Programming In C (4331105) - Summer 2024 Solution

Milav Dabgar

June 18, 2024

પ્રશ્ન 1 [a ગુણ]

3 કીવર્ડ વ્યાખ્યાયિત કરો. C ભાષા માટે કોઈપણ ચાર કીવર્ડ્સની યાદી આપો.

જવાબ

કીવર્ડ એ C માં પૂર્વવ્યાખ્યાયિત, આરક્ષિત શબ્દ છે જેનો કમ્પાઈલર માટે વિશેષ અર્થ છે અને તેનો ઉપયોગ આઇડેન્ટિફાયર તરીકે કરી શકાતો નથી.

કોષ્ટક: સામાન્ય C કીવર્ડ્સ

કીવર્ડ	હેતુ
int	ઇન્ટીજર ડેટા ટાઈપ
float	ફ્લોટિંગ-પોઈન્ટ ડેટા ટાઈપ
char	કેરેક્ટર ડેટા ટાઈપ
if	શરતી સ્ટેટમેન્ટ
for	લૂપ સ્ટેટમેન્ટ
while	લૂપ સ્ટેટમેન્ટ
void	રિટર્ન ટાઈપ/પેરામીટર
return	ફંક્શનમાંથી કિંમત પરત કરવી

- **આરક્ષિત શબ્દો**: કીવર્ડ્સનો ઉપયોગ વેરિએબલ નામો તરીકે કરી શકાતો નથી
- **પૂર્વ-વ્યાખ્યાયિત**: ભાષામાં તેમનો નિશ્ચિત અર્થ છે
- **કેસ-સેન્સિટિવ**: બધા કીવર્ડ્સ લોઅરકેસમાં હોવા જોઈએ

મેમરી ટ્રીક

"If VoId FoR WhIle" (મહત્વપૂર્ણ કીવર્ડ્સના પ્રથમ અક્ષરો)

પ્રશ્ન 1 [b ગુણ]

4 વેરિએબલનું નામકરણ કરવા માટેના નિયમો સમજાવો.

જવાબ

C માં વેરિએબલ્સ માન્ય આઇડેન્ટિફાયર હોવા માટે ચોક્કસ નામકરણ નિયમોનું પાલન કરવું આવશ્યક છે.

કોષ્ટક: C માં વેરિએબલ નામકરણ નિયમો

નિયમ	વિવરણ	માન્ય ઉદાહરણ	અમાન્ય ઉદાહરણ
પ્રથમ અક્ષર	અક્ષર અથવા અંડરસ્કોર હોવો જોઈએ	age, _count	1value
પછીના અક્ષરો	અક્ષરો, નંબરો અથવા અંડરસ્કોર	user_1, total99	user@1
કેસ સેન્સિટિવિટી	અપરકેસ અને લોઅરકેસ અલગ છે	Value $\neq$ value	-
કીવર્ડ્સ	આરક્ષિત કીવર્ડ્સનો ઉપયોગ કરી શકાતો નથી	counter	int
લંબાઈ	અર્થપૂર્ણ હોવું જોઈએ પણ વધુ લાંબુ નહીં	studentMarks	sm
વિશેષ અક્ષરો	માન્ય નથી	firstName	first-name

- **વર્ણનાત્મક નામો**: અર્થપૂર્ણ નામોનો ઉપયોગ કરો જે હેતુ સૂચવે છે
- **સુસંગત શૈલી**: સુસંગત નામકરણ સંમેલનનું પાલન કરો
- **જગ્યા નહીં**: અંડરસ્કોર અથવા camelCase નો ઉપયોગ કરો

મેમરી ટ્રીક

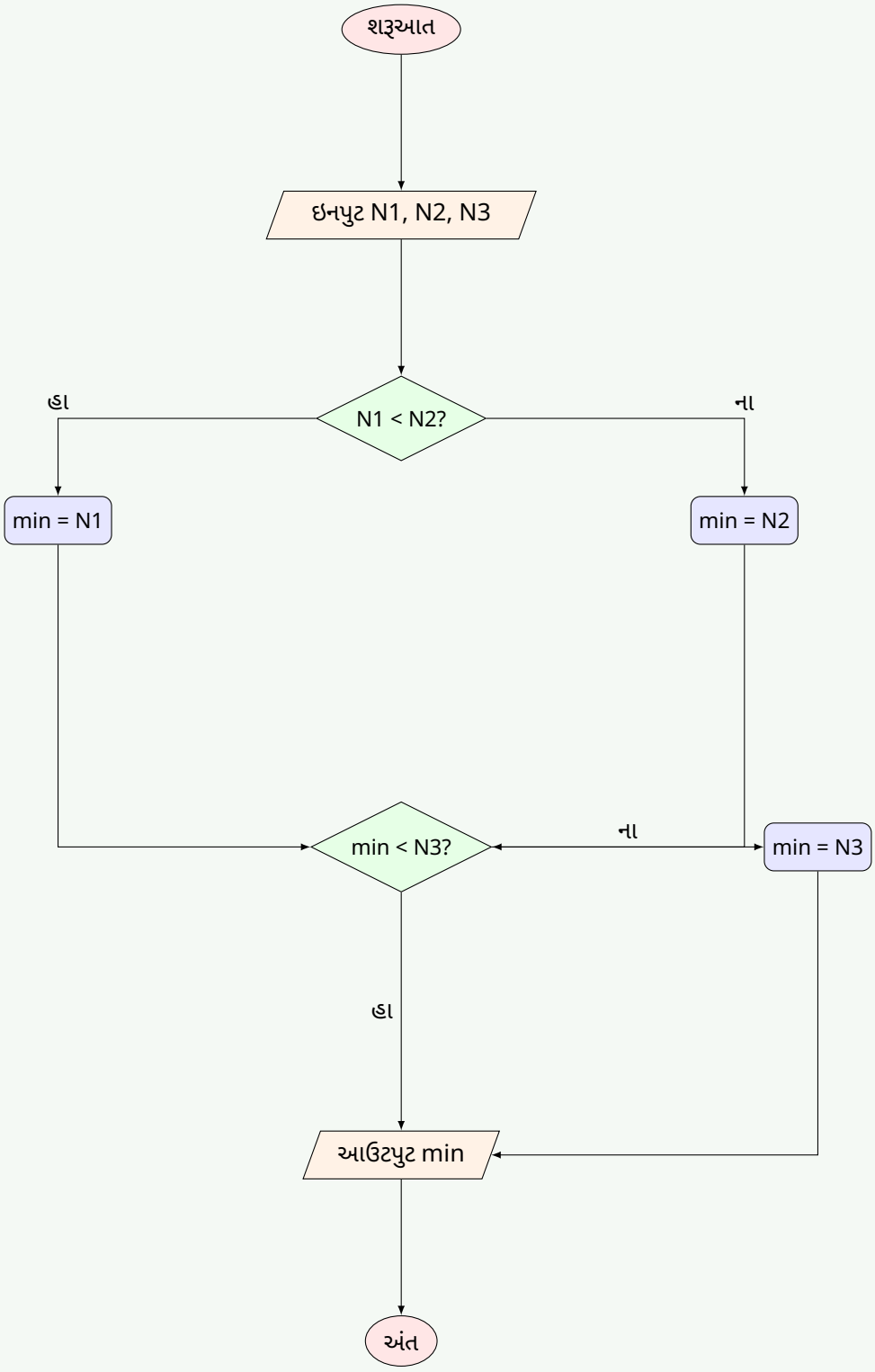
"FLASKS" (First Letter, Letters/digits, Avoid keywords, Sensitive case, Keep meaningful, Skip special chars)

પ્રશ્ન 1 [૮ ગુણ]

7 ફ્લોચાર્ટ વ્યાખ્યાયિત કરો. ત્રણ પૂર્ણાંક નંબરો N1, N2 અને N3 માંથી લઘુત્તમ શોધવા માટે ફ્લોચાર્ટ દોરો.

જવાબ

ફ્લોચાર્ટ એ અલ્ગોરિધમનું એક ગ્રાફિકલ નિરૂપણ છે જે બોક્સ તરીકે પગલાંઓ અને તીર દ્વારા તેમના ક્રમને જોડે છે.
ફ્લોચાર્ટ: ત્રણ નંબરોમાંથી લઘુત્તમ



- **પ્રતીકો**: અંડાકાર (શરૂઆત/અંત), સમાંતરભુજ (ઇનપુટ/આઉટપુટ), ડાયમંડ (નિર્ણય), લંબચોરસ (પ્રક્રિયા)
- **નિર્ણય બિંદુઓ**: મૂલ્યોની વ્યવસ્થિત સરખામણી
- **તાર્કિક પ્રવાહ**: તીર પ્રક્રિયાઓનો ક્રમ બતાવે છે

મેમરી ટ્રીક

"Start-Input-Compare-Output-End" (SICOE)

OR

પ્રશ્ન 1 [c ગુણ]

7 અલ્ગોરિધમ વ્યાખ્યાયિત કરો. ત્રણ પૂર્ણાંક નંબરો N1, N2 અને N3 માંથી લઘુત્તમ શોધવા માટે અલ્ગોરિધમ લખો.

જવાબ

અલ્ગોરિધમ એ કોઈ ચોક્કસ સમસ્યા ઉકેલવા માટેના પગલા-દર-પગલાની પ્રક્રિયા અથવા સારી રીતે વ્યાખ્યાયિત સૂચનાઓનો સમૂહ છે.

