

Programming in C (4331105) - Winter 2023 Solution

Milav Dabgar

May 21, 2024

પ્રશ્ન 1 [a ગુણ]

3 અભોરિધમ વ્યાખ્યાયિત કરો અને વર્તુળનું ક્ષેત્રફળ શોધવા માટેનું અભોરિધમ લખો.

જવાબ

જવાબ: અભોરિધમ એટલે કોઈ ચોક્કસ સમર્થાના ઉકેલ માટેની પગલાવાર પ્રક્રિયા અથવા નિયમોનો સમૂહ. વર્તુળના ક્ષેત્રફળનું અભોરિધમ:

- શરૂ
- વર્તુળની ત્રિજ્યા (r) ઇનપુટ લો
- ક્ષેત્રફળ = $\pi \times r^2$ ની ગણતરી કરો
- ક્ષેત્રફળ દર્શાવો
- અંત

મેમરી ટ્રીક

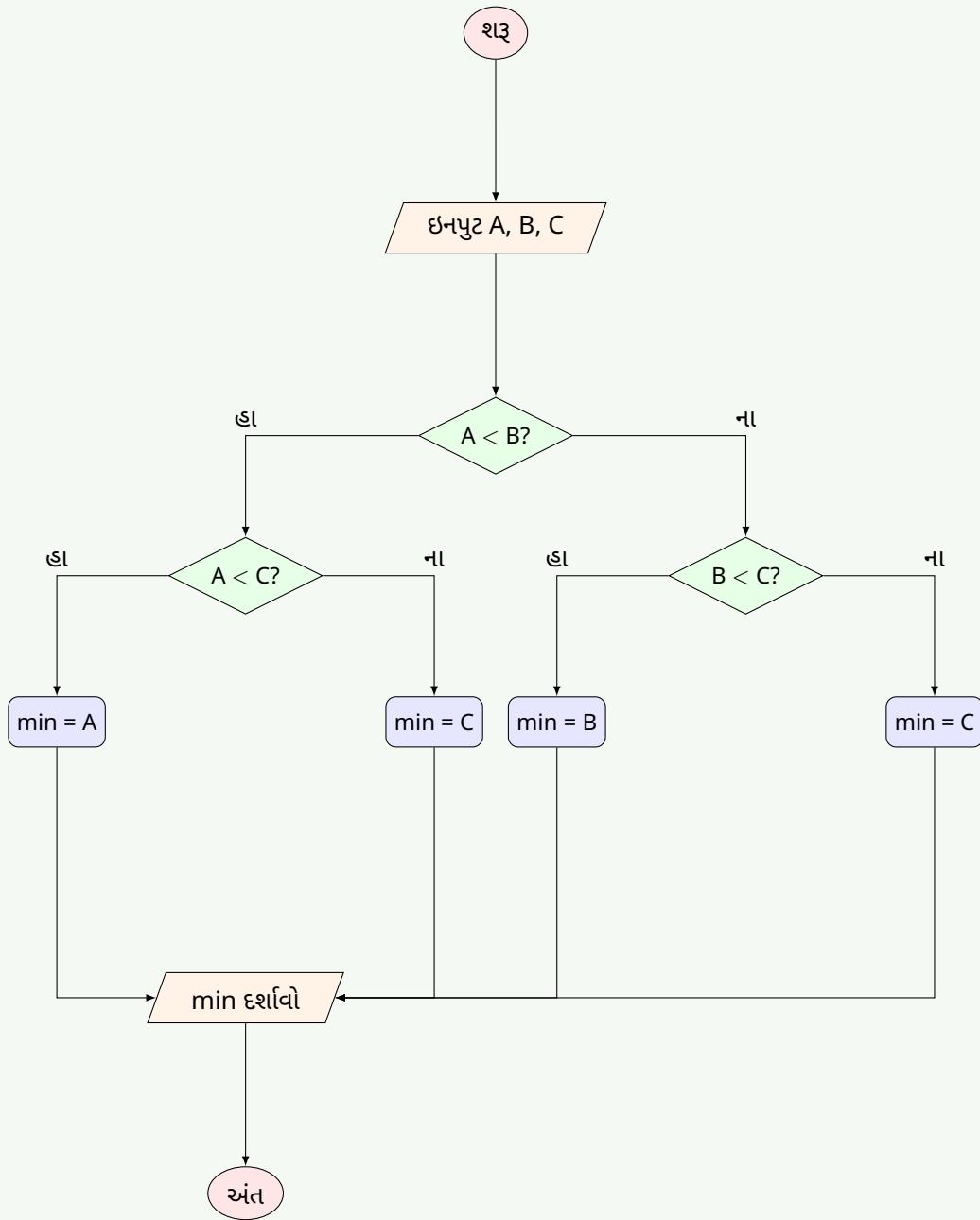
"શરૂ, વાંચો, ગણતરી, પ્રદર્શન, અંત"

પ્રશ્ન 1 [b ગુણ]

4 ફ્લોચાર્ટ વ્યાખ્યાયિત કરો અને આપેલ ત્રણ સંખ્યાઓ માંથી ઓછામાં ઓછી સંખ્યા શોધવા માટેનો ફ્લોચાર્ટ દોરો.

જવાબ

જવાબ: ફ્લોચાર્ટ એટલે પ્રમાણિત પ્રતીકો અને આકારોનો ઉપયોગ કરીને એલોરિધમનું દ્રશ્ય નિરૂપણ, જે પગલાઓના ક્રમને દર્શાવે છે. ત્રણ સંખ્યાઓમાંથી ન્યૂનતમ શોધવા માટેનો ફ્લોચાર્ટ:



- તુલના વ્યૂહરચના: પહેલા A અને B ની તુલના કરો, પછી C સાથે તુલના કરો
- ભાન્ધિંગ લોજિક: સૌથી નાની કિમત શોધવા માટે if-else સ્ટ્રક્ચરનો ઉપયોગ કરો

મેમરી ટ્રીક

"જોડાઓની તુલના કરો, દુર્લભ નાની કિમત દરેક જગ્યાએ શોધો"

પ્રશ્ન 1 [C ગુણ]

7 નીચેના સમીકરણનો ઉપયોગ કરીને સિખ્પલ ઇન્ટરેસ્ટની ગણતરી કરવા માટેનો પ્રોગ્રામ લખો. $I=PRN/100$ જ્યાં $P=$ પ્રિન્સીપલ રકમ, $R=$ વ્યાજનો દર અને $N=$ સમયગાળો.

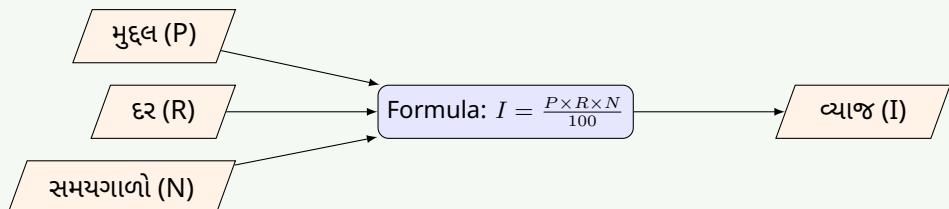
જવાબ

જવાબ:

```

1 #include <stdio.h>
2
3 int main() {
4     float P, R, N, I;
5
6     // પૂર્ણિસંપિલ રકમ, વ્યાજનો દર અને સમયગાળો ઇનપુટ લો
7     printf("પૂર્ણિસંપિલ રકમ દાખલ કરો: ");
8     scanf("%f", &P);
9
10    printf("વ્યાજનો દર દાખલ કરો: ");
11    scanf("%f", &R);
12
13    printf("સમયગાળો વર્ખમાં() દાખલ કરો: ");
14    scanf("%f", &N);
15
16    // સમીપલ ઇન્ટરેસ્ટની ગણતરી કરો
17    I = (P * R * N) / 100;
18
19    // પરિણામ દર્શાવો
20    printf("સમીપલ ઇન્ટરેસ્ટ = %.2f\n", I);
21
22    return 0;
23 }
```

આકૃતિ:



- ફ્લોટિંગ-પોઇન્ટ વેરિએબલ્સ: ચોક્સાઈ માટે દર્શાવું મૂલ્યો સ્ટોર કરે છે
- વપરાશકર્તા ઇન્ટરેક્શન: ઇનપુટ માટે સ્પષ્ટ પ્રોમ્પ્ટ્સ
- પરિણામ ફોર્મેટિંગ: %.2f બે દર્શાવું સ્થાન દર્શાવે છે

મેમરી ટ્રીક

"મુદ્દલ, દર અને સંખ્યા, સોથી ભાગીએ તો મળે વ્યાજ"

OR

પ્રશ્ન 1 [C ગુણ]

7 કીબોર્ડ દ્વારા ત્રિજ્યા(R) અને ઊંચાઈ(H) ઇનપુટ લઈ સિલિન્ડરના વોલ્યુમ(V)-ની ગણતરી કરીને પ્રિન્ટ કરવા માટેનો પ્રોગ્રામ લખો $V=\pi R^2 H$

જવાબ

જવાબ:

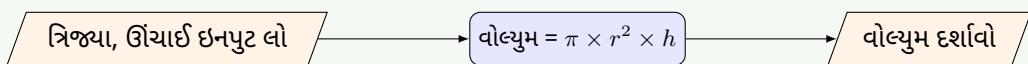
```

1 #include <stdio.h>
2
3 int main() {
```

```

4   float radius, height, volume;
5   const float PI = 3.14159;
6
7   // તરજીયા અને ઊંચાઈ ઇનપુટ લો
8   printf("સલિન્ડરની તરજીયા દાખલ કરો: ");
9   scanf("%f", &radius);
10
11  printf("સલિન્ડરની ઊંચાઈ દાખલ કરો: ");
12  scanf("%f", &height);
13
14  // સલિન્ડરના વોલ્યુમની ગણતરી કરો
15  volume = PI * radius * radius * height;
16
17  // પરિણામ દર્શાવો
18  printf("સલિન્ડરનું વોલ્યુમ = %.2f\n", volume);
19
20  return 0;
21 }
```

આફ્ટેટિંગ:



- કોન્સ્ટન્ટ્સ: સ્પષ્ટતા માટે PI કોન્સ્ટન્ટ તરીકે વ્યાખ્યાયિત કરવામાં આવ્યું છે
- ફોર્મ્યુલા: તરજીયાને બે વખત ગુણીને R^2 નો ઉપયોગ કરો
- ઇનપુટ વેલિડેશન: તરજીયા અને ઊંચાઈ માટે ધનાત્મક મૂલ્યોની ધારણા કરે છે

મેમરી ટ્રીક

"તરજીયાનો વર્ગ ગુણો ઊંચાઈ ગુણો પાઈ, આપે સિલિન્ડરનું વોલ્યુમ, ન પૂછો શા માટે"

પ્રશ્ન 2 [2 ગુણ]

3 સી પ્રોગ્રામિંગ ભાષામાં સપોર્ટ કરતા વિવિધ ઓપરેટરોની ચાહી બનાવો.

