું મૂલ્ય
લાઇન 2 - += ઓપરેટરનું ઉદાહરણ, c = 42 નું મૂલ્ય
લાઇન 3 - -= ઓપરેટરનું ઉદાહરણ, c = 21 નું મૂલ્ય
પંક્તિ 4 -  *= ઓપરેટરનું ઉદાહરણ, c નું મૂલ્ય = 441
પંક્તિ 5 - /= ઓપરેટરનું ઉદાહરણ, c = 21 નું મૂલ્ય
લાઇન 6 - %= ઓપરેટરનું ઉદાહરણ, c = 11 નું મૂલ્ય
લાઇન 7 - <<= ઓપરેટરનું ઉદાહરણ, c = 44 નું મૂલ્ય
લીટી 8 - >>= ઓપરેટરનું ઉદાહરણ, c = 11 નું મૂલ્ય
પંક્તિ 9 - &= ઓપરેટરનું ઉદાહરણ, c = 2 નું મૂલ્ય
લાઇન 10 -  ^= ઓપરેટરનું ઉદાહરણ, c = 0 નું મૂલ્ય
લાઇન 11 - |= ઓપરેટરનું ઉદાહરણ, c = 2 નું મૂલ્ય
```

6. શરતી ઓપરેટરોને ઉદાહરણ સાથે સમજાવો. (4 M)

જવાબ: કન્ડીશનલ ઓપરેટરને ટર્નરી ઓપરેટર તરીકે પણ ઓળખવામાં આવે છે. શરતી નિવેદનો એ નિર્ણય લેવાના નિવેદનો છે જે અભિવ્યક્તિના આઉટપુટ પર આધાર રાખે છે. તે બે પ્રતીકો દ્વારા રજૂ થાય છે, એટલે કે, '?'

વાક્યરચના:
(શરત)?અભિવ્યક્તિ 1:અભિવ્યક્તિ2;

ઉદાહરણ તરીકે:

```
#include<stdio.h>
પૂર્ણાંક મુખ્ય()
{
int m = 5, n = 4;
    (m > n) ?
printf("m એ n કરતા વધારે છે જે %d > %d",m, n છે):
printf("n એ m કરતાં મોટું છે જે %d > %d", n, m છે);
પરત 0;
}
```

આઉટપુટ:

m એ n કરતાં મોટો છે એટલે કે 5

7. ઉદાહરણ સાથે બીટવાઇઝ ઓપરેટરો સમજાવો. (4 M)

જવાબ: C માં, 6 ઓપરેટરો બીટવાઇઝ ઓપરેટર છે (બિટ-લેવલ પર કામ કરે છે)

C અથવા C++ માં & (bitwise AND) બે સંખ્યાઓને ઓપરેન્ડ તરીકે લે છે અને બે સંખ્યાના દરેક બીટ પર AND કરે છે. જો બંને બિટ્સ 1 હોય તો જ AND નું પરિણામ 1 છે.

આ | C અથવા C++ માં (bitwise OR) બે સંખ્યાઓ ઓપરેન્ડ તરીકે લે છે અને બે સંખ્યાના દરેક બીટ પર OR કરે છે. OR નું પરિણામ 1 છે જો બે બિટ્સમાંથી કોઈપણ 1 હોય.

C અથવા C++ માં ^ (bitwise XOR) બે સંખ્યાઓને ઓપરેન્ડ તરીકે લે છે અને બે સંખ્યાના દરેક બીટ પર XOR કરે છે. જો બે બિટ્સ અલગ હોય તો XOR નું પરિણામ 1 છે.

C અથવા C++ માં << (ડાબી પાળી) બે સંખ્યાઓ લે છે, ડાબી બાજુ પ્રથમ ઓપરેન્ડના બિટ્સને શિફ્ટ કરે છે, બીજું ઓપરેન્ડ સ્થળાંતર કરવાના સ્થાનોની સંખ્યા નક્કી કરે છે.

C અથવા C++ માં >> (જમણી પાળી) બે સંખ્યાઓ લે છે, જમણી બાજુ પ્રથમ ઓપરેન્ડના બિટ્સને શિફ્ટ કરે છે, બીજું ઓપરેન્ડ શિફ્ટ કરવા માટેની જગ્યાઓની સંખ્યા નક્કી કરે છે.

C અથવા C++ માં ~ (bitwise NOT) એક નંબર લે છે અને તેના તમામ બિટ્સને ઉલટાવે છે ઉદાહરણ તરીકે:

```
#include
<stdio.h>

પૂર્ણાંક મુખ્ય()
{
    // a = 5(00000101), b = 9(00001001)
સહી વિનાનું char a = 5, b = 9;
    // પરિણામ 00000001 છે
printf("a = %d, b = %d\n", a, b);
printf("a&b = %d\n", a & b);
    // પરિણામ 00001101 છે
printf("a|b = %d\n", a | b);
    // પરિણામ 00001100 છે
```

printf("a^b = %d\n", a^b);
// પરિણામ 11111010 છે
printf("~a = %d\n", a = ~a);
// પરિણામ 00010010 છે
printf("b<<1 = %d\n", b << 1);
// પરિણામ 00000100 છે
printf("b>>1 = %d\n", b >> 1);
પરત 0;
}

આઉટપુટ:
a = 5, b = 9
a&b = 1
a|b = 13
a^b = 12
~a = 250
b<<1 = 18
b>1 = 4

8. અંકગણિત ઓપરેટરનો ઉપયોગ કરીને કેલ્ક્યુલેટર ડિઝાઇન કરવા માટે એક પ્રોગ્રામ લખો. (4 M)
જવાબ: #include <stdio.h>
int main() {

// સ્થાનિક વેરીએબલ્સ ચાર
ઓપ્ટ જાહેર કરો; int n1, n2; ફ્લોટ
રેસ; printf (" c કેલ્ક્યુલેટરમાં ઓપરેશન
કરવા માટે ઓપરેટર (+, -, *, /) પસંદ
કરો \n "); scanf ("%c", &opt); // ઓપરેટર પ્રિન્ટ્ફ લો (" પ્રથમ નંબર દાખલ કરો: "); scanf("%d", &n1); // મુઠ્ઠી
નંબર printf લો ("બીજો નંબર દાખલ કરો:"); scanf ("%d", &n2); // બીજો નંબર લો જો (opt == '+') {

res = n1 + n2; // બે નંબરો ઉમેરો printf ("%d
અને %d નો ઉમેરો છે: %f", n1, n2, res);

} અન્ય જો (opt ==
'-') {
res = n1 - n2; // બે નંબરો પ્રિન્ટ્ફ બાદ કરો ("%d
અને %d ની બાદબાકી છે: %f", n1, n2, res);

} અન્ય જો (opt ==
'*') { res = n1 *
n2; // બે સંખ્યાઓનો ગુણાકાર કરો

printf ("%d અને %d નો ગુણાકાર છે: %f", n1, n2, res);

} અન્ય જો (opt ==
'/') {
જો (n2 == 0) // જો n2 == 0 હોય, તો બીજી સંખ્યા લો
{
printf (" \n વિભાજક શૂન્ય ન હોઈ શકે. કૃપા કરીને બીજી કિંમત દાખલ કરો "); scanf ("%d",
&n2); }

res = n1 / n2; // બે નંબરોને વિભાજિત કરો printf
("%d અને %d નો વિભાગ છે: %.2f", n1, n2, res);

} બીજું
{
printf(" \n તમે ખોટા ઇનપુટ્સ દાખલ કર્યા છે");

} વળતર
0; }

આઉટપુટ:

```
Select an operator (+, -, *, /) to perform an operation in C calculator
/
Enter the first number: 20
Enter the second number: 0

Divisor cannot be zero. Please enter another value 5
Division of 20 and 5 is: 4.00
```

9. Bitwise Operator નો ઉપયોગ કરીને બે નંબરો સ્વેપ કરવા માટે એક પ્રોગ્રામ લખો.
(4 M)

જવાબ:

```
#include<stdio.h>
int main(){
જ્યારે, b;
printf("a અને b માટે મૂલ્યો દાખલ કરો:");
scanf("%d%d",&a,&b);
printf("સ્વેપ પહેલાં a=%d અને b=%d નું મૂલ્ય\n",a,b);
   a= a^b;
   b= a^b;
   a= a^b;
printf("સ્વેપ પછી a=%d અને b=%d નું મૂલ્ય",a,b);
પરત 0;
}
```

આઉટપુટ:

a અને b માટે કિંમતો દાખલ કરો:24 56
સ્વેપ પહેલા a=24 અને b=56 ની કિંમત
સ્વેપ પછી a=56 અને b=24 ની કિંમત

1. સરળ જો નિવેદનો સમજાવો. (2 M)

જવાબ: જો વિધાન નીચે મુજબ હોય તો સરળનું સામાન્ય વાક્યરચના:

. વાક્યરચના:

જો (અભિવ્યક્તિ)

{

નિવેદન:

વિધાન 2;

}

નિવેદન - x;

. ઉપરોક્ત સિન્ટેક્સમાં સ્ટેટમેન્ટ બ્લોક એક સ્ટેટમેન્ટ અથવા એક જૂથ હોઈ શકે છે

નિવેદનો

. સરળ જો નિવેદન નીચેના ફ્રમમાં ચલાવવામાં આવે છે.

1. પ્રથમ સ્થિતિ તપાસવામાં આવે છે.

2. જો શરત સાચી હોય તો સ્ટેટમેન્ટ બ્લોક એક્ઝિક્યુટ થાય છે અને પછી સ્ટેટમેન્ટ-x

ચલાવવામાં આવે છે.

. જો શરત ખોટી હોય તો માત્ર સ્ટેટમેન્ટ –x એક્ઝિક્યુટ થાય છે.

2. જો....અન્ય નિવેદનો સમજાવો. (3 M)

વર્ષ:

. if..else સ્ટેટમેન્ટનું સામાન્ય વાક્યરચના નીચે મુજબ છે:

. વાક્યરચના:

જો (પરીક્ષણ અભિવ્યક્તિ)

{

ટ્રુ-બ્લોક સ્ટેટમેન્ટ્સ;

}

બીજું

{

ખોટા-બ્લોક નિવેદનો;

}

. if..else સ્ટેટમેન્ટ નીચેના ફોર્મમાં ચલાવવામાં આવે છે:

1. પ્રથમ સ્થિતિ તપાસવામાં આવે છે.

2. જો શરત સાચી હોય તો સાચું સ્ટેટમેન્ટ એક્ઝિક્યુટ થાય છે.

3. જો શરત ખોટી હોય તો ખોટું નિવેદન ચલાવવામાં આવે છે.

3. ઉદાહરણ સાથે Nested if-else સ્ટેટમેન્ટ સમજાવો. (4 M)

વર્ષ:

. જ્યારે વધુ એક શરત ચકાસવાની હોય ત્યારે આપણે માળખાનો ઉપયોગ કરી શકીએ છીએ

જો..નહીં તો પહેલા શરત તપાસવામાં આવે છે.

. નેસ્ટિંગનું સિન્ટેક્સ if..else સ્ટેટમેન્ટ નીચે મુજબ છે:

વાક્યરચના:

જો (પરીક્ષણ શરત 1)

{

જો (પરીક્ષણ શરત 2)

{

સાચું-બ્લોક2 નિવેદનો;

}

બીજું

{

ખોટા-બ્લોક2 નિવેદનો;

}

બીજું

{

ખોટા-બ્લોક1 નિવેદનો;

}

. નેસ્ટિંગ if-else સ્ટેટમેન્ટ નીચેની રીતે ચલાવવામાં આવે છે.

1. પ્રથમ શરત 1 તપાસવામાં આવે છે.

2. જો શરત 1 સાચી હોય તો શરત 2 તપાસવામાં આવે છે. જો શરત 2 સાચી હોય તો

સ્ટેટમેન્ટ-1 ચલાવવામાં આવે છે.

3. પરંતુ જો શરત 2 ખોટી હોય તો સ્ટેટમેન્ટ-2 એક્ઝિક્યુટ થાય છે.

4. જો શરત 1 ખોટી હોય તો સ્ટેટમેન્ટ-3 એક્ઝિક્યુટ કરવામાં આવે છે.

ઉદાહરણ:

```
#include<stdio.h>
# સમાવેશ થાય છે <conio.h>
રદબાતલ મુખ્ય()
{
    int num1, num2, num3;
    printf("ત્રણ નંબરો દાખલ કરો:\n");
    scanf("%d%d%d",&num1, &num2, &num3);
    જો(સંખ્યા1>સંખ્યા2)
    {
     /* આ નેસ્ટેડ છે જો-નહીં */
     જો(સંખ્યા1>સંખ્યા3)
     {
          printf("Largest = %d", num1);
     }
     બીજું
     {
          printf("Largest = %d", num3);
     }
    }
    બીજું
    {
     /* આ નેસ્ટેડ છે જો-નહીં */
     જો(સંખ્યા2>સંખ્યા3)
     {
          printf("મોટા = %d", num2);
     }
     બીજું
     {
```

```
                printf("Largest = %d", num3);
        }
       }
       વળતર(0);
}
```

4. ઉદાહરણ સાથે જો અન્ય-જો નિસરણી સમજાવો. (4 M)

વર્ષ:

· if..else લેડર સ્ટેટમેન્ટ દ્વિ-માર્ગી નિર્ણય પૂરો પાડે છે જ્યાં આપણે એક પસંદ કરીએ છીએ
વૈકલ્પિક.

· તેનો ઉપયોગ બહુવિધ પસંદગી માટે થાય છે.

· જો..બીજું પૂરતું ન હોય તો બે માર્ગીય નિર્ણય નેસ્ટેડ દ્વારા કરવામાં આવે છે.

· નીચે if..else લેડરનું વાક્યરચના છે.

વાક્યરચના:

જો (શરત 1)

નિવેદન 1;

અન્ય જો (શરત 2)

નિવેદન 2;

અન્ય જો (શરત 3)

નિવેદન 3;

અન્ય જો (શરત n)

નિવેદન n;

બીજું

મૂળભૂત નિવેદન;

}

· if-else-જો નિસરણી નીચેના ક્રમમાં ચલાવવામાં આવે છે:

1. પ્રથમ શરત-1 અમલમાં છે , જો શરત-1 સાચું હોય તો વિધાન-1 છે
ચલાવવામાં આવે છે.

2. જો શરત-1 ખોટી હોય તો શરત-2 ચકાસવામાં આવે છે. જો શરત-2 સાચી હોય તો
સ્ટેટમેન્ટ-2 ચલાવવામાં આવે છે.

3. જ્યાં સુધી બધી સ્થિતિ તપાસવામાં ન આવે ત્યાં સુધી આ પ્રક્રિયા પુનરાવર્તિત થાય છે. જો બધી સ્થિતિ
ખોટા બન્યા પછી ડિફોલ્ટ સ્ટેટમેન્ટ ચલાવવામાં આવે છે.

ઉદાહરણ:

```
#include<stdio.h>
# સમાવેશ થાય છે <conio.h>
પૂર્ણાંક મુખ્ય()
{
    int a,b,c;
    printf("ત્રણ નંબરો દાખલ કરો: \n");
    scanf("%d%d%d", &a, &b, &c);
    જો (a>b && a>c)
    {
        printf("સૌથી મોટી = %d", a);
    }
    બાકી જો(b>a &&b>c)
    {
        printf("સૌથી મોટી = %d", b);
    }
    બીજું
    {
        printf("મોટા = %d", c);
    }
    વળતર(0);
}
```

5. ઉદાહરણ સાથે સ્વિચ ...કેસ સ્ટેટમેન્ટ સમજાવો. (4 M)

વર્ષ:

. સ્વિચ સ્ટેટમેન્ટને બહુ-પસંદગી અથવા બહુવિધ નિર્ણય નિવેદન તરીકે પણ ઓળખવામાં આવે છે.

. બહુવિધ if..else નો ઉપયોગ કરીને કોડ લખવો લાંબો અને મુશ્કેલ પણ બને છે

વ્યવસ્થા કરો સ્વિચ સ્ટેટમેન્ટનો ઉપયોગ કરીને તે સરળ રીતે કરવામાં આવે છે.

.

તે ચલ અથવા અભિવ્યક્તિના દરેક મૂલ્ય માટે પસંદગી પ્રદાન કરે છે.

. સ્વિચ સ્ટેટમેન્ટ કેસની સૂચિ સામે આપેલ ચલના મૂલ્યનું પરીક્ષણ કરે છે

મૂલ્યો અને જ્યારે મેળ મળે છે, ત્યારે કેસ સાથે સંકળાયેલ નિવેદન છે

ચલાવવામાં આવે છે.

. વાક્યરચના:

સ્વિચ કરો (ચલ નામ અથવા અભિવ્યક્તિ)

સ્વિચ (વર્ષ) {

કેસ const-expr 1: સ્ટેટમેન્ટ 1;

વિરામ;

કેસ const-expr 2: સ્ટેટમેન્ટ 2;

વિરામ;

કેસ const-expr3: સ્ટેટમેન્ટ 3;

વિરામ;

મૂળભૂત:

નિવેદનો;

}

. અભિવ્યક્તિ અથવા ચલ નામ એ પૂર્ણાંક અથવા અક્ષરો છે.

. મૂલ્ય-1,મૂલ્ય-2 એ સ્થિર છે અથવા કેસ લેબલ તરીકે ઓળખાય છે, દરેક કેસ લેબલ મૂલ્ય હોવું આવશ્યક છે

સ્વિચ સ્ટેટમેન્ટ સાથે અનન્ય બનો. દરેક કેસ લેબલ (;) સાથે સમાપ્ત થવું જોઈએ.

. સ્વિચ..કેસ સ્ટેટમેન્ટ નીચેના ક્રમમાં ચલાવવામાં આવે છે:

1. અભિવ્યક્તિના મૂલ્યની કેસ લેબલના મૂલ્ય સાથે સરખામણી કરવામાં આવે છે.

2. જો એવો કેસ જોવા મળે કે જેનું મૂલ્ય અભિવ્યક્તિના મૂલ્ય સાથે મેળ ખાતું હોય, તો પછી

કેસ સાથે સંકળાયેલ નિવેદન ચલાવવામાં આવે છે. ત્યાં એક નિવેદન છે અથવા

બહુવિધ નિવેદન. ત્યાં વિરામ છે જે નિયંત્રણને આગામી નિવેદનમાં મોકલે છે.

3. જો અભિવ્યક્તિનું મૂલ્ય કોઈપણ કેસ મૂલ્ય સાથે મેળ ખાતું ન હોય તો

ડિફોલ્ટ કેસ સાથે સંકળાયેલ નિવેદન ચલાવવામાં આવે છે.

ઉદાહરણ:

```
# સમાવેશ થાય છે <stdio.h>
# સમાવેશ થાય છે <conio.h>
પૂર્ણાંક મુખ્ય()
{
    પૂર્ણાંક સંખ્યા = 2;
    સ્વિચ કરો(સંખ્યા+2)
    {
        કેસ 1:
            printf("કેસ1: મૂલ્ય છે: %d", સંખ્યા);
        કેસ 2:
            printf("કેસ1: મૂલ્ય છે: %d", સંખ્યા);
        કેસ 3:
            printf("કેસ1: મૂલ્ય છે: %d", સંખ્યા);
        મૂળભૂત:
            printf("ડિફોલ્ટ: મૂલ્ય છે: %d", સંખ્યા);
    }
    પરત 0;
}
```

6. ઉદાહરણ સાથે બ્રેક સ્ટેટમેન્ટ સમજાવો. (4 M)

વર્ષ:

. અમે સ્વીચ સ્ટેટમેન્ટ સાથે બ્રેક સ્ટેટમેન્ટનો ઉપયોગ કર્યો છે.

. બ્રેક સ્ટેટમેન્ટનું કાર્ય સ્વિચ બોડીમાંથી બહાર નીકળવાનું છે.

. જો તે દરેક કેસ સ્ટેટમેન્ટ પછી લખાયેલું ન હોય, તો પછીના સ્ટેટમેન્ટ પર કંટ્રોલ પાસ કરો

,તેથી આગળના કેસ સ્ટેટમેન્ટનું બાકીનું સ્ટેટમેન્ટ પણ એક્ઝિક્યુટ થશે જો કે

કેસ મૂલ્ય મેળ ખાતું નથી અને પ્રોગ્રામ યોગ્ય રીતે કાર્ય કરશે નહીં.

ઉદાહરણ:

```
# સમાવેશ થાય છે <stdio.h>
# સમાવેશ થાય છે <conio.h>
પૂર્ણાંક મુખ્ય()
```

```
{
int i;
માટે(i=10;i>0; i--)
 {
  જો(i==6)
   {
     printf("\n લૂપ માટે બહાર આવી રહ્યું છે જ્યાં i = %d\n", i);
     વિરામ;
   }
 printf("%d",i);
 }}
```

7. If...Else અને સ્વિચ કેસ વચ્ચે શું તફાવત છે. (4 M)

વર્ષ:

જો બીજુ	સ્વિચ કરો
જો સ્ટેટમેન્ટ પસંદ કરવા માટે વપરાય છે બે વિકલ્પો વચ્ચે	સ્વીચ સ્ટેટમેન્ટનો ઉપયોગ થાય છે બહુવિધ વચ્ચે પસંદ કરવા માટે વિકલ્પો
જો પર આધારિત મૂલ્યો હોઈ શકે છે અવરોધો	સ્વિચ આધારિત મૂલ્યો હોઈ શકે છે વપરાશકર્તા પસંદગી પર.
જો રેખીય શોધ લાગુ કરે છે.	દ્વિસંગી અમલીકરણ સ્વિચ કરો શોધ
ફ્લોટ, ડબલ, ચાર, પૂર્ણાંક અને અન્ય ડેટા પ્રકારો કરી શકે છે જો શરતમાં ઉપયોગ કરો.	ફક્ત પૂર્ણાંક અને ચાર ડેટા પ્રકારો સ્વીચમાં વાપરી શકાય છે બ્લોક
જો-બીજું સંપાદિત કરવું મુશ્કેલ છે સ્ટેટમેન્ટ, જો નેસ્ટેડ if-else વિધાન વપરાય છે.	સ્વિચ કેસોને સંપાદિત કરવું સરળ છે જેમ કે, તેઓ ઓળખાય છે સરળતાથી

8. બ્રેક અને ચાલુ રાખો વચ્ચેનો તફાવત આપો. (4 M)

વર્ષ:

વિરામ નિવેદન	નિવેદન ચાલુ રાખો
બ્રેક સ્ટેટમેન્ટનો ઉપયોગ લૂપ કન્સ્ટ્રક્ટ્સમાંથી બહાર નીકળવા માટે થાય છે.	ચાલુ સ્ટેટમેન્ટનો ઉપયોગ લૂપ કન્સ્ટ્રક્ટ્સમાંથી બહાર નીકળવા માટે થતો નથી.
બ્રેક સ્ટેટમેન્ટનો ઉપયોગ સામાન્ય રીતે સ્વીચ સ્ટેટમેન્ટ સાથે થાય છે, અને તે જ્યારે લૂપ, ડુ-વ્હાઈલ લૂપ અથવા ફોર લૂપમાં પણ તેનો ઉપયોગ કરી શકે છે.	ચાલુ સ્ટેટમેન્ટનો ઉપયોગ સ્વિચ સ્ટેટમેન્ટ સાથે થતો નથી, પરંતુ તેનો ઉપયોગ જ્યારે લૂપ, ડુ જ્યારે લૂપ અથવા ફોર-લૂપમાં થઈ શકે છે.
જ્યારે બ્રેક સ્ટેટમેન્ટ આવે છે ત્યારે નિયંત્રણ તરત જ લૂપ કન્સ્ટ્રક્ટમાંથી બહાર નીકળી જાય છે.	જ્યારે ચાલુ નિવેદનનો સામનો કરવો પડે છે ત્યારે નિયંત્રણ લૂપની શરૂઆતથી આપમેળે પસાર થાય છે નિવેદન
વાક્યરચના: વિરામ;	વાક્યરચના: ચાલુ રાખો

9. આપેલ કોઈ વિષમ અથવા બેકી છે તે શોધવા માટે પ્રોગ્રામ લખો. (4 M)

વર્ષ:

```
# સમાવેશ થાય છે <stdio.h>
# સમાવેશ થાય છે <conio.h>
રદબાતલ મુખ્ય()
 {
    int કે કેમ;
    printf("પૂર્ણાંક દાખલ કરો:");
    scanf("%d", &number);

    જો (નંબર % 2 == 0)
        printf("%d બરાબર છે.", num);
```

```
    બીજું

        printf("%d વિચિત્ર છે.", સંખ્યા);

        પરત 0;

}
```

10. સ્વિચ કેસનો ઉપયોગ કરીને દિવસો દર્શાવવા માટે એક પ્રોગ્રામ લખો. (4 M)

વર્ષ:

```
# સમાવેશ થાય છે <stdio.h>

 # સમાવેશ થાય છે <conio.h>

રદબાતલ મુખ્ય() {

    int દિવસ;

    printf("દિવસ નંબર દાખલ કરો (1 = સોમવાર ..... 7 = રવિવાર)\n");

    scanf("%d", &day);

     સ્વિચ(દિવસ){

        કેસ 1 : printf("સોમવાર\n");

            વિરામ;

        કેસ 2 : printf("મંગળવાર\n");

            વિરામ;

        કેસ 3 : printf("બુધવાર\n");

            વિરામ;

        કેસ 4 : printf("ગુરુવાર\n");

            વિરામ;

        કેસ 5 : printf("શુક્રવાર\n");

            વિરામ;

        કેસ 6 : printf("શનિવાર\n");

            વિરામ;

        કેસ 7 : printf("રવિવાર\n");

            વિરામ;

        ડિફોલ્ટ: printf("અમાન્ય ઇનપુટ !!!!\n");

    }

     પરત 0;

}
```

11. આપેલ નંબર ધન છે કે નકારાત્મક તે શોધવા માટે પ્રોગ્રામ લખો. (4M)

વર્ષ:

```
# સમાવેશ થાય છે <stdio.h>

# સમાવેશ થાય છે <conio.h>

રદબાતલ મુખ્ય() {

    ડબલ નંબર;

    printf("એક નંબર દાખલ કરો:");

    scanf("%lf", &number);

    જો (સંખ્યા <= 0) {

        જો (સંખ્યા == 0)

            printf("તમે 0 દાખલ કર્યું છે.");

        બીજું

            printf("તમે નકારાત્મક નંબર દાખલ કર્યો છે.");

    }

    બીજું

        printf("તમે હકારાત્મક નંબર દાખલ કર્યો છે.");

    પરત 0;