અલ્ગોરિધમ: ત્રણ નંબરોમાંથી લઘુત્તમ શોધવું

- **પગલું 1**: શરૂઆત
- **પગલું 2**: ત્રણ નંબરો N1, N2, અને N3 ઇનપુટ કરો
- **પગલું 3**: min = N1 સેટ કરો (ધારો કે પ્રથમ નંબર લઘુત્તમ છે)
- **પગલું 4**: જો N2 < min, તો min = N2 સેટ કરો
- **પગલું 5**: જો N3 < min, તો min = N3 સેટ કરો
- **પગલું 6**: લઘુત્તમ નંબર તરીકે min આઉટપુટ કરો
- **પગલું 7**: અંત

કોષ્ટક: અલ્ગોરિધમ લાક્ષણિકતાઓ

લાક્ષણિકતા	વિવરણ
સીમિતતા (Finiteness)	અલ્ગોરિધમ નિશ્ચિત પગલાં પછી સમાપ્ત થવું જોઈએ
નિશ્ચિતતા (Definiteness)	દરેક પગલું ચોક્કસ રીતે વ્યાખ્યાયિત હોવું જોઈએ
ઇનપુટ (Input)	અલ્ગોરિધમ શૂન્ય અથવા વધુ ઇનપુટ લે છે
આઉટપુટ (Output)	અલ્ગોરિધમ એક અથવા વધુ આઉટપુટ આપે છે
અસરકારકતા (Effectiveness)	પગલાં સરળ અને અમલ યોગ્ય હોવા જોઈએ

- **ક્રમિક પગલાં**: તાર્કિક ક્રમ અનુસરે છે
- **તુલનાત્મક અભિગમ**: વ્યવસ્થિત રીતે લઘુત્તમ શોધે છે
- **સરળતા**: સમજવા અને અમલ કરવા માટે સરળ

મેમરી ટ્રીક

"FIDEO" (Finiteness, Input, Definiteness, Effectiveness, Output)

પ્રશ્ન 2 [a ગુણ]

3 gets() અને puts() વચ્ચે તફાવત આપો.

જવાબ

gets() અને puts() એ સ્ટ્રિંગ સાથે ઇનપુટ અને આઉટપુટ ઓપરેશન્સ માટે C માં સ્ટાન્ડર્ડ લાઇબ્રેરી ફંક્શન્સ છે.

કોષ્ટક: gets() અને puts() ની સરખામણી

લાક્ષણિકતા	gets()	puts()
હેતુ	stdin માંથી સ્ટ્રિંગ વાંચે છે	stdout માં સ્ટ્રિંગ લખે છે
પ્રોટોટાઇપ	char *gets(char *str)	int puts(const char *str)
વર્તણૂક	ન્યૂલાઇન સુધી વાંચે છે	સ્વચાલિત રીતે ન્યૂલાઇન ઉમેરે છે
રિટર્ન વેલ્યુ	સફળતા પર str, નિષ્ફળતા પર NULL	સફળતા પર નોન-નેગેટિવ, એરર પર EOF
સુરક્ષા	અસુરક્ષિત (બફર ઓવરફ્લો જોખમ)	સુરક્ષિત
ભલામણ	ના (ડિપ્રિકેટેડ)	હા

- **ઇનપુટ/આઉટપુટ**: ઇનપુટ માટે gets(), આઉટપુટ માટે puts()
- **ટર્મિનેશન**: gets() ન્યૂલાઇન પર અટકે છે, puts() ન્યૂલાઇન ઉમેરે છે
- **સુરક્ષા**: gets() માં કોઈ બફર લિમિટ ચેક નથી

મેમરી ટ્રીક

"Gets In, Puts Out" (gets અંદર વાંચે છે, puts બહાર લખે છે)

પ્રશ્ન 2 [b ગુણ]

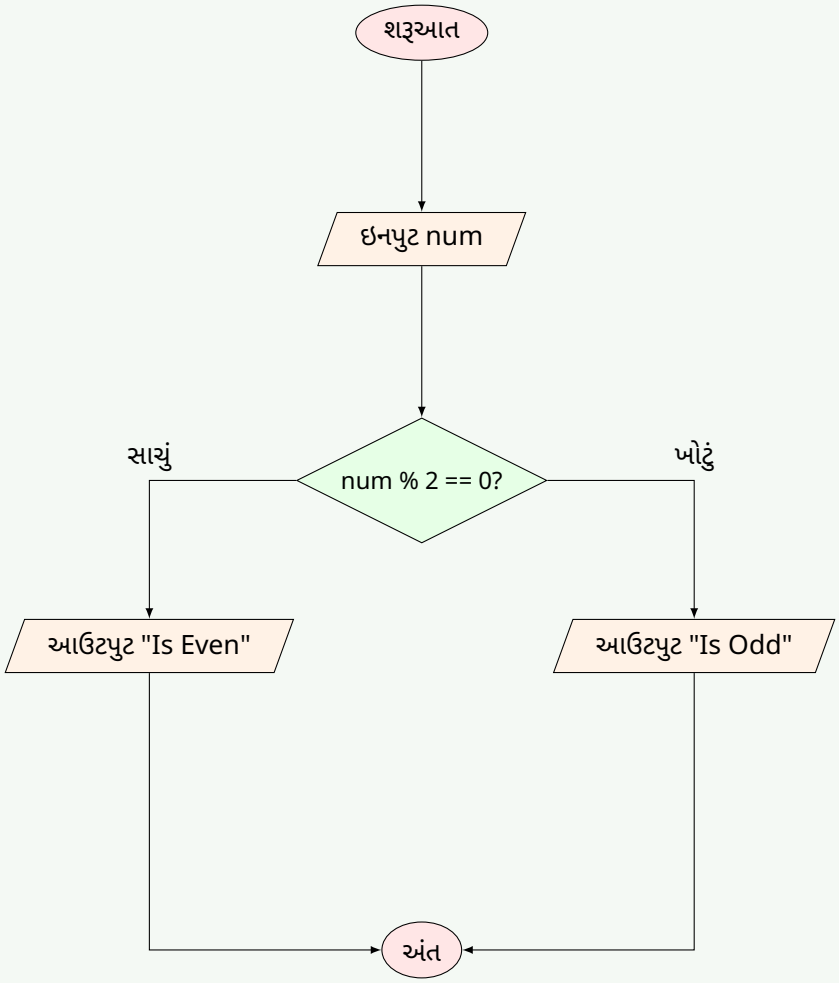
4 દાખલ કરેલ નંબર બેકી છે કે એકી તે શોધવા માટે કંડિશનલ ઓપરેટરનો ઉપયોગ કરીને C પ્રોગ્રામ બનાવો.

જવાબ

આ પ્રોગ્રામ નંબર બેકી છે કે એકી તે તપાસવા માટે કંડિશનલ ઓપરેટરનો ઉપયોગ કરે છે.

```c
#include <stdio.h>

int main() {
    int num;

    printf("Enter a number: ");
    scanf("%d", &num);

    // બેકી અથવા એકી તપાસવા માટે કંડશિનલ ઓપરેટરનો ઉપયોગ
    (num % 2 == 0) ? printf("%d is even\n", num) : printf("%d is odd\n", num);

    return 0;
}
```

ફ્લોચાર્ટ: બેકી (Even) અથવા એકી (Odd)



- **કંડિશનલ ઓપરેટર**: ? : એ ટર્નરી ઓપરેટર છે
- **મોડ્યુલસ ઓપરેશન**: % ભાગાકાર પછી શેષ આપે છે
- **ટેસ્ટ કન્ડિશન**: num % 2 == 0 બેકી નંબર માટે તપાસે છે

મેમરી ટ્રીક

"REMinder 0 = Even" (શેષ 0 એટલે બેકી)

પ્રશ્ન 2 [૮ ગુણ]

7 લોજીકલ અને રિલેશનલ ઓપરેટરો ઉદાહરણ સાથે સમજાવો.

જવાબ

લોજીકલ અને રિલેશનલ ઓપરેટરોનો ઉપયોગ C પ્રોગ્રામ્સમાં શરતો બનાવવા અને નિર્ણયો લેવા માટે થાય છે.
કોષ્ટક: રિલેશનલ ઓપરેટરો

ઓપરેટર	અર્થ	ઉદાહરણ	પરિણામ
==	સમાન છે	5 == 5	સાચું (1)
!=	સમાન નથી	5 != 3	સાચું (1)
>	કરતા મોટું	7 > 3	સાચું (1)
<	કરતા નાનું	2 < 8	સાચું (1)
>=	કરતા મોટું કે સમાન	4 >= 4	સાચું (1)
<=	કરતા નાનું કે સમાન	6 <= 9	સાચું (1)

કોષ્ટક: લોજીકલ ઓપરેટરો

ઓપરેટર	અર્થ	ઉદાહરણ	પરિણામ
&&	લોજીકલ AND	(5>3) && (8>5)	સાચું (1)
||	લોજીકલ OR	(5>7) || (3<6)	સાચું (1)
!	લોજીકલ NOT	!(5>7)	સાચું (1)

```c
int age = 20;
int score = 75;

// રિલેશનલ અને લોજીકલ બંને ઓપરેટરોનો ઉપયોગ
if ((age >= 18) && (score > 70)) {
   printf("Eligible");
}
```

- **સરખામણી**: રિલેશનલ ઓપરેટરો કિંમતોની સરખામણી કરે છે
- **શરતો સંયોજન**: લોજીકલ ઓપરેટરો બહુવિધ શરતોને જોડે છે
- **સત્ય કિંમત**: બધા ઓપરેટરો 1 (સાચું) અથવા 0 (ખોટું) પરત કરે છે

મેમરી ટ્રીક

"CORNL" (Compare with relational, OR/AND/NOT with logical)

OR

પ્રશ્ન 2 [a ગુણ]

3 ઓપરેટરોના પ્રેસીડન્સને ધ્યાનમાં રાખીને, જો 16 + (216 / ((3 + 6) * 12)) -10 એક્સપ્રેશનને ઉકેલવામાં આવે તો દરેક સ્ટેપ અને અંતિમ જવાબ લખો.

જવાબ

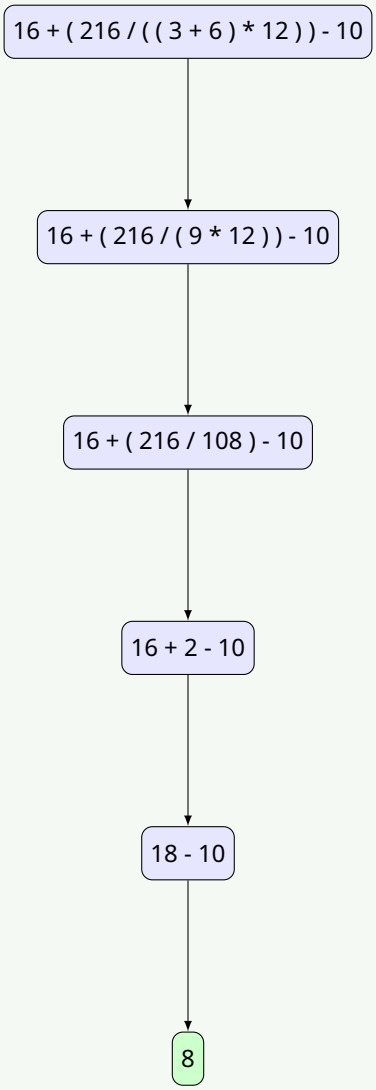
ચાલો ઓપરેટર પ્રેસીડન્સને અનુસરીને એક્સપ્રેશન 16 + (216 / ((3 + 6) * 12)) - 10 નું મૂલ્યાંકન કરીએ.