જવાબ

જવાબ:

| વર્ગ | ઓપરેટર્સ |
|--------------------------|--|
| અંકગણિત | +, -, *, /, % (સરવાળો, બાદબાકી, ગુણાકાર, ભાગાકાર, મોડ્યુલસ) |
| રિલેશનલ | ==, !=, >, <, >=, <= (સમાન, અસમાન, મોટું, નાનું, મોટું અથવા સમાન, નાનું અથવા સમાન) |
| લોજિકલ | &&, , ! (AND, OR, NOT) |
| એસાઇનમેન્ટ | =, +=, -=, *=, /=, %= (એસાઇન, પલસ-એસાઇન, માઇન્સ-એસાઇન, વગેરે) |
| ઇન્ક્રિમેન્ટ/ડિક્રિમેન્ટ | ++, -- (ઇન્ક્રિમેન્ટ, ડિક્રિમેન્ટ) |
| બિટવાઈજ | &, , ^, ~, <<, >> (AND, OR, XOR, કોમ્પલિમેન્ટ, લેફ્ટ શિફ્ટ, રાઇટ શિફ્ટ) |
| કન્ડિશનલ | ? : |
| સ્પેશિયલ | sizeof(), &, *, -, >, . (સાઈઝ, એડ્રેસ, પોઇન્ટર, સ્ટ્રક્ચર) |

મેમરી ટ્રીક

"ARABIA CS" (અંકગણિત, રિલેશનલ, એસાઇનમેન્ટ, બિટવાઈજ, ઇન્ક્રિમેન્ટ, એસાઇનમેન્ટ, કન્ડિશનલ, સ્પેશિયલ)

પ્રશ્ન 2 [b ગુણ]

4 ઉદાહરણ સાથે રિલેશનલ ઓપરેટર અને ઇન્ક્રિમેન્ટ/ડિક્રિમેન્ટ ઓપરેટર સમજાવો.

| જવાબ | | | | |
|--------------|--|-------|--|--------------|
| જવાબ: | ઓપરેટર પ્રકાર | વર્ણન | ઉદાહરણ | આઉટપુટ |
| રિલેશનલ | બે મૂલ્યોની વરચેના સંબંધની તપાસ કરે છે | | int a = 5, b = 10; printf("%d", a < b); | 1 (સાચું) |
| | સમાન (==) | | printf("%d", 5 == 5); | 1 (સાચું) |
| | અસમાન (!=) | | printf("%d", 5 != 10); | 1 (સાચું) |
| | મોટું/નાનું | | printf("%d %d", 5 > 3, 5 < 3); | 1 0 |
| ઇન્ક્રિમેન્ટ | મૂલ્યમાં 1 વધારો કરે છે. પ્રી-ઇન્ક્રિમેન્ટ (++x): પહેલા વધારો પછી ઉપયોગ. પારસ્ટ-ઇન્ક્રિમેન્ટ (x++): પહેલા ઉપયોગ પછી વધારો | | int x = 5; printf("%d ", ++x); printf("%d", x); | 6 6 |
| ડિક્રિમેન્ટ | મૂલ્યમાં 1 ઘટાડો કરે છે. પ્રી-ડિક્રિમેન્ટ (--x): પહેલા ઘટાડો પછી ઉપયોગ. પારસ્ટ-ડિક્રિમેન્ટ (x--): પહેલા ઉપયોગ પછી ઘટાડો | | int y = 5; printf("%d ", y--); printf("%d", y); | 5 4 |

• રિલેશનલ ઓપરેટર્સ: 1 (સાચું) અથવા 0 (ખોટું) પરત કરે છે
 • ઇન્ક્રિમેન્ટ/ડિક્રિમેન્ટ: વેરિએબલ મૂલ્ય બદલે છે અને મૂલ્ય પરત કરે છે

મેમરી ટ્રીક

"રિલેશનલ કહે સાચું કે ખોટું, ઇન્ક્રિમેન્ટ/ડિક્રિમેન્ટ કરે ચઢાવ કે ઉતાર"

પ્રશ્ન 2 [c ગુણ]

7 1 થી 100 નો સરવાળો અને એવરેજ પ્રિન્ટ કરવા માટેનો પ્રોગ્રામ લખો.

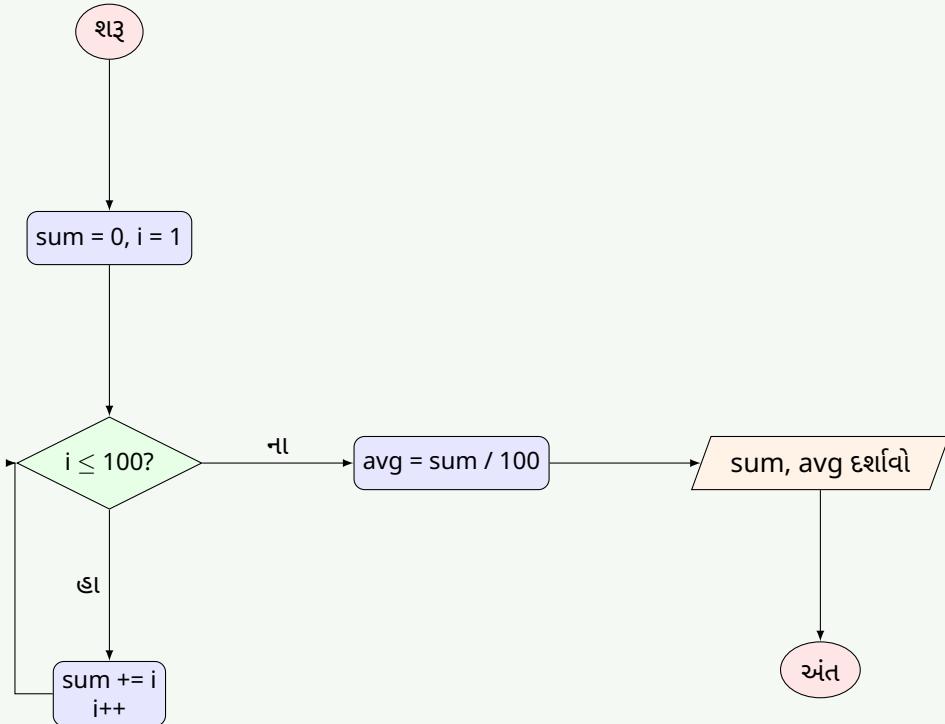
| જવાબ | |
|---|--|
| જવાબ: | |
| <pre> 1 #include <stdio.h> 2 3 int main() { 4 int i, sum = 0; 5 float average; 6 7 // 1 થી 100 સુધીની સંખ્યાઓનો સરવાળો ગણો 8 for(i = 1; i <= 100; i++) { 9 sum += i; 10 } 11 12 // એવરેજ ગણો 13 average = (float)sum / 100; 14 15 // પરણિામો દર્શાવો 16 printf("1 થી 100 સુધીની સંખ્યાઓનો સરવાળો = %d\n", sum); </pre> | |

```

17 printf("1 થી 100 સુધીની સંખ્યાઓની સરેરાશ = %.2f\n", average);
18
19 return 0;
20 }

```

આકૃતિ:



- લૂપ કાઉન્ટર: વેરિએબલ *i* 1 થી 100 સુધીની સંખ્યાઓ ટ્રેક કરે છે
- સરવાળાની ગણતરી: *sum* વેરિએબલમાં મૂલ્યો એકત્રિત કરે છે
- ટાઇપ કાસ્ટિંગ: (float) સરવાળાને ચોક્કસ ભાગાકાર માટે ફ્લોટિંગ-પોઇન્ટમાં કન્વર્ટ કરે છે

મેમરી ટ્રીક

"એક થી સો સરવાળો, પછી ભાગવાથી એવરેજ"

OR

પ્રશ્ન 2 [ગુજરાતી]

3 gets(S) અને scanf("%s",S) ફંક્શન વચ્ચેનો તફાવત લખો જ્યાં S સ્ટ્રિંગ છે.

જવાબ

જવાબ:

| લક્ષ્યાણ | gets(S) | scanf("%s",S) |
|----------------------|--------------------------------------|---|
| ઇનપુટ સમાપ્તિ | ન્યૂલાઇન કેરેક્ટર (\n) સુધી વાંચે છે | બહાઈટસ્પેસ (સ્પેસ, ટેબ, ન્યૂલાઇન) સુધી વાંચે છે |
| બહાઈટસ્પેસ હેન્ડલિંગ | સ્પેસ સાથેની સ્ટ્રિંગ વાંચી શકે છે | પ્રથમ બહાઈટસ્પેસ પર વાંચવાનું બંધ કરે છે |
| બફર ઓવરફલો | બાઉન્ડ્સ ચેકિંગ નથી (અસુરક્ષિત) | બાઉન્ડ્સ ચેકિંગ નથી (અસુરક્ષિત) |
| રિટર્ન વેલ્યુ | સફળતા પર S, ભૂલ પર NULL પરત કરે છે | સફળતાપૂર્વક વાંચેલી આઇટમ્સની સંખ્યા પરત કરે છે |
| રિલેસમેન્ટ | fgets() વધુ સુરક્ષિત વિકલ્પ છે | વિઝિલિમિટ સાથે scanf("%ns",S) વધુ સુરક્ષિત છે |

- સુરક્ષા ચિંતા: બને ફંક્શન બફર ઓવરફલો કરી શકે છે
- વ્યવહારિક ઉપયોગ: gets() પૂર્ણ લાઇન્સ માટે, scanf() એકલ શબ્દો માટે

મેમરી ટ્રીક

"gets મેળવે બધું ન્યૂલાઇન સુધી, scanf અટકે સફેદી જોતાં જ"

OR

પ્રશ્ન 2 [b ગુણ]

4 ઉદાહરણ સાથે લોજિકલ ઓપરેટર અને એસાઇનમેન્ટ ઓપરેટર સમજાવો.