}
```

એકમ- 5 લૂપ કંટ્રોલ સ્ટેટમેન્ટ

1. લૂપ નિયંત્રણ નિવેદનો વ્યાખ્યાયિત કરો અને તેના પ્રકારોની સૂચિ બનાવો. (3 M)

જવાબ: લૂપ્સ એ કોડના બ્લોકના અમલને પુનરાવર્તિત કરવા માટે છે.
લૂપિંગ દરમિયાન સ્ટેટમેન્ટનો સમૂહ અમુક સુધી એક્ઝિક્યુટ કરવામાં આવે છે
સમાપ્તિ માટેની શરત અગણિત છે.

સામાન્ય લૂપ પ્રક્રિયામાં નીચેના ચાર પગલાં શામેલ હશે:

આઈ. કાઉન્ટરની શરૂઆત

II. સમાપ્તિની સ્થિતિ માટે પરીક્ષણ કરો

III. લૂપ બોડી સ્ટેટમેન્ટ

IV. કાઉન્ટર વધારો

. સ્થિતિ ક્યાં તપાસવામાં આવે છે તેના આધારે, અમારી પાસે બે પ્રકારના લૂપ હોઈ શકે છે
માળખું

1. પ્રવેશ નિયંત્રણ લૂપ.
2. એક્ઝિટ કંટ્રોલ લૂપ.

એન્ટ્રી કંટ્રોલ લૂપ: પહેલા શરત લખવામાં આવે છે અને પછી સ્ટેટમેન્ટનો મુખ્ય ભાગ. જો લૂપની બોડી પહેલાં કંડીશનની ચકાસણી કરવામાં આવે તો તેને એન્ટ્રી કંટ્રોલ લૂપ કહેવામાં આવે છે. જો કન્ડિશન સાચી હોય તો લૂપની બોડી એક્ઝિક્યુટ થાય છે અન્યથા લૂપ એક્ઝિક્યુટ થતી નથી.

. એક્ઝિટ કંટ્રોલ લૂપ: સ્ટેટમેન્ટનો મુખ્ય ભાગ પહેલા લખવામાં આવે છે પછી શરત લખવામાં આવે છે.
જો લૂપના મુખ્ય ભાગ પછી કંડીશન કન્ડીશનનું પરીક્ષણ કરવામાં આવે તો તેને એક્ઝિટ કંટ્રોલ લૂપ તરીકે ઓળખવામાં આવે છે. તેથી પ્રથમ લૂપનું મુખ્ય ભાગ ચલાવવામાં આવે છે અને પછી સ્થિતિ તપાસવામાં આવે છે.

2. પ્રથમ 10 નંગનો સરવાળો પ્રિન્ટ કરવા માટે એસી પ્રોગ્રામ લખો. લૂપ માટે ઉપયોગ કરીને. (4 M)
વર્ષ:

```
# સમાવેશ થાય છે <stdio.h>
# સમાવેશ થાય છે <conio.h>
int main() {
    int n, i, sum = 0;

    printf("ધન પૂર્ણાંક દાખલ કરો:");
    scanf("%d", &n);

    માટે (i = 1; i <= n; ++i) {
        રકમ += i;
    }

    printf("સમ = %d", સરવાળો);
    પરત 0;
}
```

3. લૂપ માટે નેસ્ટેડનો ઉપયોગ કરવાના રાજ્ય ફાયદા. (3 M)
વર્ષ:

. લૂપ માટે નેસ્ટેડ એ લૂપ અખરોટ છે જે બીજા લૂપની અંદર લૂપ ધરાવે છે. ફક્ત એક લૂપ
અન્ય લૂપની અંદર નેસ્ટેડ લૂપ તરીકે ઓળખવામાં આવે છે.

. જ્યારે આપણે વધુ સંખ્યામાં પુનરાવર્તનો સાથે કામ કરી રહ્યા હોઈએ ત્યારે તે ઉપયોગી છે.

. પ્રોગ્રામની લાઇનની સંખ્યા ઘટાડે છે

. કોડનો પુનઃઉપયોગ શક્ય છે

. મેમરી સાઈઝ વપરાશમાં ઘટાડો થશે.

4. જ્યારે લૂપ કરો ત્યારે તફાવત કરો. (3 M)
વર્ષ:

જ્યારે લૂપ	ડુ... જ્યારે લૂપ

જ્યારે (શરત) {
નિવેદનો; // લૂપનું શરીર
 	
}
	કરો{
	.
	નિવેદનો; // લૂપનું શરીર.
	
	.
	} જ્યારે(શરત);

'while' લૂપમાં લૂપની શરૂઆતમાં કંટ્રોલિંગ કન્ડીશન દેખાય છે.	'ડૂ-વ્હાઈલ' લૂપમાં લૂપના અંતમાં કન્ટ્રોલિંગ કન્ડીશન દેખાય છે.
પુનરાવૃત્તિઓ થતી નથી જો, પ્રથમ પુનરાવૃત્તિની સ્થિતિ ખોટી હોય.	પુનરાવૃત્તિ ઓછામાં ઓછી એક વાર થાય છે, ભલે પ્રથમ પુનરાવર્તનમાં સ્થિતિ ખોટી હોય.
પ્રવેશ-નિયંત્રિત લૂપ	બહાર નીકળો-નિયંત્રિત લૂપ

5. WHILE LOOP ઉદાહરણ સાથે સમજાવો. (4 M)
વર્ષ:

. જ્યારે લૂપ એ એન્ટ્રી કંટ્રોલ લૂપ છે. કારણ કે જ્યારે લૂપ પહેલા આ
સ્થિતિ તપાસવામાં આવે છે.

. જ્યારે લૂપનું સિન્ટેક્સ નીચે મુજબ છે.

. વાક્યરચના:

જ્યારે (શરત)
{
લૂપનું શરીર
}

. {} એ લૂપના મુખ્ય ભાગ તરીકે ઓળખાય છે. જો બૉડીમાં એક સ્ટેટમેન્ટ હોય તો લૂપના બૉડીને બ્રેસ {}ની
જોડીમાં બંધ કરવું જરૂરી નથી.

. જ્યારે લૂપ નીચેના ફોર્મેટમાં ચલાવવામાં આવે છે.

. અહીં પહેલા શરતનું મૂલ્યાંકન કરવામાં આવે છે અને જો તે સાચું હોય તો માંનું નિવેદન

લૂપનું શરીર ચલાવવામાં આવે છે.

- બોડીને એક્ઝિક્યુટ કર્યા પછી, સ્થિતિનું ફરીથી મૂલ્યાંકન કરવામાં આવે છે અને જો તે સાચું હોય તો શરીરને ફરીથી ચલાવવામાં આવે છે. જ્યાં સુધી સ્થિતિ સાચી હોય ત્યાં સુધી આ પ્રક્રિયાને પુનરાવર્તિત કરવામાં આવે છે. એકવાર શરત ખોટી થઈ જાય પછી નિયંત્રણ બહાર નીકળી જાય છે.

ઉદાહરણ:

```
#include<stdio.h>
# સમાવેશ થાય છે <conio.h>
રદબાતલ મુખ્ય()
{
પૂર્ણાંક સંખ્યા, મર્યાદા;
printf ("તમારી ઉમર દાખલ કરો:" );
scanf ("%d", &limit );
જ્યારે (ગણતરી<=મર્યાદા)
{
printf (“%d”, ગણતરી);
કાઉન્ટ=કાઉન્ટ+1;
}
getch();
}
```

6. ઉદાહરણ સાથે DO...WHILE LOOP સમજાવો. (4 M)

વર્ષ:

- Do..while લૂપ એ એક્ઝિટ કંટ્રોલ લૂપ છે.
- નીચે એક્ઝિટ કંટ્રોલ લૂપનું સિન્ટેક્સ છે. કારણ કે લૂપના શરીરને એક્ઝેક્યુટ કર્યા પછી સ્થિતિ તપાસવામાં આવે છે.
- વાક્યરચના:

```
કરો
{
લૂપનું શરીર
}જ્યારે(શરત);
```

- ડુ જ્યારે લૂપ નીચેના ફોર્મેટમાં ચલાવવામાં આવે છે.
- ડુ..વહીલ લૂપમાં બોડી એક્ઝિક્યુટ થાય છે અને પછી કન્ડીશન ચેક કરવામાં આવે છે.
- જો સ્થિતિ સાચી હોય તો શરીરને ફરીથી અને ફરીથી ચલાવવામાં આવે છે, જ્યાં સુધી શરત સાચી છે.
- નિયંત્રણ એકવાર લૂપમાંથી બહાર નીકળી જાય છે, સ્થિતિ ખોટી બની જાય છે.
- do..while લૂપ એ એક્ઝિટ કંટ્રોલ લૂપ છે તેથી પ્રથમ બોડી એક્ઝિક્યુટ થાય છે અને શરત તપાસવામાં આવે છે. આ સુનિશ્ચિત કરે છે કે લૂપનો મુખ્ય ભાગ એક વાર લીઝ પર એક્ઝિક્યુટ કરવામાં આવે છે, ભલે શરત પહેલી વખત ખોટી હોય.

ઉદાહરણ:

```
#include <stdio.h>
પૂર્ણાંક મુખ્ય()
{
ડબલ નંબર, સરવાળો = 0;
કરવું
{
```

```
printf("એક નંબર દાખલ કરો:");
scanf("%lf", &number); સરવાળો += સંખ્યા;

}
જ્યારે(સંખ્યા != 0.0);
printf("સમ = %.2lf", રકમ);
પરત 0;
}
```

7. 2 થી 100 ની વચ્ચે અવિભાજ્ય સંખ્યાઓ શોધવા માટે નેસ્ટેડ ફોર લૂપ્સનો ઉપયોગ કરીને ac પ્રોગ્રામ લખો. (4 M)

વર્ષ:

```
#include<stdio.h>
#include<math.h> void main()
  { int start, end, num, count,
prime, temp, inum;
    printf("પ્રારંભ અને અંતિમ મૂલ્ય દાખલ કરો\n"); scanf("%d%d",
      &start, &end); જો (પ્રારંભ > અંત) {



        temp = શરુઆત;
        શરૂઆત = અંત;
        અંત = તાપમાન; }
    printf("%d અને %d
      વચ્ચેના મુખ્ય નંબરો\n છે", પ્રારંભ, અંત); for(num = start; num <= end; num++) { prime = 1;
      inum = sqrt(num); માટે(કાઉન્ટ = 2; ગણતરી <= ઇનમ; કાઉન્ટ++) { જો(સંખ્યા % કાઉન્ટ
    == 0) { પ્રાઇમ = 0; વિરામ;









          }

        } if(prime) printf("%d,\t", num);

    } વળતર 0;
}
```