કોષ્ટક: સ્ટેપ-બાય-સ્ટેપ મૂલ્યાંકન

સ્ટેપ	ઓપરેશન	આ સ્ટેપ પછીનું એક્સપ્રેશન
1	(3 + 6) ગણતરી	16 + (216 / (9 * 12)) - 10
2	(9 * 12) ગણતરી	16 + (216 / 108) - 10
3	(216 / 108) ગણતરી	16 + 2 - 10
4	16 + 2 ગણતરી	18 - 10
5	18 - 10 ગણતરી	8

અંતિમ જવાબ: 8

મૂલ્યાંકન ટ્રી:



- **કૌંસ પહેલા:** સૌથી અંદરના કૌંસનું પહેલા મૂલ્યાંકન થાય છે
- **ભાગાકાર પહેલા ગુણાકાર:** ડાબી બાજુથી જમણી બાજુ ગણતરી
- **સરવાળો અને બાદબાકી છેલ્લે:** ડાબી બાજુથી જમણી બાજુ

મેમરી ટ્રીક

"PEMDAS" (Parentheses, Exponents, Multiplication/Division, Addition/Subtraction)

પ્રશ્ન 2 [b ગુણ]

4 વર્તુળનો પરિઘ અને ક્ષેત્રફળ શોધવા માટે C પ્રોગ્રામ લખો.

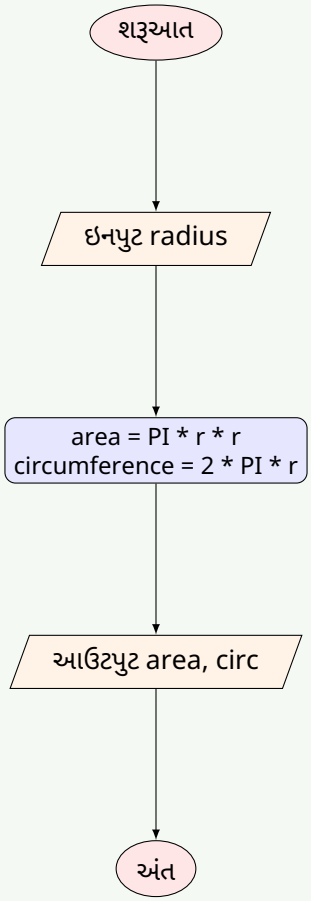
જવાબ

આ પ્રોગ્રામ ત્રિજ્યાના આધારે વર્તુળનું ક્ષેત્રફળ અને પરિઘ ગણે છે.

```c
#include <stdio.h>
#define PI 3.14159

int main() {
```

```c
    float radius, area, circumference;

    printf("Enter the radius of circle: ");
    scanf("%f", &radius);

    // ક્ષેત્રફળ અને પરિધની ગણતરી
    area = PI * radius * radius;
    circumference = 2 * PI * radius;

    printf("Area of circle = %.2f square units\n", area);
    printf("Circumference of circle = %.2f units\n", circumference);

    return 0;
}
```

ફ્લોચાર્ટ: ક્ષેત્રફળ અને પરિઘ



- **ફોર્મ્યુલા**: ક્ષેત્રફળ = $\pi \times r^2$ અને પરિઘ = $2\pi \times r$
- **કોન્સ્ટન્ટ ડેફિનેશન**: PI માટે #define નો ઉપયોગ
- **ફ્લોટ વેરિએબલ્સ**: દશાંશ ચોકસાઈ માટે

મેમરી ટ્રીક

"PIR^2" ક્ષેત્રફળ માટે, "2PIR" પરિઘ માટે

પ્રશ્ન 2 [c ગુણ]

7 એરીથમેટીક અને બીટ-વાઈઝ ઓપરેટરો ઉદાહરણ સાથે સમજાવો.

જવાબ

એરીથમેટીક ઓપરેટરો ગાણિતિક ક્રિયાઓ કરે છે જ્યારે બીટ-વાઈઝ ઓપરેટરો પૂર્ણાંકોના વ્યક્તિગત બીટ્સને મેનિપ્યુલેટ કરે છે.

કોષ્ટક: એરીથમેટીક ઓપરેટરો

Op	વિવરણ	ઉદાહરણ	પરિણામ
+	સરવાળો	5 + 3	8
-	બાદબાકી	7 - 2	5
*	ગુણાકાર	4 * 3	12
/	ભાગાકાર	10 / 3	3
%	મોડ્યુલસ	10 % 3	1
++	ઇન્ક્રીમેન્ટ	a++	1 ઉમેરે
--	ડિક્રીમેન્ટ	--b	1 બાદ કરે

કોષ્ટક: બીટવાઈઝ ઓપરેટરો

Op	વિવરણ	Exp (bin)	Res
&	બીટવાઈઝ AND	5(101) & 3(011)	1(001)
|	બીટવાઈઝ OR	5(101) | 3(011)	7(111)
^	બીટવાઈઝ XOR	5(101) ^ 3(011)	6(110)
~	બીટવાઈઝ NOT	5(101)	-6
<<	લેફ્ટ શિફ્ટ	5 << 1	10(1010)
>>	રાઈટ શિફ્ટ	5 >> 1	2(10)

```c
int a = 5, b = 3;
printf("a + b = %d\n", a + b);     // 8
printf("a & b = %d\n", a & b);     // 1
printf("a << 1 = %d\n", a << 1);   // 10
```

- **ગાણિતિક ક્રિયાઓ**: ગણતરી માટે એરીથમેટીક ઓપરેટરો
- **બીટ મેનિપ્યુલેશન**: બીટવાઈઝ ઓપરેટરો બાઈનરી સ્તરે કામ કરે છે
- **કાર્યક્ષમતા**: બીટવાઈઝ ઓપરેશન્સ અમુક કાર્યો માટે ઝડપી છે

મેમરી ટ્રીક

"SAME BARON" (Subtraction Addition Multiplication, Bitwise AND/OR/NOT)

પ્રશ્ન 3 [a ગુણ]

3 'go to' સ્ટેટમેન્ટનો ઉપયોગ ઉદાહરણ સાથે સમજાવો.

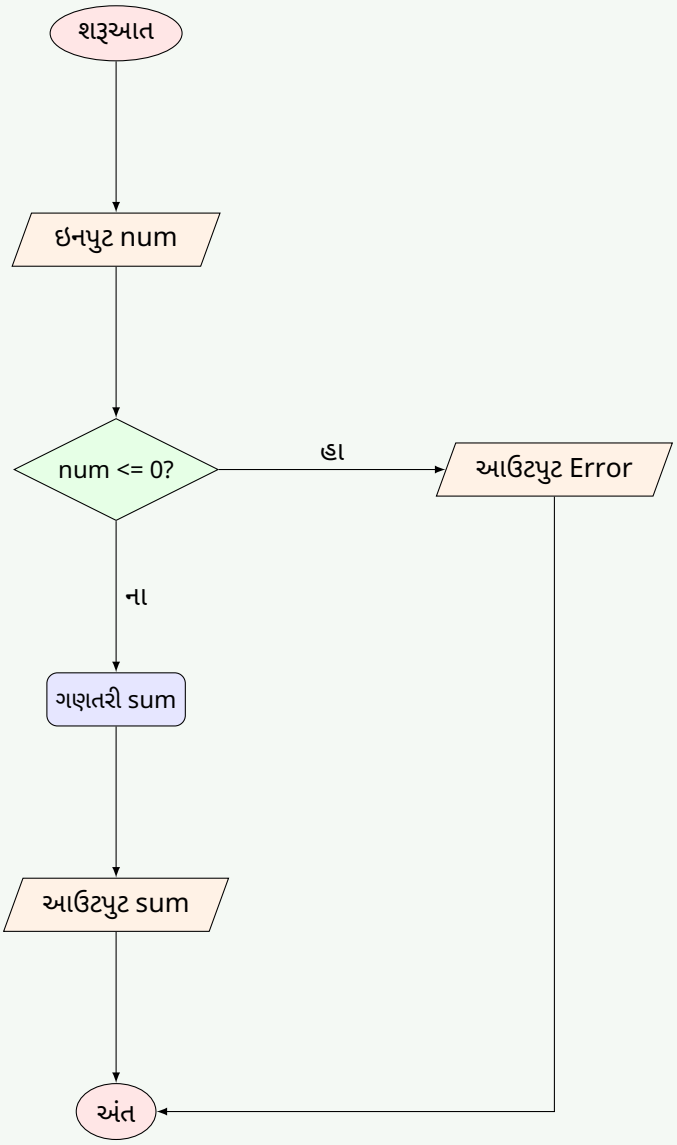
જવાબ

goto સ્ટેટમેન્ટનો ઉપયોગ પ્રોગ્રામ કંટ્રોલને લેબલવાળા સ્ટેટમેન્ટમાં બિનશરતી ટ્રાન્સફર કરવા માટે થાય છે.

```c
#include <stdio.h>

int main() {
    int num, sum = 0;

    printf("Enter a positive number: ");
    scanf("%d", &num);

    if (num <= 0) {
        goto error;
    }

```

```
    sum = num * (num + 1) / 2;
    printf("Sum of first %d numbers = %d\n", num, sum);
    goto end;

    error:
        printf("Error: Please enter a positive number!\n");

    end:
        return 0;
}
```

ફ્લોચાર્ટ: goto ઉદાહરણ



- **લેબલ ડિક્લેરેશન**: લેબલ્સ કોલોન (:) સાથે સમાપ્ત થાય છે
- **જમ્પ સ્ટેટમેન્ટ**: goto કંટ્રોલને લેબલ પર ટ્રાન્સફર કરે છે
- **સાવચેતી**: વધુ પડતો ઉપયોગ "સ્પાઘેટ્ટી કોડ" બનાવે છે

મેમરી ટ્રીક

"JUMPing LABEL" (લેબલવાળા સ્ટેટમેન્ટ પર જમ્પ કરો)

પ્રશ્ન 3 [b ગુણ]

4 કીબોર્ડ દ્વારા ઇનપુટ કરેલા 5 વિષયોમાં વિદ્યાર્થીએ મેળવેલા ગુણના આધારે વિદ્યાર્થીને નીચેના નિયમો મુજબ ગ્રેડ મળે છે: 90 કે તેથી વધુ ટકા - ગ્રેડ A. 80 અને 89 ની વચ્ચે ટકા - ગ્રેડ B. 70 અને 79 ની વચ્ચે ટકા - ગ્રેડ C. 60 અને 69 ની વચ્ચે ટકા - ગ્રેડ D. 50 અને 59 ની વચ્ચે ટકા - ગ્રેડ E. 50 કરતા ઓછા - ગ્રેડ F. વિદ્યાર્થીએ મેળવેલ ગ્રેડ પ્રદર્શિત કરતો C પ્રોગ્રામ લખો.