જવાબ

જવાબ:

| ઓપરેટર પ્રકાર | વર્ણન | ઉદાહરણ | આઉટપુટ |
|---------------|--------------------------------|------------------------------------|-----------|
| લોજિકલ | શરતો પર લોજિકલ ઓપરેશન્સ કરે છે | int a = 5, b = 10; | |
| | લોજિકલ AND (&&) | printf("%d", (a > 0) && (b > 0)); | 1 (સાચું) |
| | લોજિકલ OR () | printf("%d", (a > 10) (b > 5)); | 1 (સાચું) |
| | લોજિકલ NOT (!) | printf("%d", !(a == b)); | 1 (સાચું) |
| એસાઇનમેન્ટ | વેરિએબલ્સને મૂલ્યો આપે છે | int x = 10; | x = 10 |
| | સિમ્પલ એસાઇનમેન્ટ (=) | x = 20; | x = 20 |
| | એડ અને એસાઇન (+=) | x += 5; | x = 25 |
| | સબટ્રેક્ટ અને એસાઇન (-=) | x -= 10; | x = 15 |
| | મલિટિપ્લાય અને એસાઇન (*=) | x *= 2; | x = 30 |
| | ડિવાઇડ અને એસાઇન (/=) | x /= 3; | x = 10 |

- લોજિકલ ઓપરેટર્સ: નિષ્ણય લેવામાં ઉપયોગ થાય છે
- શોર્ટ-સર્કિટ ઇવેલ્યુઅશન: && અને || જરૂરી હોય એટલું જ મૂલ્યાંકન કરે છે
- કંપાઉન્ડ એસાઇનમેન્ટ: ઓપરેશન અને એસાઇનમેન્ટ જોડે છે

મેમરી ટ્રીક

"AND માંગે બધા સાચા, OR માંગે એક; એસાઇનમેન્ટ લે જમણું, ડાબે મૂકે એક"

OR

પ્રશ્ન 2 [c ગુણ]

7 આપેલ બે ફલોટિંગ પોર્ટન નંબરો વચ્ચેના તમામ પૂર્ણાંકોને પ્રિન્ટ કરવા માટેનો પ્રોગ્રામ લખો.

જવાબ

જવાબ:

```

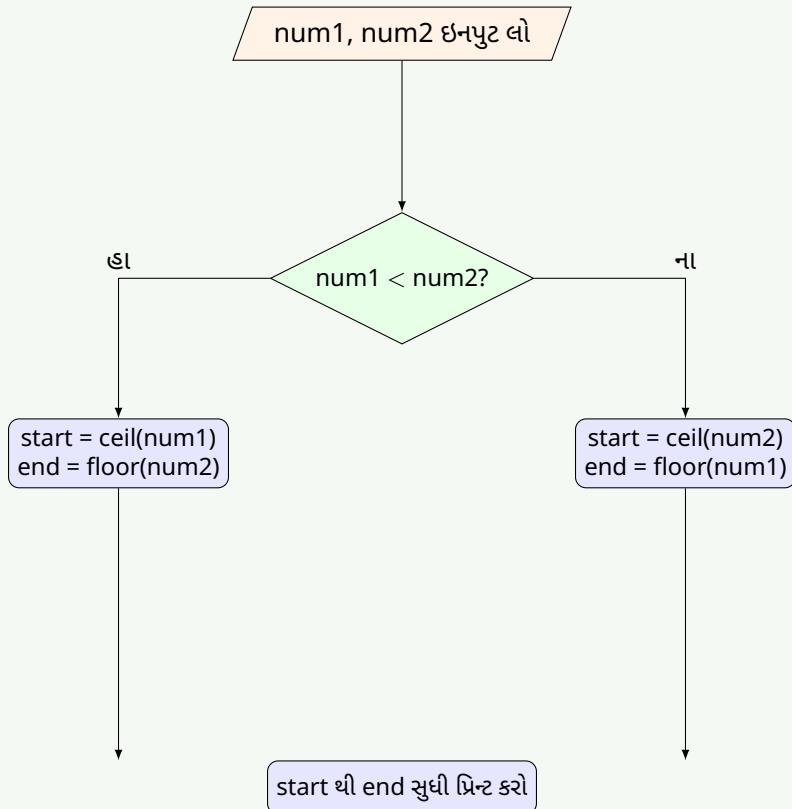
1 #include <stdio.h>
2 #include <math.h>
3
4 int main() {
5     float num1, num2;
6     int start, end, i;

```

```

7 // બે ફ્લોટિંગ પોઇન્ટ નંબર ઇનપુટ લો
8 printf("પ્રથમ ફ્લોટિંગ પોઇન્ટ નંબર દાખલ કરો: ");
9 scanf("%f", &num1);
10
11 printf("બીજો ફ્લોટિંગ પોઇન્ટ નંબર દાખલ કરો: ");
12 scanf("%f", &num2);
13
14 // નાની સંખ્યાનો સીલિંગ અને મોટી સંખ્યાનો ફ્લોર શોધો
15 if(num1 < num2) {
16     start = ceil(num1);
17     end = floor(num2);
18 } else {
19     start = ceil(num2);
20     end = floor(num1);
21 }
22
23 // બે સંખ્યાઓ વચ્ચેના તમામ પૂર્ણાંકો પ્રણિટ કરો
24 printf("%.2f અને %.2f વચ્ચેના પૂર્ણાંકો છે:\n", num1, num2);
25 for(i = start; i <= end; i++) {
26     printf("%d ", i);
27 }
28 printf("\n");
29
30 return 0;
31 }
32 }
```

આકૃતિ:



- મેથ ફંક્શન્સ: ceil() ઉપર રાઉન્ડ કરે છે, floor() નીચે રાઉન્ડ કરે છે
- રેન્જ નિર્ધારણ: ઇનપુટ ઓર્ડરથી સ્વતંત્ર કામ કરે છે
- ઇન્ટીજર એક્સટ્રેક્શન: ફ્લોટસ વચ્ચેના ફક્ત પૂર્ણાંકો પ્રિન્ટ કરે છે

પ્રેમરી ટ્રીક

"નાનાને છત બનાવો, મોટાને ભોયતળિયું, પછી પ્રિન્ટ કરો વર્ચ્યેના બધા પૂણીઓ!"

પ્રક્રિયા 3 [ગુણ]

3 ઉદાહરણ સાથે multiple if-else સ્ટેટમેન્ટ સમજાવો.

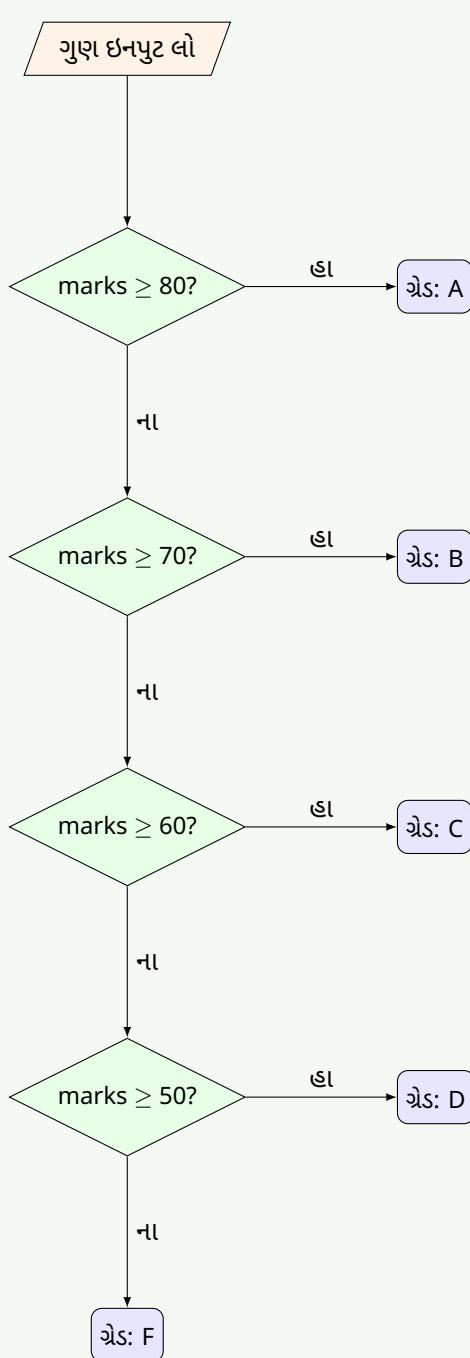
જવાબ

જવાબ: Multiple if-else સ્ટેટમેન્ટ્સ ક્રમશ: અનેક શરતોની તપાસ કરવા માટે વપરાય છે, જેમાં દરેક શરત માત્ર ત્યારે જ ચકાસવામાં આવે છે જ્યારે અગાઉની શરતો ખોટી હોય.

```

1 #include <stdio.h>
2
3 int main() {
4     int marks;
5
6     printf("ગુણ દાખલ કરો (0-100): ");
7     scanf("%d", &marks);
8
9     if(marks >= 80) {
10         printf("ગ્રેડ: A\n");
11     } else if(marks >= 70) {
12         printf("ગ્રેડ: B\n");
13     } else if(marks >= 60) {
14         printf("ગ્રેડ: C\n");
15     } else if(marks >= 50) {
16         printf("ગ્રેડ: D\n");
17     } else {
18         printf("ગ્રેડ: F\n");
19     }
20
21     return 0;
22 }
```

આફ્ટરિટી:



- કભિક પરીક્ષાણ: ફકત એક બ્લોક જ એક્ઝિક્યુટ થાય છે
- કાર્યક્ષમતા: સારી શરત મળ્યા પછી તપાસ બંધ થઈ જાય છે

મેમરી ટ્રીક

"જો આ તો એ, નહીં તો જો પેલું તો એમ, નહીં તો જો અન્ય તો અલગ"

પ્રશ્ન 3 [b ગુણ]

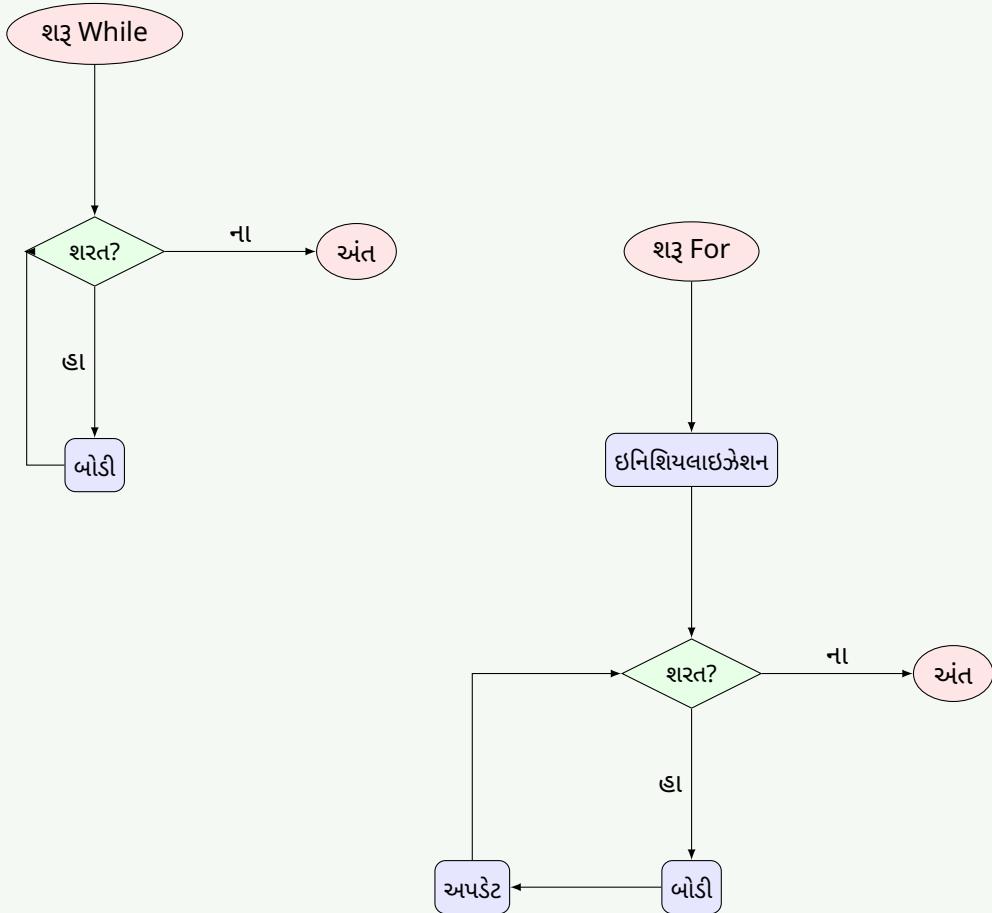
4 While લૂપ અને for લૂપની કામગીરી જણાવો.

જવાબ

જવાબ:

| લૂપ પ્ર- કાર | કામગીરી | સિન્ટેક્સ | ઉપયોગ કેસ |
|--------------------|---|--|---|
| while લૂપ | 1. શરત ચકાસો 2. જો સાચી હોય તો બોડી એક્ઝિક્યુટ કરો 3. શરત ખોટી થાય ત્યાં સુધી 1-2 પગલાં પુનરાવર્તિત કરો | while(condition) { // સ્ટેટમેન્ટ્સ } | જ્યારે પુનરાવર્તનની સંખ્યા અગાઉથી ખબર ન હોય |
| for લૂપ | 1. ઇનિશિયલાઇઝેશન એક વખત એક્ઝિક્યુટ કરો 2. શરત ચકાસો 3. જો સાચી હોય તો બોડી એક્ઝિક્યુટ કરો 4. અપડેટ સ્ટેટમેન્ટ એક્ઝિક્યુટ કરો 5. શરત ખોટી થાય ત્યાં સુધી 2-4 પગલાં પુનરાવર્તિત કરો | for(init; cond; update) { // સ્ટેટમેન્ટ્સ } | જ્યારે પુનરાવર્તનની સંખ્યા અગાઉથી ખબર હોય |

તુલના:



- એન્ટ્રી કંટ્રોલ: બને એક્ઝિક્યુશન પહેલાં શરત ચકાસે છે
- ઘટકો: for લૂપ ઇનિશિયલાઇઝેશન, શરત અને અપડેટ જોડે છે

મેમરી ટ્રીક

"WHILE તપાસે પછી કરે, FOR શરૂ કરે તપાસે કરે અપડેટ કરે"

પ્રક્રિયા 3 [C ગુણ]

7 આપેલ સંખ્યાના ફેક્ટોરિયલ શોધવા માટેનો પ્રોગ્રામ લખો.

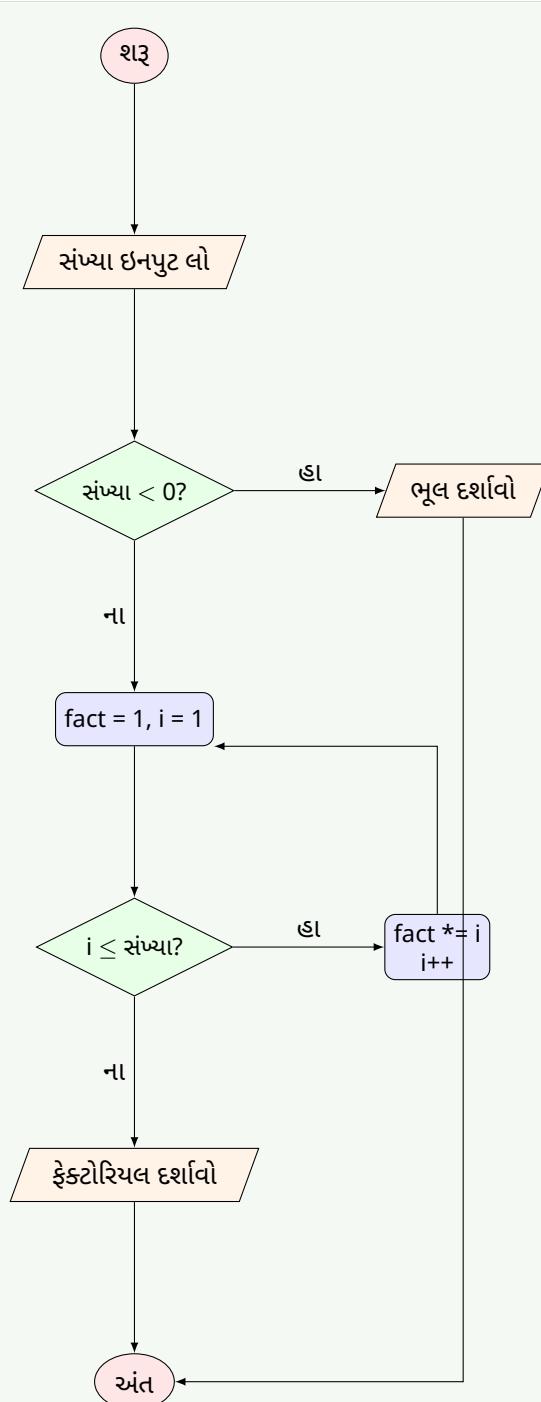
જવાબ

જવાબ:

```

1 #include <stdio.h>
2
3 int main() {
4     int num, i;
5     unsigned long long factorial = 1;
6
7     // એક સંખ્યા ઇનપુટ લો
8     printf("ધ્નાત્મક પૂર્ણાંક દાખલ કરો: ");
9     scanf("%d", &num);
10
11    // તપાસો કે સંખ્યા નકારાત્મક તો નથી
12    if(num < 0) {
13        printf("ભૂલ: નકારાત્મક સંખ્યાઓનું ફેક્ટોરિયલ વ્યાખ્યાયિત નથી.\n");
14    } else {
15        // ફેક્ટોરિયલની ગણતરી કરો
16        for(i = 1; i <= num; i++) {
17            factorial *= i;
18        }
19
20        printf("%d નું ફેક્ટોરિયલ = %llu\n", num, factorial);
21    }
22
23    return 0;
24 }
```

આકૃતિ:



- ડેટા ટાઇપ: મોટા ફેક્ટોરીયલ માટે `unsigned long long`
- ભૂલ હેન્ડલિંગ: નકારાત્મક ઇનપુટ માટે ચકાસણી
- લૂપ અમલીકરણ: કભિક પૂર્ણકોનો ગુણાકાર

મેમરી ટ્રીક

"ફક્ટોરીયલ ફોર્મ્યુલા: એકથી સંખ્યા સુધી ગુણાકાર"

OR

પ્રશ્ન 3 [વ ગુણ]

3 ઉદાહરણ સાથે switch-case સ્ટેટમેન્ટની કામગીરી સમજાવો.

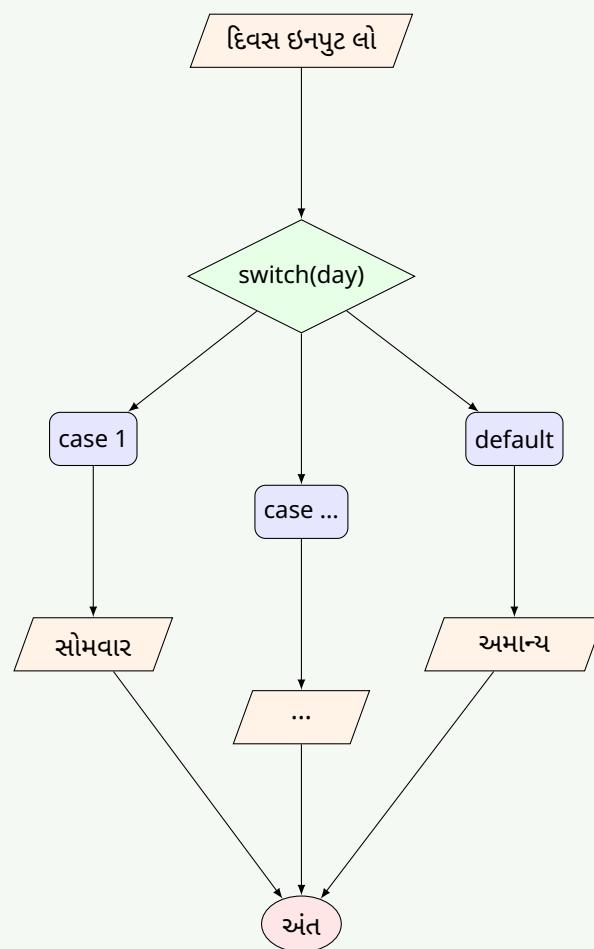
જવાબ

જવાબ: Switch-case સ્ટેટમેન્ટ એ એક મલ્ટી-વે ડિસીજન મેકર છે જે અભિવ્યક્તિના મૂલ્યને વિવિધ કેસ મૂલ્યો સામે તપાસે છે અને મેચ થતા કેસ બલોકને એક્ઝિક્યુટ કરે છે.

```

1 #include <stdio.h>
2
3 int main() {
4     int day;
5
6     printf("દિવિસનો નંબર દાખલ કરો (1-7): ");
7     scanf("%d", &day);
8
9     switch(day) {
10     case 1: printf("સોમવાર\n"); break;
11     case 2: printf("મંગળવાર\n"); break;
12     case 3: printf("બુધવાર\n"); break;
13     case 4: printf("ગુરુવાર\n"); break;
14     case 5: printf("શુક્રવાર\n"); break;
15     case 6: printf("શનિવાર\n"); break;
16     case 7: printf("ચંદ્રવાર\n"); break;
17     default: printf("અમાન્દ્ય દિવિસ નંબર\n");
18 }
19
20     return 0;
21 }
```

આકૃતિ:



- અભિવ્યક્તિ મૂલ્યાંકન: ફક્ત ઇન્ટીજર અથવા કેરેક્ટર ટાઇપ્સ
- કેસ મેચિંગ: break સુધી મેચીંગ કેસ એક્ઝિક્યુટ કરે છે
- ડિફોલ્ટ કેસ: કોઈ કેસ મેચ ન થાય ત્યારે એક્ઝિક્યુટ થાય છે

મેમરી ટ્રીક

"SWITCH મૂલ્ય, CASE મેળ, BREAK બહાર, DEFAULT બચાવ"

OR

પ્રશ્ન 3 [b ગુણ]