જવાબ

આ પ્રોગ્રામ 5 વિષયોના સરેરાશ ગુણના આધારે ગ્રેડની ગણતરી કરે છે.

```c
#include <stdio.h>

int main() {
    int marks[5], total = 0, i;
    float percentage;
    char grade;

    // 5 વિષયો માટે ગુણ ઇનપુટ
    for (i = 0; i < 5; i++) {
        printf("Enter marks for subject %d (out of 100): ", i+1);
        scanf("%d", &marks[i]);
        total += marks[i];
    }

    // ટકાવારીની ગણતરી
    percentage = total / 5.0;

    // ગ્રેડ નક્કી કરો
    if (percentage >= 90)
        grade = 'A';
    else if (percentage >= 80)
        grade = 'B';
    else if (percentage >= 70)
        grade = 'C';
    else if (percentage >= 60)
        grade = 'D';
    else if (percentage >= 50)
        grade = 'E';
    else
        grade = 'F';

    printf("Percentage: %.2f%%\n", percentage);
    printf("Grade: %c\n", grade);

    return 0;
}
```

કોષ્ટક: ગ્રેડિંગ માપદંડ

ટકાવારી શ્રેણી	ગ્રેડ
≥ 90	A
80-89	B
70-79	C
60-69	D
50-59	E
< 50	F

- **ઇનપુટ એરે**: 5 વિષયોના ગુણ સ્ટોર કરે છે
- **ટકાવારી ગણતરી**: સરવાળાને વિષયોની સંખ્યા વડે ભાગવું
- **ગ્રેડ નિર્ધારણ**: if-else લેડરનો ઉપયોગ કરીને

મેમરી ટ્રીક

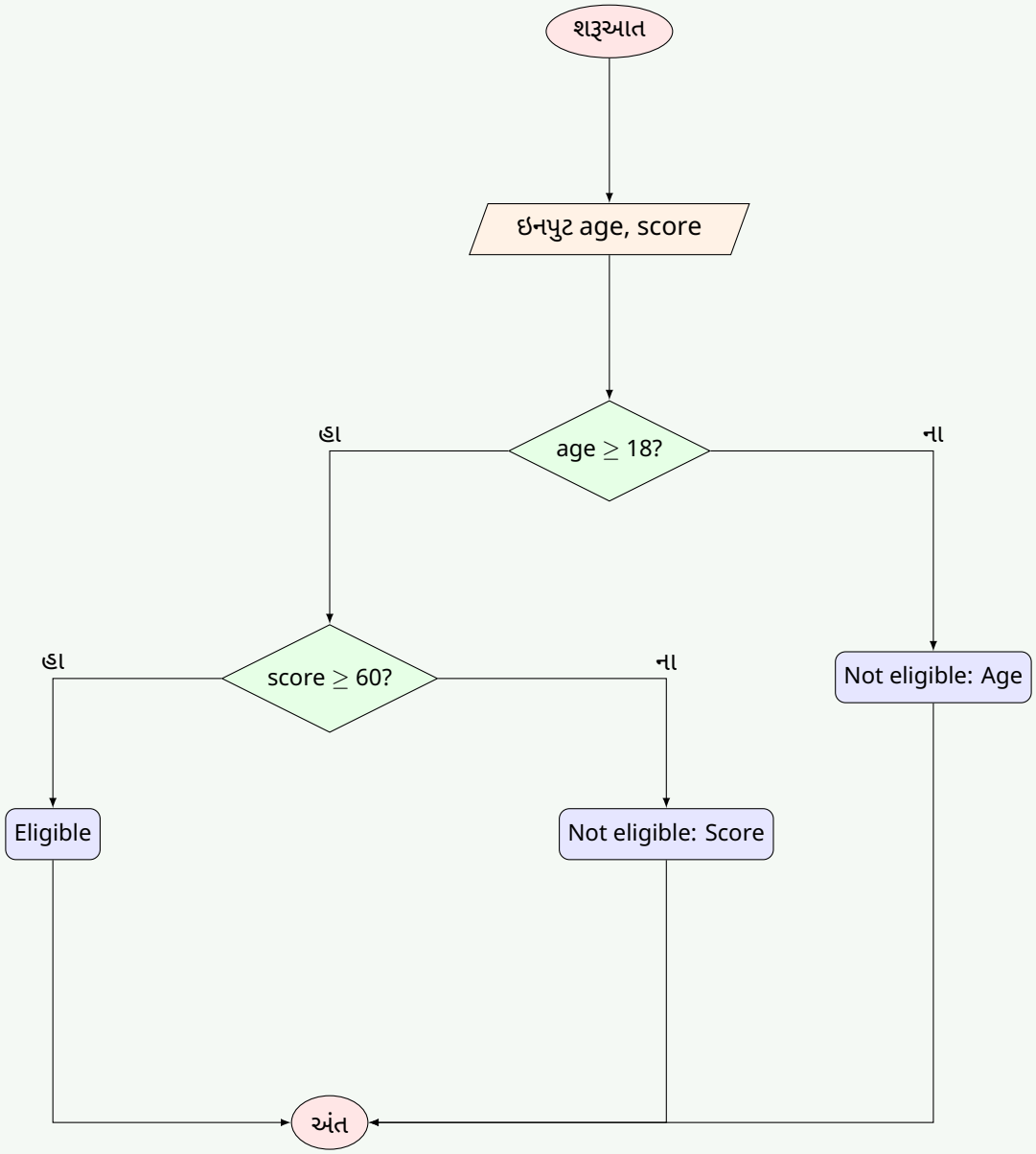
"ABCDEF-90-80-70-60-50" (ગ્રેડ્સ અને તેમની ટકાવારી મર્યાદા)

પ્રશ્ન 3 [c ગુણ]

7 નેસ્ટેડ if-else ફ્લોચાર્ટ દોરી ઉદાહરણ સાથે સમજાવો.

જવાબ

નેસ્ટેડ if-else એ કંટ્રોલ સ્ટ્રક્ચર છે જ્યાં if અથવા else સ્ટેટમેન્ટની અંદર બીજું if-else સ્ટેટમેન્ટ હોય છે.
ફ્લોચાર્ટ: નેસ્ટેડ if-else




```c
#include <stdio.h>

```

```c
int main() {
    int age, score;

    printf("Enter age: ");
    scanf("%d", &age);
    printf("Enter score: ");
    scanf("%d", &score);

    if (age >= 18) {
        if (score >= 60) {
            printf("Eligible for admission");
        } else {
            printf("Not eligible: Score criteria not met");
        }
    } else {
        printf("Not eligible: Age criteria not met");
    }

    return 0;
}
```

- **બહુવિધ શરતો**: ક્રમમાં અનેક શરતો ચકાસે છે
- **હાયરાર્કિકલ નિર્ણય**: અંદરની શરત માત્ર ત્યારે જ મૂલ્યાંકન થાય છે જો બહારની સાચી હોય
- **ઇન્ડેન્ટેશન**: યોગ્ય ઇન્ડેન્ટેશન માળખું સમજવામાં મદદ કરે છે

મેમરી ટ્રીક

"CONE" (Check Outer, Nest Evaluation inside)

OR

પ્રશ્ન 3 [a ગુણ]

3 continue અને break સ્ટેટમેન્ટનો ઉપયોગ સમજાવો.

જવાબ

break અને continue સ્ટેટમેન્ટ્સ લૂપ્સના પ્રવાહને અલગ અલગ રીતે નિયંત્રિત કરે છે.

કોષ્ટક: break અને continue વચ્ચે સરખામણી

લાક્ષણિકતા	break	continue
હેતુ	લૂપમાંથી તરત જ બહાર નીકળે છે	વર્તમાન પુનરાવર્તન છોડી દે છે
લૂપ પર અસર	સંપૂર્ણપણે સમાપ્ત કરે છે	આગલા પુનરાવર્તન પર આગળ વધે છે
ઉપયોગ	switch, for, while, do-while	for, while, do-while
ક્યારે વાપરવું	જ્યારે શરત પૂરી થાય અને વધુ પુનરાવર્તનોની જરૂર ન હોય	જ્યારે વર્તમાન પુનરાવર્તન છોડી દેવું જોઈએ

```c
// break સાથે ઉદાહરણ
for (int i = 1; i <= 10; i++) {
    if (i == 5)
        break;    // i 5 હોય ત્યારે લૂપમાંથી બહાર નીકળો
    printf("%d ", i);  // આઉટપુટ: 1 2 3 4
}
```

```c
// continue સાથે ઉદાહરણ
for (int i = 1; i <= 10; i++) {
    if (i % 2 == 0)
        continue;  // બેકી નંબરો છોડી દો
```

```
    printf("%d ", i);  // આઉટપુટ: 1 3 5 7 9
}
```

- **લૂપ કંટ્રોલ**: બંને લૂપ એક્ઝિક્યુશન મેનેજ કરવા માટે વપરાય છે
- **Break એક્ઝિટ**: લૂપને સંપૂર્ણપણે અટકાવે છે
- **Continue સ્કિપ**: માત્ર વર્તમાન પુનરાવર્તન છોડે છે

મેમરી ટ્રીક

"BEC" (Break Exits Completely, Continue only current)

પ્રશ્ન 3 [b ગુણ]

4 for લૂપનો ઉપયોગ કરીને આ આઉટપુટ પ્રિન્ટ કરવા માટે પ્રોગ્રામ લખો:

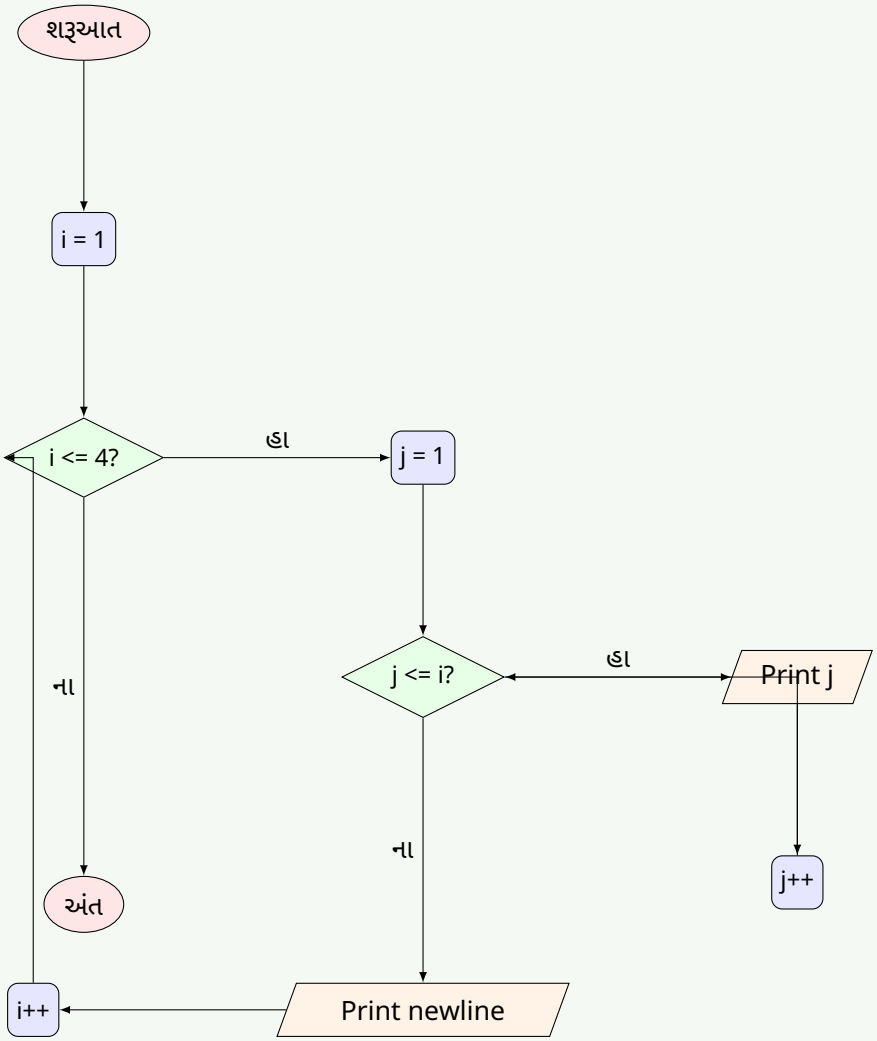
1
1 2
1 2 3
1 2 3 4

જવાબ

આ પ્રોગ્રામ નંબરોની પેટર્ન પ્રિન્ટ કરવા માટે નેસ્ટેડ for લૂપ્સનો ઉપયોગ કરે છે.

```c
#include <stdio.h>

int main() {
    int i, j;

    // પંક્તિઓ માટે બહારની લૂપ (1 થી 4)
    for (i = 1; i <= 4; i++) {
        // કોલમ માટે અંદરની લૂપ (1 થી i)
        for (j = 1; j <= i; j++) {
            printf("%d ", j);
        }
        printf("\n");  // દરેક પંક્તિ પછી નવી લાઇન પર જાઓ
    }

    return 0;
}
```

ફ્લોચાર્ટ: પેટર્ન પ્રિન્ટિંગ



- **નેસ્ટેડ લૂપ્સ**: પંક્તિઓ માટે બહારની લૂપ, કોલમ માટે અંદરની
- **ડાયનેમિક લિમિટ**: અંદરની લૂપ j ને 1 થી વર્તમાન i સુધી ચલાવે છે
- **વધતી પેટર્ન**: દરેક પંક્તિમાં એક વધુ નંબર હોય છે

મેમરી ટ્રીક

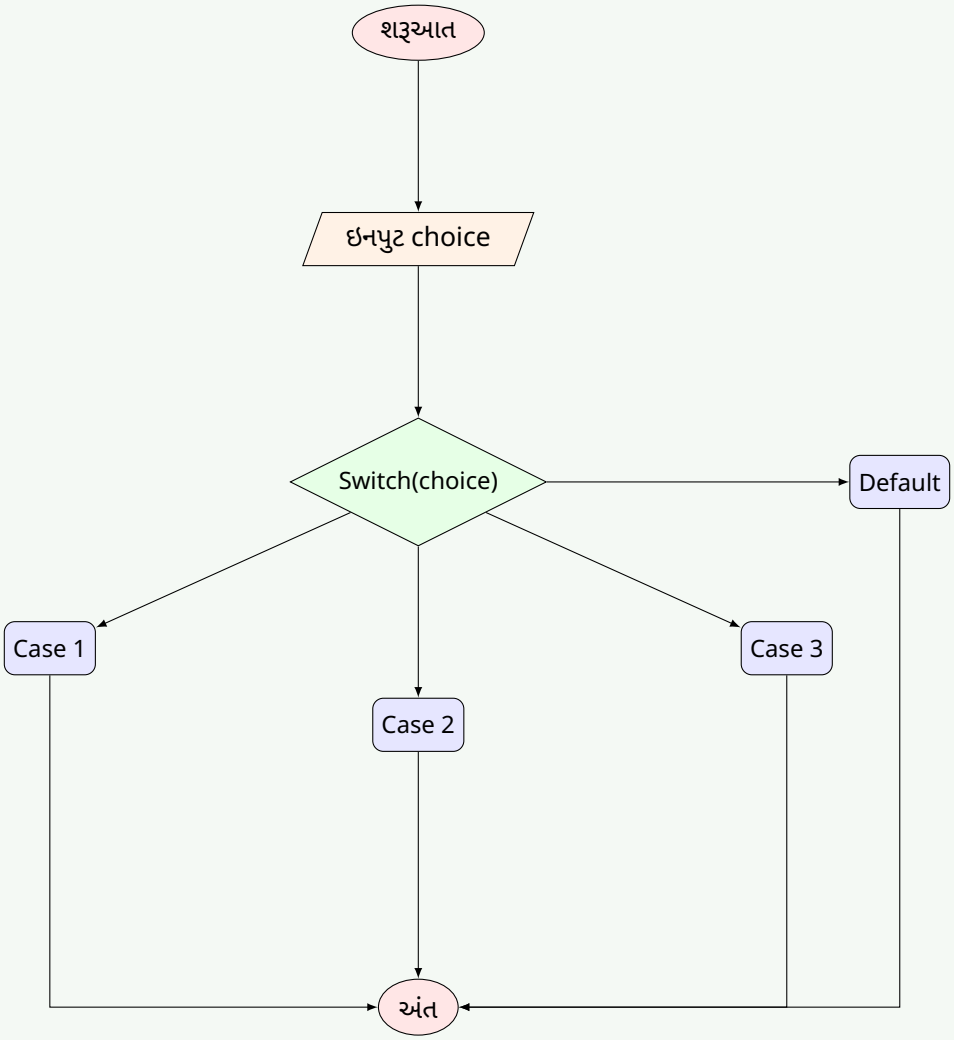
"RICI" (Row Increases, Column Increases based on row number)

પ્રશ્ન 3 [c ગુણ]

7 સ્વિચ સ્ટેટમેન્ટ ફ્લોચાર્ટ દોરી ઉદાહરણ સાથે સમજાવો.

જવાબ

switch સ્ટેટમેન્ટ એ મલ્ટી-વે નિર્ણય લેનાર છે જે વેરિએબલને વિવિધ કેસ મૂલ્યો સામે ચકાસે છે.
ફ્લોચાર્ટ: સ્વિચ સ્ટેટમેન્ટ




```c
#include <stdio.h>

int main() {
    int choice;

    printf("Menu:\n");
    printf("1. Add\n");
    printf("2. Subtract\n");
    printf("3. Multiply\n");
    printf("Enter your choice (1-3): ");
    scanf("%d", &choice);

    switch (choice) {
        case 1:
            printf("Addition selected\n");
            break;
        case 2:
            printf("Subtraction selected\n");
            break;
        case 3:
            printf("Multiplication selected\n");
            break;
        default:
            printf("Invalid choice\n");
    }
```

```
    return 0;
}
```

- **બહુવિધ કેસો**: એક વેરિએબલને અનેક મૂલ્યો સામે ચકાસે છે
- **બ્રેક સ્ટેટમેન્ટ**: આગલા કેસમાં જવાથી રોકે છે
- **ડિફોલ્ટ કેસ**: કોઈપણ કેસ સાથે મેચ ન થતા મૂલ્યોને હેન્ડલ કરે છે
- **કેસનો ક્રમ**: કોઈપણ ક્રમમાં હોઈ શકે છે, ડિફોલ્ટ સામાન્ય રીતે છેલ્લે હોય છે

મેમરી ટ્રીક

"CASED" (Check All Switch Expression's Destinations)

પ્રશ્ન 4 [a ગુણ]

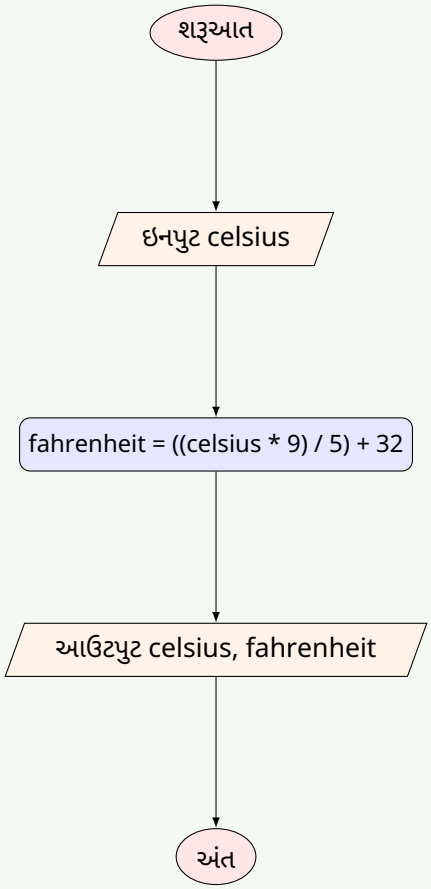
3 તાપમાનને સેલ્સિયસમાંથી ફેરનહીટમાં રૂપાંતરિત કરવા માટેનો C પ્રોગ્રામ fahrenheit= ((celsius*9)/5)+32 સૂત્રનો ઉપયોગ કરીને વિકસાવો.