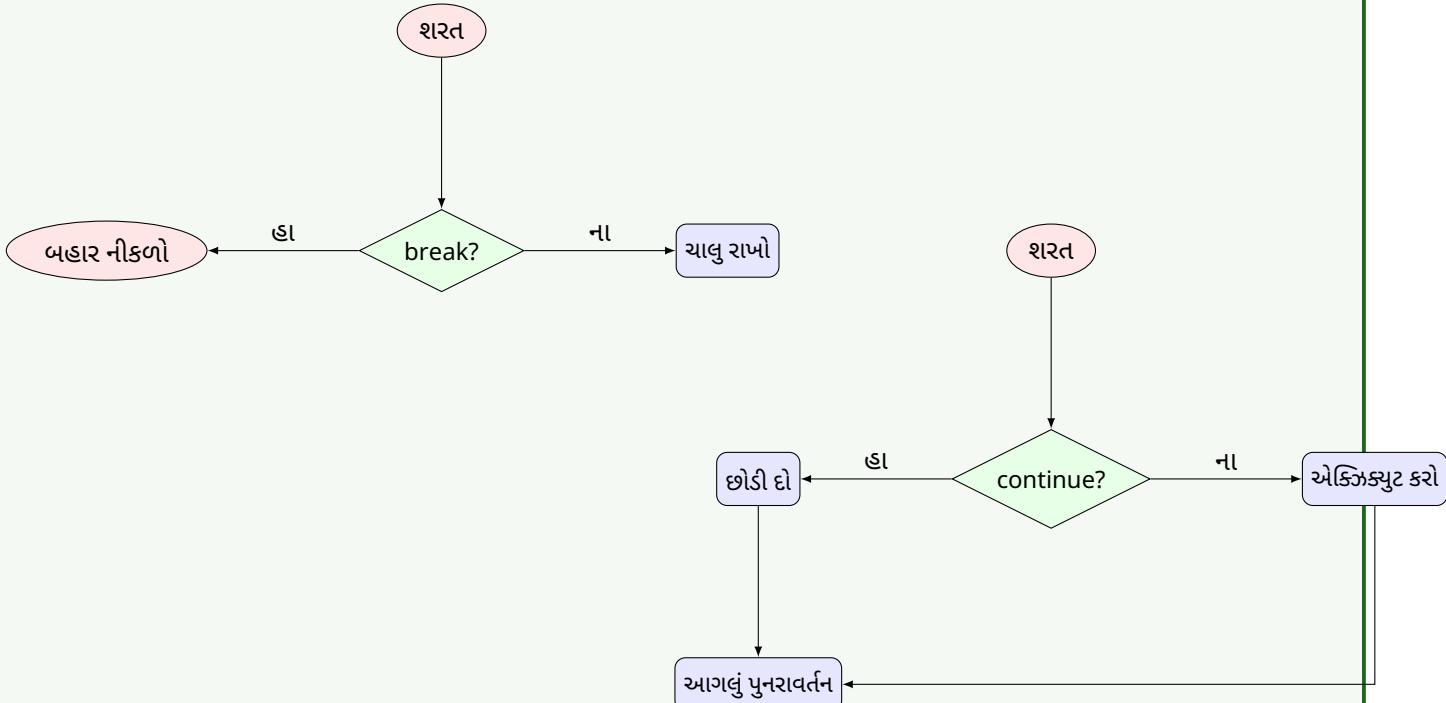
4 break અને continue કીવર્ડ વ્યાખ્યાયિત કરો.

જવાબ

જવાબ:

| કીવર્ડ | વ્યાખ્યા | હેતુ | ઉદાહરણ |
|----------|--|---|--|
| break | સૌથી અંદરના લૂપ અથવા switch સ્ટેટમેન્ટને તરત જ સમાપ્ત કરે છે | જ્યારે કોઈ ચોક્કસ શરત પૂરી થાય ત્યારે લૂપમાંથી બહાર નીકળવા માટે | for(i=1; i<=10; i++) { if(i == 5) break; printf("%d", i); } // 1 2 3 4 |
| continue | પ્રકૃતિના વર્તમાન પુનરાવર્તનના બાકીના ભાગને છોડીને લૂપના આગલા પુનરાવર્તન પર જાય છે | લૂપને સમાપ્ત કર્યા વિના ચોક્કસ પુનરાવર્તનો છોડવા માટે | for(i=1; i<=10; i++) { if(i == 5) continue; printf("%d", i); } // 1 2 3 4 6 7 |

વર્તન તુલના:



- સ્કોપ: બંને માત્ર સૌથી અંદરના લૂપને અસર કરે છે
- કંટ્રોલ ટ્રાન્સફર: break લૂપમાંથી બહાર નીકળે છે, continue આગલા પુનરાવર્તન પર જાય છે

મેમરી ટ્રીક

"BREAK રૂમ છોડે છે, CONTINUE આગલી ડાન્સ મૂવ પર જાય છે"

OR

પ્રશ્ન 3 [C ગુણ]

7 કીવર્ડ પરથી લાઈન(n) ની સંખ્યા વાંચી અને નીચે દર્શાવેલ ટ્રાઇંગલ પ્રિન્ટ કરવા માટેનો પ્રોગ્રામ લાખો.

```

1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
  
```

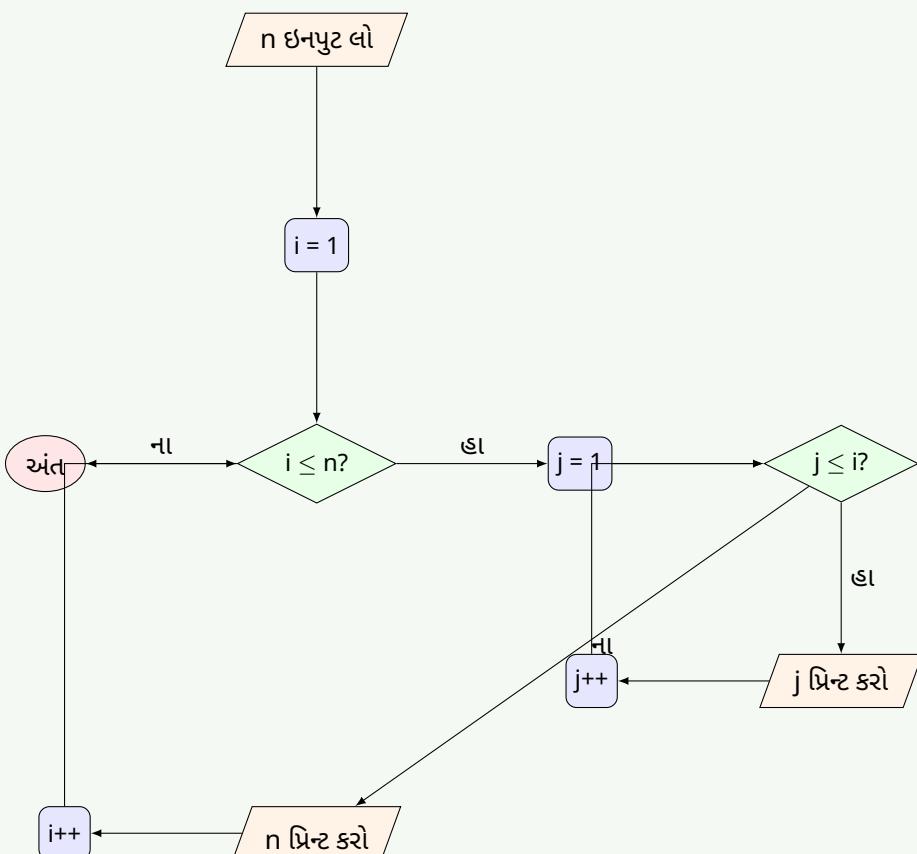
જવાબ

જવાબ:

```

1 #include <stdio.h>
2
3 int main() {
4     int n, i, j;
5
6     // લાઈનની સંખ્યા ઇનપુટ લો
7     printf("લાઈનની સંખ્યા દાખલ કરો: ");
8     scanf("%d", &n);
9
10    // ટ્રાયેગલ પેટરન પ્રનિઃઠ કરો
11    for(i = 1; i <= n; i++) {
12        // દરેક રોમાં 1 થી i સુધીની સંખ્યાઓ પ્રનિઃઠ કરો
13        for(j = 1; j <= i; j++) {
14            printf("%d ", j);
15        }
16        printf("\n");
17    }
18
19    return 0;
20 }
```

પ્રોગ્રામ પ્રવાહ:



- નેસ્ટેડ લૂપ્સ: આઉટર લૂપ રો માટે, ઇનર લૂપ કોલમ માટે
- પેટર્ન લીઝિક: રો નંબર નક્કી કરે છે કે કેટલી સંખ્યાઓ પ્રિન્ટ કરવી
- સંખ્યા ક્રમ: દરેક રો 1 થી રો નંબર સુધી પ્રિન્ટ કરે છે

પેમરી ટ્રીક

"રો નક્કી કરે મર્યાદા, કોલમ પ્રિન્ટ કરે એકથી રો સુધી"

પ્રક્રિયા 4 [a ગુણ]

3 Nested if-else સ્ટેપેન્ટને ઉદાહરણ સાથે સમજાવો.

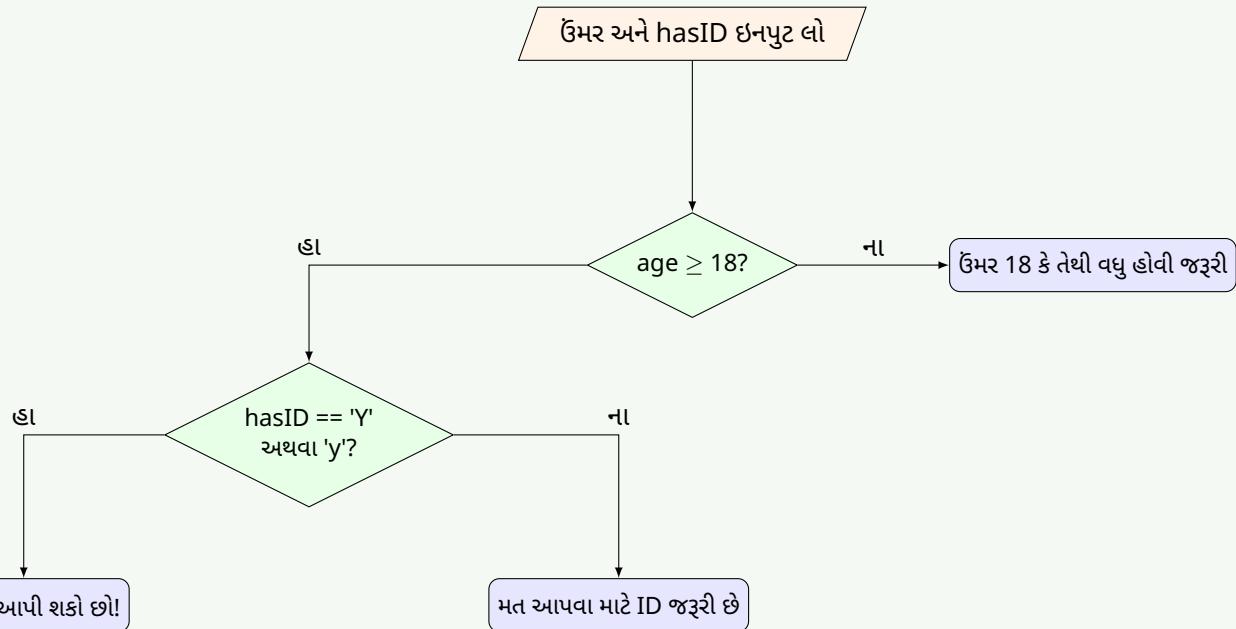
જવાબ

જવાબ: Nested if-else સ્ટેપેન્ટને અને નિર્ણય લેવાના બહુવિધ સ્તરોની મંજૂરી આપે છે.

```

1 #include <stdio.h>
2
3 int main() {
4     int age;
5     char hasID;
6
7     printf("ઉંમર દાખલ કરો: ");
8     scanf("%d", &age);
9
10    printf("શું તમારી પાસે ID છે? (Y/N): ");
11    scanf(" %c", &hasID);
12
13    if(age >= 18) {
14        if(hasID == 'Y' || hasID == 'y') {
15            printf("તમે મત આપ્યો શકો છો!\n");
16        } else {
17            printf("મત આપવા માટે ID જરૂરી છે.\n");
18        }
19    } else {
20        printf("મત આપવા માટે તમારી ઉંમર 18 કે તેથી વધુ હોવી જોઈએ.\n");
21    }
22
23    return 0;
24 }
```