જવાબ

આ પ્રોગ્રામ સેલ્સિયસમાંથી ફેરનહીટમાં તાપમાન મૂલ્યને રૂપાંતરિત કરે છે.

```c
#include <stdio.h>

int main() {
    float celsius, fahrenheit;

    printf("Enter temperature in Celsius: ");
    scanf("%f", &celsius);

    // સેલ્સિયસને ફેરનહીટમાં કન્વર્ટ કરો
    fahrenheit = ((celsius * 9) / 5) + 32;

    printf("%.2f Celsius = %.2f Fahrenheit\n", celsius, fahrenheit);

    return 0;
}
```

ફ્લોચાર્ટ: સેલ્સિયસ થી ફેરનહીટ



- **ફોર્મ્યુલા**: $F = ((C \times 9) \div 5) + 32$
- **ફ્લોટ વેરિએબલ્સ**: દશાંશ ચોકસાઈ માટે
- **ફોર્મેટેડ આઉટપુટ**: બે દશાંશ સ્થાનો માટે %.2f નો ઉપયોગ કરીને

મેમરી ટ્રીક

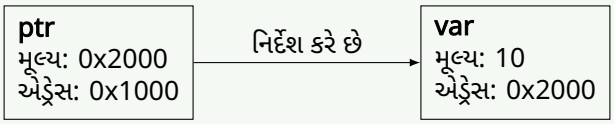
"C95+32=F" (Celsius x 9 / 5 + 32 = Fahrenheit)

પ્રશ્ન 4 [b ગુણ]

4 પોઈન્ટર એટલે શું? ઉદાહરણ સાથે સમજાવો.

જવાબ

પોઈન્ટર એ એક વેરિએબલ છે જે બીજા વેરિએબલનું મેમરી એડ્રેસ સ્ટોર કરે છે.
મેમરી લેઆઉટ ચિત્રણ




```c
#include <stdio.h>

int main() {
    int var = 10;   // સામાન્ય વેરએિબલ
    int *ptr;       // પોઈન્ટર વેરએિબલ
```

```c

    ptr = &var;     // ptr માં var નું એડ્રેસ સ્ટોર કરો

    printf("Value of var: %d\n", var);      // આઉટપુટ: 10
    printf("Address of var: %p\n", &var);   // આઉટપુટ: memory address
    printf("Value of ptr: %p\n", ptr);      // આઉટપુટ: same memory address
    printf("Value at address stored in ptr: %d\n", *ptr); // આઉટપુટ: 10

    // પોઈન્ટરનો ઉપયોગ કરીને મૂલ્ય બદલો
    *ptr = 20;
    printf("New value of var: %d\n", var);  // આઉટપુટ: 20

    return 0;
}
```

કોષ્ટક: પોઈન્ટર ઓપરેશન્સ

ઓપરેશન	ચિહ્ન	વિવરણ	ઉદાહરણ
એડ્રેસ-ઓફ	&	વેરિએબલનું એડ્રેસ મેળવે છે	&var
ડિરેફરન્સ	*	એડ્રેસ પરનું મૂલ્ય મેળવે છે	*ptr
ડિક્લેરેશન	*	પોઈન્ટર વેરિએબલ બનાવે છે	int *ptr;
એસાઇનમેન્ટ	=	પોઈન્ટરને એડ્રેસ સોંપે છે	ptr = &var;

- **મેમરી એડ્રેસ**: પોઈન્ટર સ્થાન સ્ટોર કરે છે, મૂલ્ય નહીં
- **ઇનડાયરેક્શન**: એડ્રેસનો ઉપયોગ કરીને પરોક્ષ રીતે મૂલ્ય મેળવે છે
- **મેમરી મેનિપ્યુલેશન**: ડાયનેમિક મેમરી એક્સેસની મંજૂરી આપે છે

મેમરી ટ્રીક

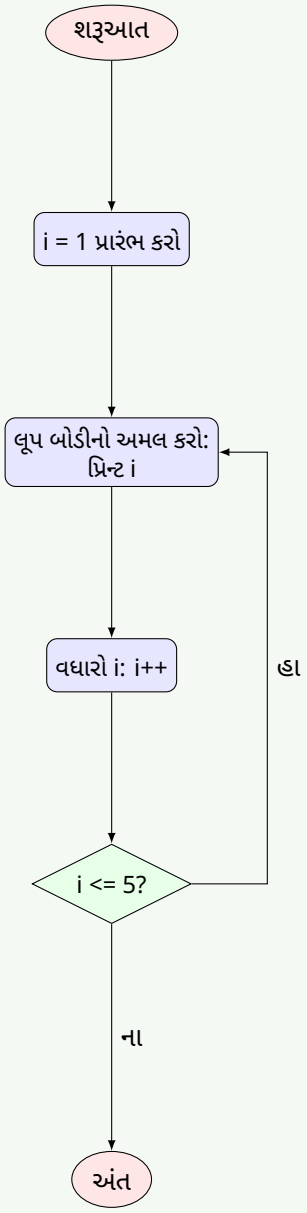
"ADA" (Address Dereferencing Access)

પ્રશ્ન 4 [C ગુણ]

7 ફ્લોચાર્ટ દોરો અને ઉદાહરણ સાથે ડૂ-વાઇલ લૂપ સમજાવો.

જવાબ

do-while લૂપ એ પોસ્ટ-ટેસ્ટ લૂપ છે જે શરત તપાસતા પહેલા ઓછામાં ઓછા એક વખત તેના બોડીને એક્ઝિક્યુટ કરે છે.
ફ્લોચાર્ટ: do-while લૂપ




```c
#include <stdio.h>

int main() {
    int i = 1;

    do {
        printf("%d ", i);
        i++;
    } while (i <= 5);  // પ્રથમ એક્ઝિક્યુશન પછી શરત ચેક થાય છે

    // આઉટપુટ: 1 2 3 4 5

    return 0;
}
```

કોષ્ટક: ડૂ-વાઇલ લૂપની લાક્ષણિકતાઓ

લાક્ષણિકતા	વિવરણ
એક્ઝિક્યુશન ક્રમ	પહેલા બોડી, પછી શરત
ન્યૂનતમ પુનરાવર્તન	ઓછામાં ઓછું એક
શરત ચેક	લૂપના અંતે
સમાપ્તિ	જ્યારે શરત ખોટી થાય ત્યારે
સિન્ટેક્સ	do { statements; } while (condition);

- **પોસ્ટ-ટેસ્ટ લૂપ**: લૂપ બોડી પછી શરત મૂલ્યાંકન
- **ગેરંટેડ એક્ઝિક્યુશન**: લૂપ બોડી હંમેશા ઓછામાં ઓછી એક વખત ચાલે છે
- **સેમિકોલોન**: વાઇલ કન્ડિશન પછી જરૂરી છે

મેમરી ટ્રીક

"DECAT" (Do Execute Check After That)

OR

પ્રશ્ન 4 [a ગુણ]

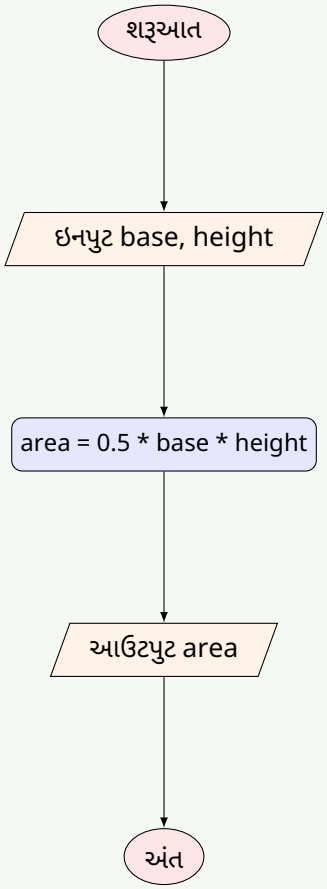
3 ત્રિકોણનું ક્ષેત્રફળ (૦.૫ * પાયો * ઉંચાઈ) શોધવા માટેનો C પ્રોગ્રામ લખો.

જવાબ

આ પ્રોગ્રામ ક્ષેત્રફળ = $0.5 \times base \times height$ સૂત્રનો ઉપયોગ કરીને ત્રિકોણનું ક્ષેત્રફળ ગણે છે.

```c
#include <stdio.h>

int main() {
    float base, height, area;

    printf("Enter base of triangle: ");
    scanf("%f", &base);
    printf("Enter height of triangle: ");
    scanf("%f", &height);

    // ક્ષેત્રફળ ગણો
    area = 0.5 * base * height;

    printf("Area of triangle = %.2f square units\n", area);

    return 0;
}
```

ફ્લોચાર્ટ: ત્રિકોણનું ક્ષેત્રફળ



- **ફોર્મ્યુલા**: ક્ષેત્રફળ = 0.5 * બેઝ * હાઈટ
- **ફ્લોટ વેરિએબલ્સ**: દશાંશ ચોકસાઈ માટે
- **યુઝર ઇનપુટ**: યુઝર પાસેથી પાયો અને ઊંચાઈ મેળવે છે

મેમરી ટ્રીક

"Half-BH" (Half times Base times Height)

પ્રશ્ન 4 [b ગુણ]

4 પોઈન્ટરનું ડીકલેરેશન અને ઈનીશ્યલાઈઝેશન સમજાવો.

જવાબ

પોઈન્ટર ડિક્લેરેશન અને ઇનિશિયલાઇઝેશનમાં પોઈન્ટર વેરિએબલ બનાવવાનો અને તેને મેમરી એડ્રેસ સોંપવાનો સમાવેશ થાય છે.