નિર્ણય વૃક્ષ:



- હાયરાર્ડિકલ શરતો: શરતોનું સ્તરમાં મૂલ્યાંકન કરે છે
- ઇન્ડેટેશન: નેસ્ટેડ સ્ટ્રક્ચર્સની વાંચનક્ષમતા સુધારે છે
- મલ્ટી-ફેક્ટર નિષ્યાઓ: એકાધિક માપદંડો જોડે છે

મેમરી ટ્રીક

"If ની અંદર if, તોડી શરતો ચકાસો"

પ્રશ્ન 4 [b ગુણ]

4 One-dimensional array ના initialization નું વર્ણન કરો.

જવાબ

જવાબ:

| ઇનિશિયલાઇઝેશન પદ્ધતિ | સિન્ટેક્સ | ઉદાહરણ | વર્ણન |
|------------------------------|---------------------------------|---------------------------------|---|
| સાઈઝ સાથે ડેક્લરેશન | datatype arrayname[size]; | int marks[5]; | નિર્દિષ્ટ સાઈઝ સાથે એરે બનાવે છે, એલિમેન્ટ્સમાં ગાર્બેજ વેલ્યુ હોય છે |
| ઇનિશિયલાઇઝેશન સાથે ડેક્લરેશન | datatype arrayname[s] = (v); | int ages[4] = {21, 19, 25, 32}; | ચોક્કસ મૂલ્યો સાથે એરે બનાવે અને ઇનિશિયલાઇઝ કરે છે |
| આંશિક ઇનિશિયલાઇઝેશન | datatype arrayname[S] = (v); | int nums[5] = (1, 2); | પ્રથમ એલિમેન્ટ્સ ઇનિશિયલાઇઝ કરે છે, બાકીના શૂચ થાય છે |
| સાઈઝ ઇન્ફરન્સ | datatype arrayname[] = (v); | int scores[] = (95, 88); | ઇનિશિયલાઇઝરની સંચાચ દ્વારા સાઈઝ નક્કી થાય છે |
| વ્યક્તિગત એલિમેન્ટ | arrayname[index] = value; | marks[0] = 85; | ચોક્કસ એલિમેન્ટને મૂલ્ય આપે છે |

એરે વિઝ્યુલાઇઝેશન:

| | | | | | |
|--------|-----|-----|-----|-----|-----|
| Index: | 10 | 20 | 30 | 40 | 50 |
| | [0] | [1] | [2] | [3] | [4] |

- જીરો-ઇન્ડેક્સિંગ: પ્રથમ એલિમેન્ટ ઇન્ડેક્સ 0 પર
- કન્ટિગ્યુઅસ મેમરી: એલિમેન્ટ્સ કમશ: સ્ટોર થાય છે
- સાઇજ લિમિટેશન: સાઇજ કંપાઇલ ટાઇમે જાણીતી હોવી જરૂરી છે

મેમરી ટ્રીક

"પહેલા સાઇજ જાહેર કરો, પછી મૂલ્યો ભરો અથવા કંપાઇલરને ગણવા દો"

પ્રક્રિયા 4 [C ગુણ]

7 અરેને વ્યાખ્યાયિત કરો અને રિઝિંગને રિવર્સ કરવા માટેનો પ્રોગ્રામ લખો.

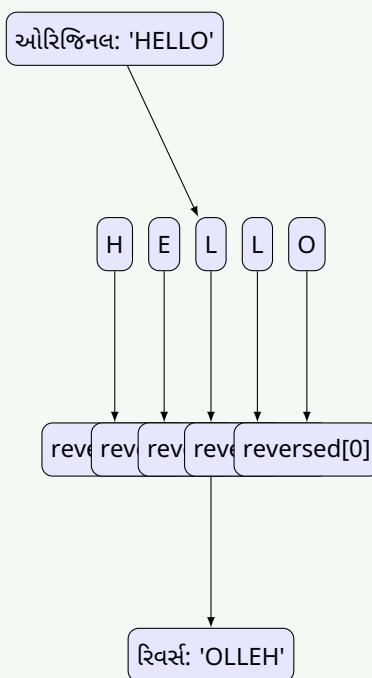
જવાબ

જવાબ: એરે એ સમાન ડેટા આઇટમ્સનો સંગ્રહ છે જે સંબંધ મેમરી સ્થાનો પર સંગ્રહિત થયેલા હોય છે અને એક સામાન્ય નામનો ઉપયોગ કરીને એકસેસ કરવામાં આવે છે.

```

1 #include <stdio.h>
2 #include <string.h>
3
4 int main() {
5     char str[100], reversed[100];
6     int i, j, length;
7
8     // સૂટરાંગ ઇનપુટ લો
9     printf("એક સૂટરાંગ દાખલ કરો: ");
10    gets(str);
11
12    // સૂટરાંગની લંબાઈ શોધો
13    length = strlen(str);
14
15    // સૂટરાંગને રવિરસ કરો
16    for(i = length - 1, j = 0; i >= 0; i--, j++) {
17        reversed[j] = str[i];
18    }
19
20    // NULL ટ્રમનિટર ઉમેરો
21    reversed[j] = '\0';
22
23    // રવિરસ કરેલી સૂટરાંગ દર્શાવો
24    printf("રવિરસ કરેલી સૂટરાંગ: %s\n", reversed);
25
26    return 0;
27 }
```

એટ્યોરિધમ વિઝ્યુલાઇઝનન:



- ક્રેક્ટર એરે: NULL ટમ્પિનેટર સાથે સિંગ્લ સ્ટોર કરે છે
- ટુ-પોઇન્ટર ટેકનિક: એક ઓરિજિનલ માટે, એક રિવર્સ માટે
- ઝીરો-બેઝ ઇન્ડેક્સિંગ: એરે ઇન્ડેક્સ 0 થી શરૂ થાય છે

પ્રેમરી ટ્રીક

"અંતથી શરૂ કરો, શરૂઆતમાં મૂકો, શૂન્ય પર અટકો"

OR

પ્રશ્ન 4 [a ગુણ]

3 do while loop ઉદાહરણ સાથે સમજાવો

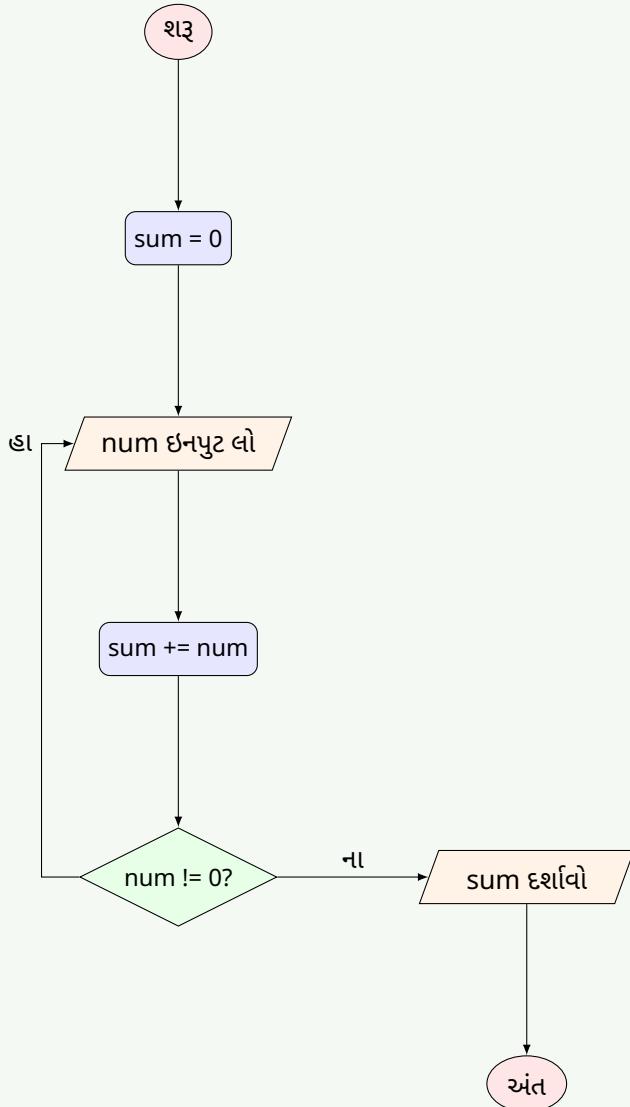
જવાબ

જવાબ: do-while લૂપ એ એક એક્ઝિટ-ક્રોલ લૂપ છે જે શરત ચકાસ્યા પહેલાં ઓછામાં ઓછી એક વખત લૂપ બોડી એક્ઝિક્યુટ કરે છે.

```

1 #include <stdio.h>
2
3 int main() {
4     int num, sum = 0;
5
6     do {
7         printf("એક સંખ્યા દાખલ કરો (0 રોકવા માટે): ");
8         scanf("%d", &num);
9         sum += num;
10    } while(num != 0);
11
12    printf("બધી દાખલ કરેલી સંખ્યાઓનો સરવાળો: %d\n", sum);
13
14    return 0;
15 }
  
```

લૂપ એક્ઝિક્યુશન ફ્લો:



- એક્ઝિક્યુશન ઓર્ડર: પહેલા બોડી, પછી શરત ચકાસણી
- ગેસ્ટર્ડ એક્ઝિક્યુશન: લૂપ બોડી હમેશા ઓછામાં ઓછી એક વખત એક્ઝિક્યુટ થાય છે
- ટર્મિનેશન: શરત લૂપના તળિયે મૂલ્યાંકિત થાય છે

મેમરી ટ્રીક

"પહેલા કરો, પછી પૂછો"

OR

પ્રશ્ન 4 [b ગુણ]

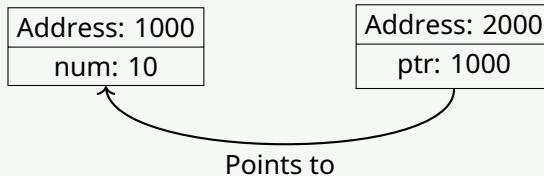
4 પોઇન્ટ્રને વ્યાખ્યાયિત કરો અને ઉદાહરણ સાથે પોઇન્ટ્રનું વર્ણન કરો.

જવાબ

જવાબ: પોઇન્ટર એક એવું વેરિયેબલ છે જે અન્ય વેરિયેબલનું મેમરી એડ્રેસ સ્ટોર કરે છે.

| પોઇન્ટર કોન્સ્ટ | વર્ણન | ઉદાહરણ |
|-----------------|----------------------------------|----------------------------|
| ડેક્લરેશન | Data_type *pointer_name; | int *ptr; |
| ઇનિશિયલાઇઝેશન | વેરિયેબલનું એડ્રેસ એસાઇન કરવું | int n = 10; int *ptr = &n; |
| ડિરેક્સ | એડ્રેસ પરના મૂલ્યને એક્સેસ કરવું | *ptr (10 આપે છે) |
| એડ્રેસ ઓપરેટર | વેરિયેબલનું એડ્રેસ મેળવે છે | &n |
| NULL પોઇન્ટર | કશું પોઇન્ટ ન કરતાં પોઇન્ટર | int *ptr = NULL; |

પોઇન્ટર વિનિયુલાઇઝેશન:



- ઇનડાયરેક્ટ એક્સેસ: તેમના એડ્રેસ દ્વારા વેરિયેબલસ એક્સેસ કરે છે
- મેમરી મેનિયુલેશન: કાર્યક્ષમતા માટે ડાયરેક્ટ મેમરી એક્સેસ
- ડાયનામિક મેમરી: રનાઇમ દરમિયાન એલોકેશન/ડીએલોકેશન સક્ષમ કરે છે

મેમરી ટ્રીક

"પોઇન્ટર્સ એડ્રેસને પોઇન્ટ કરે છે, સ્ટાર્સ મૂલ્યોને ડિરેક્સ કરે છે"

OR

પ્રશ્ન 4 [C ગુણ]

7 પોઇન્ટર વ્યાખ્યાયિત કરો અને પોઇન્ટર આગ્રહીતનો ઉપયોગ કરીને બે પૂર્ણકોની અદલા બદલી કરવા માટેનો પ્રોગ્રામ લખો.

જવાબ

જવાબ: પોઇન્ટર એ એક વેરિયેબલ છે જે અન્ય વેરિયેબલના મેમરી એડ્રેસને ધરાવે છે, જે ડેટાનો પરોક્ષ એક્સેસ અને મેનિયુલેશન કરવાની મંજૂરી આપે છે.

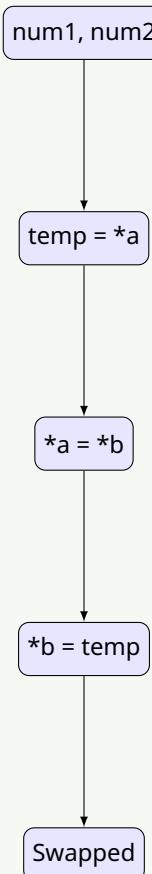
```

1 #include <stdio.h>
2
3 // પોઇન્ટર્સનો ઉપયોગ કરીને બે ઇન્ટીજર્સની અદલાબદલી કરવાનું ફૂદ્ધારણ
4 void swap(int *a, int *b) {
5     int temp = *a;
6     *a = *b;
7     *b = temp;
8 }
9
10 int main() {
11     int num1, num2;
12
13     // બે પૂર્ણાંક ઇનપુટ લો
14     printf("પ્રથમ સંખ્યા દાખલ કરો: ");
15     scanf("%d", &num1);
16
17     printf("દ્વિજી સંખ્યા દાખલ કરો: ");
18     scanf("%d", &num2);
19
  
```

```

20 printf("અદલાબદલી પહેલાં: num1 = %d, num2 = %d\n", num1, num2);
21
22 // num1 અને num2 ના એડ્રેસ સાથે swap ફંક્શન કોલ કરો
23 swap(&num1, &num2);
24
25 printf("અદલાબદલી પછી: num1 = %d, num2 = %d\n", num1, num2);
26
27 return 0;
28 }
```

સ્વેપ પ્રોસેસ વિજ્ઞુલાઇઝનાઃ



- પાસ બાય રેફારન્સ: પોઇન્ટર્સ ફંક્શન્સને મૂળ વેરિએબલ્સ મોડિફાય કરવાની મંજૂરી આપે છે
- ટેમ્પરારી વેરિએબલ: ડેટા નુકસાન વિના સ્વેપ કરવા માટે જરૂરી છે
- ફંક્શન પેરામીટર: પોઇન્ટર આર્ગ્યુમેન્ટ્સ એડ્રેસ પાસ કરે છે

મેમરી ટ્રીક

"એન્ટેસથી પકડો, કન્ટેન બદલો, હાજરી વિના"

પ્રશ્ન 5 [2 ગુણ]

3 50 અને 500 ની વચ્ચે 7 વડે ભાગી શકાય તેવી સંખ્યાઓ શોધવા માટેનો પ્રોગ્રામ લખો.

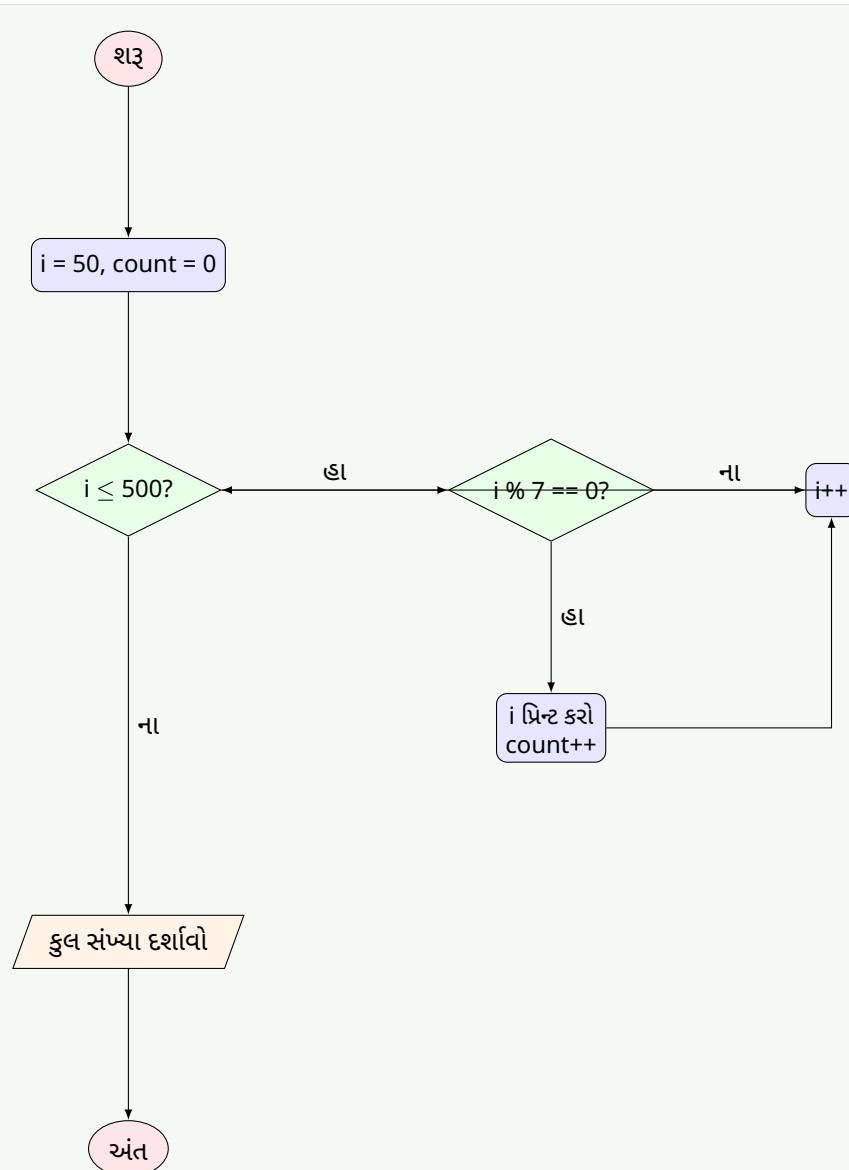
જવાબ

જવાબ:

```

1 #include <stdio.h>
2
3 int main() {
4     int i, count = 0;
5
6     printf("50 અને 500 ની વચ્ચે 7 વડે ભાગી શકાય તેવી સંખ્યાઓ:\n");
7
8     // 7 વડે ભાગી શકાય તેવી સંખ્યાઓ શોધો અને પ્રણિટ કરો
9     for(i = 50; i <= 500; i++) {
10         if(i % 7 == 0) {
11             printf("%d ", i);
12             count++;
13
14             // વધુ સારી વાંચનક્ષમતા માટે દર લાઇને 10 સંખ્યાઓ પ્રણિટ કરો
15             if(count % 10 == 0)
16                 printf("\n");
17         }
18     }
19
20     printf("કુલન સંખ્યા: %d\n", count);
21
22     return 0;
23 }
```

એળોરિધમ વિઝ્યુલાઇઝનાન:



- મોડ્યુલો ઓપરેટર: $i \% 7 == 0$ વિભાજ્યતા ચકાસે છે
- આઉટપુટ ફોર્મેટિંગ: વાંચવા માટે લાઇન બ્રેક
- કાઉન્ટર વેરિએબલ: કેટલી સંખ્યાઓ મળી તે ટ્રેક કરે છે

મેમરી ટ્રીક

"સાતથી ભાગો, શૂન્ય શેષ જીતે"

પ્રશ્ન 5 [b ગુણ]

4 કીબોર્ડ પરથી પૂર્ણક વાંચી આપેલ સંખ્યા એકી છે કે બેકી છે તે પ્રિન્ટ કરવા માટેનો પ્રોગ્રામ લખો.