કોષ્ટક: પોઈન્ટર ડિક્લેરેશન અને ઇનિશિયલાઇઝેશન

ઓપરેશન	સિન્ટેક્સ	ઉદાહરણ	સમજૂતી
ડિક્લેરેશન	type *name;	int *ptr;	int પોઈન્ટર બનાવે છે
ઇનિશિયલાઇઝેશન	name = &var;	ptr = #	num નું એડ્રેસ સોંપે છે
સંયુક્ત	type *n = &v;	int *p = &n;	ડિક્લેર અને ઇનિશિયલાઇઝ
નલ પોઈન્ટર	name = NULL;	ptr = NULL;	કંઈ નહીં બતાવે

```
int main() {
    // ડક્લિરેશન
    int *ptr1;

```

```c
    // ડક્લેરેશન અને ઇનશિયિલાઇઝેશન એકસાથે
    int num = 10;
    int *ptr2 = &num;

    // NULL સાથે ઇનશિયિલાઇઝેશન
    int *ptr3 = NULL;

    printf("Value at address ptr2: %d\n", *ptr2);  // આઉટપુટ: 10

    return 0;
}
```

- **એસ્ટરિસ્ક સિન્ટેક્સ**: * નો ઉપયોગ ડિક્લેરેશનમાં પોઈન્ટર બનાવવા માટે
- **એડ્રેસ ઓપરેટર**: & વેરિએબલનું એડ્રેસ મેળવે છે
- **NULL ઇનિશિયલાઇઝેશન**: વાઈલ્ડ પોઈન્ટર્સ ટાળવા માટે સુરક્ષિત પદ્ધતિ
- **પોઈન્ટર ટાઈપ**: જે ડેટા ટાઈપને પોઈન્ટ કરે છે તેને મેચ કરવું જોઈએ

મેમરી ટ્રીક

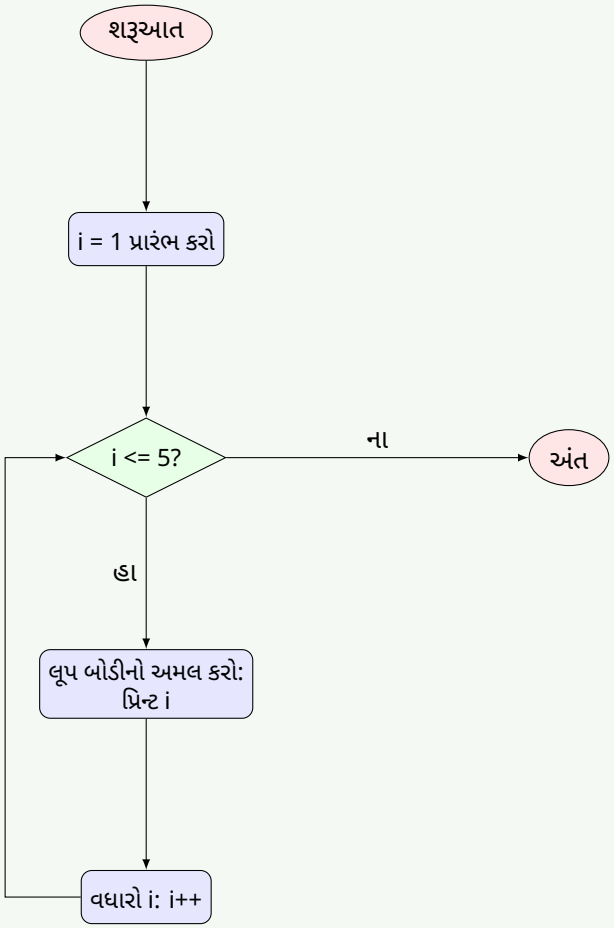
"DINA" (Declare, Initialize with NULL or Address)

પ્રશ્ન 4 [c ગુણ]

7 વાઇલ લૂપ ફ્લોચાર્ટ દોરી ઉદાહરણ સાથે સમજાવો.

જવાબ

વાઇલ લૂપ એક પ્રી-ટેસ્ટ લૂપ છે જે શરત સાચી રહે ત્યાં સુધી તેના બોડીને વારંવાર એક્ઝિક્યુટ કરે છે.
ફ્લોચાર્ટ: while લૂપ




```c
#include <stdio.h>

int main() {
    int i = 1;

    while (i <= 5) {  // દરેક એક્ઝિક્યુશન પહેલા શરત ચેક થાય છે
        printf("%d ", i);
        i++;
    }

    // આઉટપુટ: 1 2 3 4 5

    return 0;
}
```

કોષ્ટક: વાઇલ લૂપની લાક્ષણિકતાઓ

લાક્ષણિકતા	વિવરણ
એક્ઝિક્યુશન ક્રમ	પહેલા શરત, પછી બોડી
ન્યૂનતમ પુનરાવર્તન	શૂન્ય (જો શરત પ્રારંભમાં ખોટી હોય)
શરત ચેક	લૂપની શરૂઆતમાં
સમાપ્તિ	જ્યારે શરત ખોટી થાય ત્યારે
સિન્ટેક્સ	while (condition) { statements; }

- **પ્રી-ટેસ્ટ લૂપ**: લૂપ બોડી પહેલા શરત મૂલ્યાંકન
- **શૂન્ય પુનરાવર્તન શક્ય**: જો શરત પ્રારંભમાં ખોટી હોય તો બોડી ક્યારેય એક્ઝિક્યુટ ન થઈ શકે
- **લૂપ વેરિએબલ**: લૂપ પહેલા ઇનિશિયલાઇઝ થવું જોઈએ

મેમરી ટ્રીક

"CELT" (Check, Execute, Loop, Terminate)

પ્રશ્ન 5 [a ગુણ]

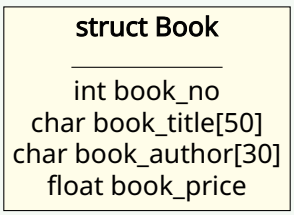
3 બુકની આપેલ માહિતી સ્ટોર કરવાનું સ્ટ્રક્ચર બનાવો: book_no, book_title, book_author, book_price

જવાબ

આ પ્રોગ્રામ સ્પષ્ટ કરેલા ફિલ્ડ્સ સાથે પુસ્તક માહિતી સંગ્રહ કરવા માટે સ્ટ્રક્ચર બનાવે છે.

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

// પુસ્તક માહિતી માટે સ્ટ્રક્ચર ડેફિનિશન
struct Book {
    int book_no;
    char book_title[50];
    char book_author[30];
    float book_price;
};

int main() {
    // Book સ્ટ્રક્ચરના વેરિએબલની ઘોષણા
    struct Book book1;

    // સ્ટ્રક્ચર મેમ્બર્સને મૂલ્યો સોંપો
    book1.book_no = 101;
    strcpy(book1.book_title, "Programming in C");
    strcpy(book1.book_author, "Dennis Ritchie");
    book1.book_price = 450.75;

    // પુસ્તક માહિતી પ્રદર્શિત કરો
    printf("Book No: %d\n", book1.book_no);
    printf("Title: %s\n", book1.book_title);
    printf("Author: %s\n", book1.book_author);
    printf("Price: Rs. %.2f\n", book1.book_price);

    return 0;
}
```

સ્ટ્રક્ચર રજૂઆત:



- **સ્ટ્રક્ચર ડેફિનિશન**: કંપોઝિટ ડેટા પ્રકાર વ્યાખ્યાયિત કરવા struct કીવર્ડનો ઉપયોગ કરે છે
- **મેમ્બર એક્સેસ**: સભ્યો એક્સેસ કરવા માટે ડોટ (.) ઓપરેટરનો ઉપયોગ કરે છે
- **સ્ટ્રિંગ કોપિંગ**: કેરેક્ટર એરે માટે strcpy()

મેમરી ટ્રીક

"NTAP" (Number, Title, Author, Price)

પ્રશ્ન 5 [b ગુણ]

4 (1)sqrt() (2)pow() (3)strlen() (4)strcpy() ફંક્શનો ઉદાહરણ સાથે સમજાવો.

જવાબ

આ ફંક્શન C માં ગણિત ગણતરી અને સ્ટ્રિંગ મેનિપ્યુલેશન માટે વપરાતા સ્ટાન્ડર્ડ લાઇબ્રેરી ફંક્શન્સ છે.

કોષ્ટક: લાઇબ્રેરી ફંક્શન્સ

ફંક્શન	હેડર	હેતુ	ઉદાહરણ
sqrt()	math.h	વર્ગમૂળ	sqrt(16) -> 4.0
pow()	math.h	પાવર	pow(2,3) -> 8.0
strlen()	string.h	સ્ટ્રિંગ લંબાઈ	strlen("Hi") -> 2
strcpy()	string.h	સ્ટ્રિંગ કોપી	strcpy(d, "Hi")

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

int main() {
    // sqrt() અને pow() ઉદાહરણો
    printf("sqrt(25): %.2f\n", sqrt(25));
    printf("pow(2, 4): %.2f\n", pow(2, 4));

    // strlen() ઉદાહરણ
    char str[] = "C Prog";
    printf("Length: %d\n", strlen(str));

    // strcpy() ઉદાહરણ
    char dest[10];
    strcpy(dest, "Hello");
    printf("Copied: %s\n", dest);

    return 0;
}
```

- **મેથ ફંક્શન્સ**: ગાણિતિક ગણતરી માટે sqrt() અને pow()
- **સ્ટ્રિંગ ફંક્શન્સ**: સ્ટ્રિંગ મેનિપ્યુલેશન માટે strlen() અને strcpy()
- **હેડર ફાઈલ્સ**: આ ફંક્શન્સ વાપરવા જરૂરી છે

મેમરી ટ્રીક

"MPSL" (Math Power and String Length)

પ્રશ્ન 5 [c ગુણ]

7 એરે અને એરેનું ઈનીશ્યલાઈઝેશન ઉદાહરણ સાથે સમજાવો.

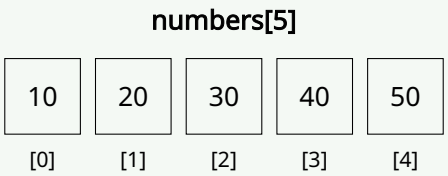
જવાબ

એરે એ એક જ ડેટા પ્રકારના એલિમેન્ટ્સનો સમૂહ છે જે લગાતાર મેમરી લોકેશન્સમાં સ્ટોર થયેલા હોય છે.

કોષ્ટક: એરે પ્રકારો અને ઇનિશિયલાઇઝેશન

પ્રકાર	ડિક્લેરેશન	ઇનિશિયલાઇઝેશન
ઇન્ટીજર	int a[5];	int a[5] = {1,2};
કેરેક્ટર	char s[10];	char s[] = "Hi";
ફ્લોટ	float f[3];	float f[3] = {1.1};
સાઇઝ ઇન્ફરન્સ	-	int n[] = {1,2,3};

ડાયાગ્રામ: એરે મેમરી લેઆઉટ




```c
#include <stdio.h>

int main() {
   int numbers[5] = {10, 20, 30, 40, 50};

   printf("Array elements: ");
   for (int i = 0; i < 5; i++) {
      printf("%d ", numbers[i]);
   }

   return 0;
}
```

- **ઝીરો-બેઝ્ડ ઇન્ડેક્સિંગ**: પ્રથમ એલિમેન્ટ ઇન્ડેક્સ 0 પર
- **કન્ટિગ્યુઅસ મેમરી**: એલિમેન્ટ્સ લાગલાગટ સ્ટોર થાય છે
- **ફિક્સ્ડ સાઇઝ**: સાઇઝ કમ્પાઇલ ટાઇમે નક્કી થાય છે

મેમરી ટ્રીક

"DICE" (Declaration, Initialization, Contiguous storage, Element access)

OR

પ્રશ્ન 5 [a ગુણ]

3 સ્ટ્રક્ચરનું ડીકલેરેશન ઉદાહરણ સાથે સમજાવો.

જવાબ

C માં સ્ટ્રક્ચર ડિક્લેરેશનમાં એક નવો ડેટા પ્રકાર વ્યાખ્યાયિત કરવાનો સમાવેશ થાય છે જે વિવિધ ડેટા પ્રકારોને એક નામ હેઠળ જોડે છે.

કોષ્ટક: સ્ટ્રક્ચર ડિક્લેરેશન પદ્ધતિઓ

પદ્ધતિ	ઉદાહરણ
બેઝિક ડિક્લેરેશન	struct Student { int id; };
વેરિએબલ સાથે	struct Point { int x; } p1;
ટેગ વગર	struct { float r; } c1;
ટાઇપડેફ	typedef struct { int w; } Rect;

```c
struct Student {
    int id;
    char name[30];
    float percentage;
};

int main() {
    struct Student s1;
    s1.id = 101;
    return 0;
}
```

- **સ્ટ્રક્ચર કીવર્ડ**: નવો ડેટા પ્રકાર વ્યાખ્યાયિત કરવા struct નો ઉપયોગ
- **મેમ્બર એક્સેસ**: મેમ્બર્સ એક્સેસ કરવા માટે . (ડોટ) ઓપરેટર
- **વિષમ ડેટા**: વિવિધ ડેટા પ્રકારોને જોડી શકે છે

મેમરી ટ્રીક

"SMUVT" (Structure Mostly Uses Various Types)

પ્રશ્ન 5 [b ગુણ]

4 યુઝર ડીફાઈન ફંક્શન એટલે શું? ઉદાહરણ સાથે સમજાવો.

જવાબ

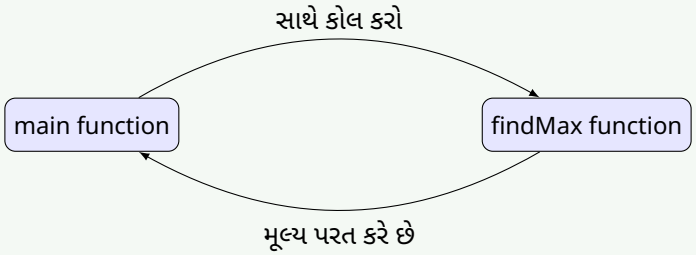
યુઝર-ડિફાઇન્ડ ફંક્શન એ પ્રોગ્રામર દ્વારા લખાયેલો કોડનો બ્લોક છે જે ચોક્કસ કાર્ય કરે છે અને પ્રોગ્રામના અન્ય ભાગોથી કોલ કરી શકાય છે.

કોષ્ટક: ફંક્શન કોમ્પોનન્ટ્સ

કોમ્પોનન્ટ	ઉદાહરણ
રિટર્ન ટાઈપ	int, float, void
ફંક્શન નામ	findMax
પેરામીટર્સ	(int a, int b)
ફંક્શન બોડી	{ return a + b; }
ફંક્શન કોલ	result = findMax(5, 3);

```c
#include <stdio.h>

int findMax(int a, int b);

int main() {
    int max = findMax(10, 20);
    printf("Max: %d\n", max);
    return 0;
}

int findMax(int a, int b) {
    if (a > b) return a;
    else return b;
}
```

ફ્લોચાર્ટ: ફંક્શન કોલ



- **મોડ્યુલર કોડ**: મોટા પ્રોગ્રામને નાના ભાગોમાં વિભાજિત કરે છે
- **રીયુઝેબિલિટી**: ફંક્શનને અનેક વખત કોલ કરી શકાય
- **ડિક્લેરેશન vs ડેફિનિશન**: પ્રોટોટાઇપ vs અમલીકરણ

મેમરી ટ્રીક

"CDRP" (Create, Define, Return, Pass)

પ્રશ્ન 5 [૮ ગુણ]

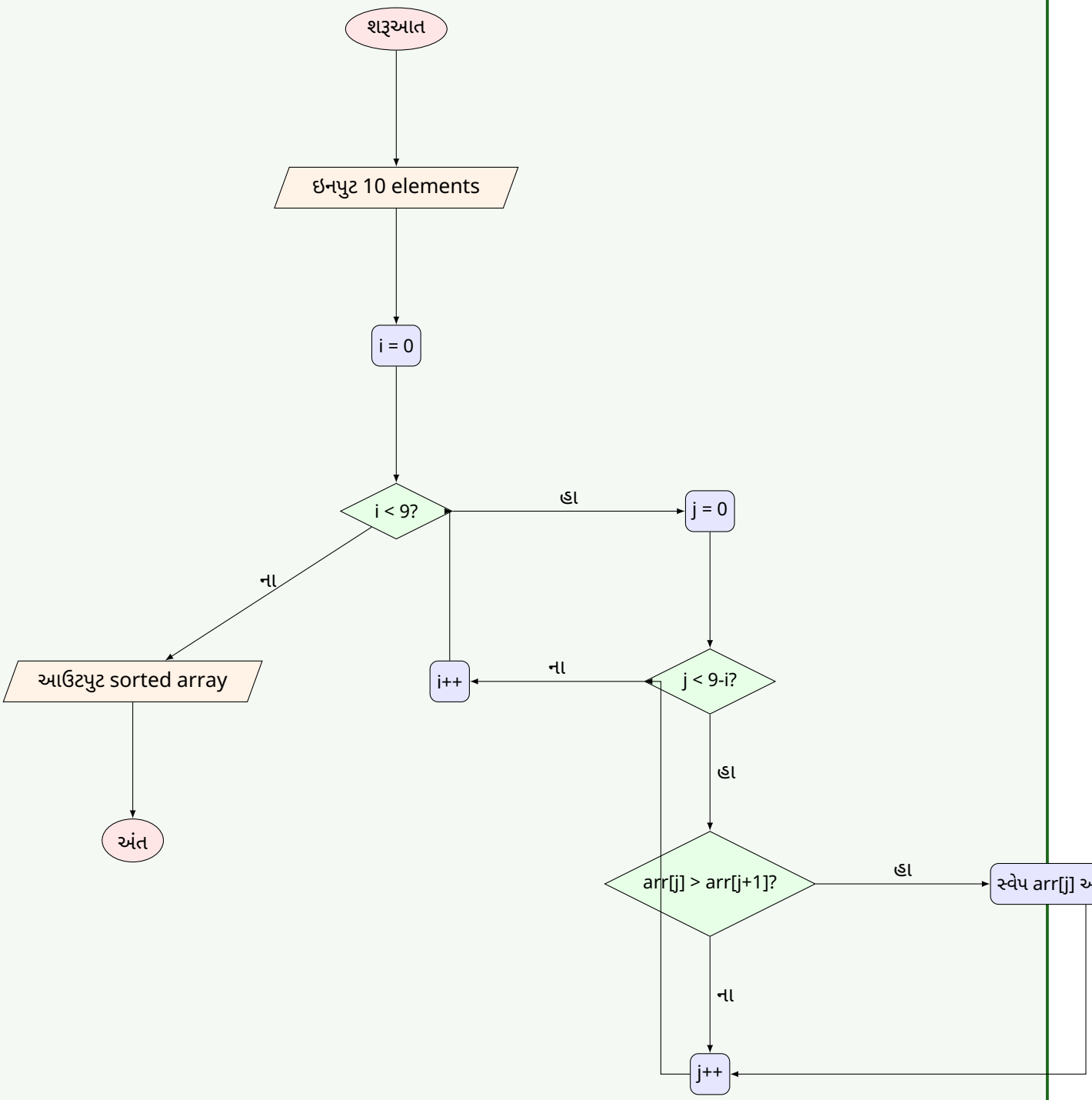
7 ૧૦ નંબરવાળા એરેના ઘટકોને ચઢતા ક્રમમાં ગોઠવવા માટેનો C પ્રોગ્રામ લખો.

જવાબ

આ પ્રોગ્રામ બબલ સોર્ટ અલ્ગોરિધમનો ઉપયોગ કરીને 10 ઇન્ટીજરના એરેને ચઢતા ક્રમમાં સોર્ટ કરે છે.

```c
#include <stdio.h>

int main() {
    int arr[10], i, j, temp;

    // એરે એલમિન્ટ્સ ઇનપુટ કરો
    printf("Enter 10 integers: \n");
    for (i = 0; i < 10; i++) {
        scanf("%d", &arr[i]);
    }

    // ચઢતા ફ્રમ માટે બબલ સોર્ટ અલ્ગોરધિમ
    for (i = 0; i < 9; i++) {
        for (j = 0; j < 9 - i; j++) {
            if (arr[j] > arr[j + 1]) {
                // સ્વેપ કરો
                temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }

    // સોર્ટેડ એરે ડસ્પ્લે કરો
    printf("Array in ascending order: \n");
    for (i = 0; i < 10; i++) {
        printf("%d ", arr[i]);
    }

    return 0;
}
```

ફ્લોચાર્ટ: બબલ સોર્ટ



- **બબલ સોર્ટ**: બાજુના એલિમેન્ટની સરખામણી કરે અને જરૂર હોય તો સ્વેપ કરે
- **નેસ્ટેડ લૂપ્સ**: બહારની લૂપ પાસ માટે, અંદરની તુલના માટે
- **ઓપ્ટિમાઇઝેશન**: દરેક પાસ ઓછામાં ઓછા એક એલિમેન્ટને ફિક્સ કરે છે

મેમરી ટ્રીક

"BSCOT" (Bubble Sort Compares and Orders Things)