જવાબ

જવાબ:

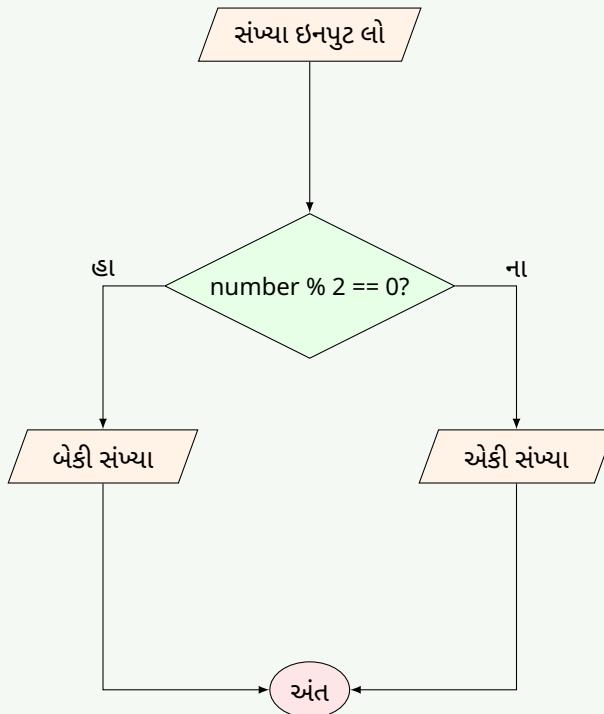
```

1 #include <stdio.h>
2
  
```

```

3 int main() {
4     int number;
5
6     printf("એક પૂર્ણાંક દાખલ કરો: ");
7     scanf("%d", &number);
8
9     if(number % 2 == 0) {
10         printf("%d એક બેકી સંખ્યા છે.\n", number);
11     } else {
12         printf("%d એક અદેખી સંખ્યા છે.\n", number);
13     }
14
15     return 0;
16 }
```

નિર્ણય લોજિક:



- મોડ્યુલો ટેસ્ટ: બેકી સંખ્યાઓને 2 વડે ભાગતાં શેષ 0 આવે છે
- બાઇનરી રીપ્રોક્ટેશન: બેકી સંખ્યાનો અંતિમ બિટ 0 હોય છે, અદેખી સંખ્યાનો 1 હોય છે
- સિંપ્લ એલ્ગોરિદમ: નેગેટિવ સંખ્યાઓ સહિત બધા પૂર્ણાંકો માટે કામ કરે છે

મેમરી ટ્રીક

"બેકી અંતે શુન્ય, અદેખી અંતે એક"

પ્રશ્ન 5 [C ગુણ]

7 સ્ટ્રક્ચર વ્યાખ્યાયિત કરો? સમજાવો કે તે એરેથી કેવી રીતે અલગ છે? પુસ્તકો વિશે નીચેની માહિતી સાચવવા માટે પુસ્તક નામનું સ્ટ્રક્ચર વિકસાવો. પુસ્તકનું શીર્ષક, લેખકનું નામ, કિંમત અને પાનાંની સંખ્યા.

ଜ୍ଵାବ

જવાબ: સ્ક્રુક્યર એ વપરાશકર્તા-વ્યાખ્યાયિત ડેટા ટાઇપ છે જે એક જ નામ હેઠળ વિવિધ ડેટા ટાઇપ્સના વેરિએબલ્સના સમૂહને મંજૂરી આપે છે. સ્ક્રુક્યર અને એકે વચ્ચેનો તફાવત:

| લક્ષણ | સ્ટ્રક્ચર | એરે |
|-----------|--|---|
| ડેટા ટાઇપ | વિવિધ ડેટા ટાઇપ્સ સ્ટોર કરી શકે | સમાન ડેટા ટાઇપના એલિમેન્ટ્સ સંગ્રહે છે |
| એક્સેસ | સભ્યો ડોટ(.) ઓપરેટર દ્વારા એક્સેસ થાય છે | એલિમેન્ટ્સ ઇન્ડેક્સ [] દ્વારા એક્સેસ થાય છે |
| ડેકલેરેશન | struct કીવર્ડનો ઉપયોગ કરે છે | સ્કવર બ્રેકેટ્સ []નો ઉપયોગ કરે છે |
| હેતુ | સંબંધિત વિષમ ડેટાને સંગઠિત કરે છે | સમાન ડેટાને સંગઠિત કરે છે |

બુક સ્ટ્રક્ચર પ્રોગ્રામ:

```
#include <stdio.h>
#include <string.h>

struct Book {
    char title[100];
    char author[50];
    float price;
    int pages;
};

int main() {
    struct Book myBook;

    strcpy(myBook.title, "C प्रोग्रामिंग");
    strcpy(myBook.author, "डेनिस रथी");
    myBook.price = 350.50;
    myBook.pages = 285;

    printf("पुस्तक विवर:\n");
    printf("शीरक: %s\n", myBook.title);
    printf("वेख: %s\n", myBook.author);
    printf("किंमत: %.2f\n", myBook.price);
    printf("पाँच: %d\n", myBook.pages);

    return 0;
}
```

સ્ટકચર વિદ્યુલાઇઝન:

```
struct Book  
{  
    title: "C Programming"  
    author: "Dennis Ritchie"  
    price: 350.50    pages: 285
```

- સ્ટ્રક્ચર વ્યાખ્યા: ડેટા માટે ટેમ્પલેટ બનાવે છે
 - સભ્ય એક્સેસ: ડોટ ઓપરેટર (structure.member) નો ઉપયોગ કરો
 - સ્ટ્રિંગ હન્ડલિંગ: કેરેક્ટર એરે માટે સ્ટ્રિંગ ફંક્શન્સનો ઉપયોગ કરે છે

ਮੇਮਰੀ ਟ੍ਰੈਕ

"સ્ટ્રક્ચર જુદું એકત્ર કરે, એરે એકસરખું રાખે"

OR

પ્રશ્ન 5 [વ ગુણ]

3 કીબોર્ડ પરથી વાસ્તવિક સંખ્યા વાંચી અને તેના કરતા મોટો સૌથી નાનો પૂર્ણાંક પ્રિન્ટ કરવા માટેનો પ્રોગ્રામ લખો.

જવાબ

જવાબ:

```

1 #include <stdio.h>
2 #include <math.h>
3
4 int main() {
5     float number;
6     int result;
7
8     printf("એક વાસ્તવિક સંખ્યા દાખલ કરો: ");
9     scanf("%f", &number);
10
11     result = ceil(number);
12
13     printf("%.2f કરતાં મોટો સૌથી નાનો પૂર્ણાંક %d છે.\n", number, result);
14
15     return 0;
16 }
```

- મેથ ફંક્શન: ceil() આગળના પૂર્ણાંક પર રાઉન્ડ કરે છે
- પરિણામ ટાઇપ: ઇનપુટથી મોટો નાનામાં નાનો પૂર્ણાંક પરત કરે છે

મેમરી ટ્રીક

"CEILING ફંક્શન, ઉપર જઈએ, આગળનો પૂર્ણાંક બતાવીએ"

OR

પ્રશ્ન 5 [બ ગુણ]

4 કીબોર્ડ પરથી અક્ષર વાંચી અને તેની ASCII વેલ્યુ પ્રિન્ટ કરવા માટેનો પ્રોગ્રામ લખો.

જવાબ

જવાબ:

```

1 #include <stdio.h>
2
3 int main() {
4     char ch;
5
6     printf("એક અક્ષર દાખલ કરો: ");
7     scanf("%c", &ch);
8
9     printf("%c ની ASCII વેલ્યુ %d છે.\n", ch, ch);
10
11     return 0;
12 }
```

- કેરેક્ટર સ્ટોરેજ: કેરેક્ટર્સ મેમરીમાં ઇન્ટીજર તરીકે સંગ્રહ થાય છે
- ટાઇપ કન્વર્ટન: char થી int માં ઓટોમેટિક કન્વર્ટન

મેમરી ટ્રીક

"અક્ષરો નીચે સંખ્યાઓ છુપાવે, પ્રિન્ટ બંને બાજુ બતાવે"

OR

પ્રશ્ન 5 [C ગુણ]

7 ફુંક્શન ને વ્યાખ્યાપિત કરો? તેનો ફાયદો સમજાવો. આપેલ પૂર્ણક સંખ્યાના વર્ગની ગણતરી કરવા માટેનું ફુંક્શન લખો.

જવાબ

જવાબ: ફુંક્શન એ કોડનો સેલ્ફ-કન્ટેઇન્ડ બ્લોક છે જે ચોક્સ કાર્ય કરવા માટે ડિઝાઇન કરવામાં આવ્યો છે. તે ઇનપુટ લે છે, તેને પ્રોસેસ કરે છે, અને આઉટપુટ પરત કરે છે.

ફુંક્શનના ફાયદાઓ:

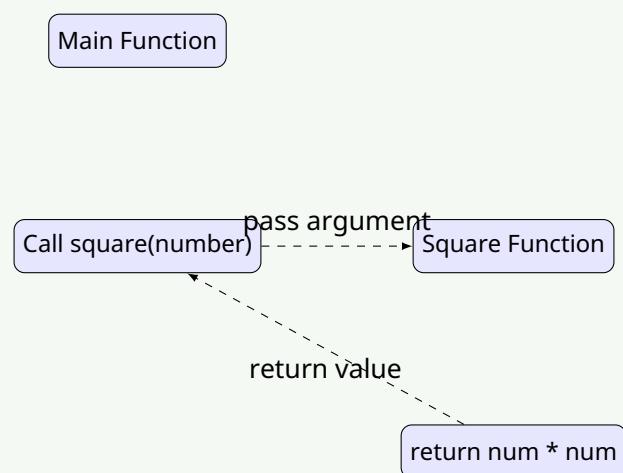
| ફાયદો | વર્ણન |
|-------------------|--|
| કોડ રીયુઝનિભિલિટી | એક વખત લખો, અનેક વખત વાપરો |
| મોડ્યુલારિટી | જટિલ સમર્થ્યાઓને સંચાલિત ભાગોમાં વિભાજિત કરો |
| મેન્ટેનિબિલિટી | ઇસોલેટેડ કોડને ડિબગ અને મોડિફિય કરવું સરળ છે |
| એબ્સ્ટ્રેક્શન | અમલીકરણની વિગતો છુપાવો |

સ્ક્રેન ફુંક્શન સાથે પ્રોગ્રામ:

```

1 #include <stdio.h>
2
3 // પૂર્ણાંકના વર્ગની ગણતરી કરવાનું ફુંક્શન
4 int square(int num) {
5     return num * num;
6 }
7
8 int main() {
9     int number, result;
10
11     printf("એક પૂર્ણાંક દાખલ કરો: ");
12     scanf("%d", &number);
13
14     result = square(number);
15
16     printf("%d નો વર્ગ %d છે\n", number, result);
17
18     return 0;
19 }
```

ફુંક્શન ફ્લો:



- ફંક્શન પ્રોટોટાઇપ: ફંક્શન સિગ્નર જાહેર કરે છે
- પેરામીટર્સ: ફંક્શનમાં પાસ કરેલા ઈનપુટ મૂલ્યો
- રિટર્ન વેલ્યુ: ફંક્શનમાંથી આઉટપુટ અથવા પરિણામ

મેમરી ટ્રીક

"ફંક્શન કાર્યોને ENCAPSULATE કરે, INPUTS લે, OUTPUTS આપે"