

પ્રોગ્રામિંગ ઇન સી (4331105) - વિન્ટર 2024 સોલ્યુશન

Milav Dabgar

May 21, 2024

પ્રશ્ન 1(a) [3 ગુણ]

C લેંગ્વેજના કોઈ પણ છ કીવર્ડ લખો.

જવાબ

કોષ્ટક 1. Keywords Categories

| વર્ગ | કીવર્ડ |
|--------------|------------------|
| ડેટા ટાઈપ્સ | int, float, char |
| કંટ્રોલ ફ્લો | if, for, return |

મેમરી ટ્રીક

"I Find Clever Reasons For Results"

પ્રશ્ન 1(b) [4 ગુણ]

ઓપરેટરની વ્યાખ્યા આપી operand ના આધારે તેના પ્રકાર જણાવો.

જવાબ

ઓપરેટર: એવું ચિહ્ન કે જે ઓપરન્ડ્સ પર કામ કરીને પરિણામ આપે છે.

કોષ્ટક 2. Types of Operators

| પ્રકાર | વિગત | ઉદાહરણ |
|--------|-------------|---------------|
| યુનરી | એક ઓપરન્ડ | ++, --, ! |
| બાયનરી | બે ઓપરન્ડ | +, -, *, /, % |
| ટર્નરી | ત્રણ ઓપરન્ડ | ?: |

મેમરી ટ્રીક

"U-B-T: Use Binary Then Ternary"

પ્રશ્ન 1(c) [7 ગુણ]

ફ્લોચાર્ટની વ્યાખ્યા લખો. ફ્લોચાર્ટના સિમ્બોલ દોરો. બે પૂર્ણાંક નંબર N1 અને N2 માંથી નાનો નંબર શોધવા માટેનો ફ્લોચાર્ટ દોરો.

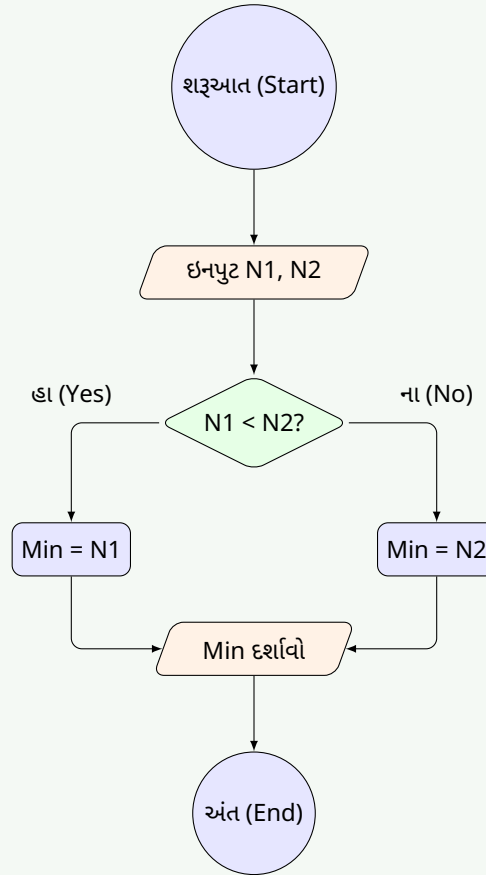
જવાબ

ફ્લોચાર્ટ: અલ્ગોરિથમનું ગ્રાફિકલ નિરૂપણ જેમાં પ્રમાણિત ચિહ્નો દ્વારા ક્રમબદ્ધ ઓપરેશનો દર્શાવવામાં આવે છે.
ફ્લોચાર્ટના સામાન્ય ચિહ્નો:

કોષ્ટક 3. Flowchart Symbols

| ચિહ્ન | અર્થ |
|---------------|--------------|
| ઓવલ | શરૂઆત/અંત |
| પેરાલેલોગ્રામ | ઇનપુટ/આઉટપુટ |
| લંબચોરસ | પ્રક્રિયા |
| ડાયમંડ | નિર્ણય |
| એરો | ફ્લો દિશા |

N1 અને N2 માંથી નાનો નંબર શોધવા માટેનો ફ્લોચાર્ટ:



આકૃતિ 1. Flowchart for Minimum of Two Numbers

મેમરી ટ્રીક

“SPADE: Start-Process-Arrow-Decision-End”

OR

પ્રશ્ન 1(c) [7 ગુણ]

અલ્ગોરિથમની વ્યાખ્યા લખો. વર્તુળનું ક્ષેત્રફળ અને પરિઘ શોધવા માટેનો અલ્ગોરિથમ લખો.

જવાબ

અલગોરિધમ: કોઈ સમસ્યાને ઉકેલવા માટેનું પગલે પગલે પદ્ધતિસરનું કાર્યવાહી, જેમાં સુનિશ્ચિત સંખ્યામાં સુવ્યાખ્યાયિત સૂચનાઓનો ઉપયોગ થાય છે.

વર્તુળનું ક્ષેત્રફળ અને પરિઘ શોધવા માટેનો અલગોરિધમ:

1. શરૂઆત
2. ત્રિજ્યા r ઇનપુટ કરો
3. $PI = 3.14159$ સેટ કરો
4. ક્ષેત્રફળ $= PI \times r \times r$ ગણો
5. પરિઘ $= 2 \times PI \times r$ ગણો
6. ક્ષેત્રફળ અને પરિઘ દર્શાવો
7. અંત

કોષ્ટક 4. Table of formulas used

| માપન | સૂત્ર |
|-----------|-------------------------|
| ક્ષેત્રફળ | $\pi \times r^2$ |
| પરિઘ | $2 \times \pi \times r$ |

મેમરી ટ્રીક

“RICARD: Radius Input, Calculate Area, Reveal Dimensions”

પ્રશ્ન 2(a) [3 ગુણ]

printf() અને scanf() વચ્ચેનો તફાવત સમજાવો.

જવાબ

કોષ્ટક 5. Difference between printf() and scanf()

| લક્ષણ | printf() | scanf() |
|--------------------|----------------------------|--------------------------|
| હેતુ | સ્ક્રીન પર ડેટા આઉટપુટ કરે | કીબોર્ડથી ડેટા ઇનપુટ કરે |
| દિશા | આઉટપુટ ફંક્શન | ઇનપુટ ફંક્શન |
| ફોર્મેટ સ્પેસિફાયર | જરૂરી | જરૂરી |
| પેરામીટર | વાસ્તવિક મૂલ્યો | ચલનું સરનામું (&) |

મેમરી ટ્રીક

“OIAD: Output-Input, Actual-Destination”

પ્રશ્ન 2(b) [4 ગુણ]

1 થી n સુધીનો સરવાળો અને સરેરાશ પ્રિન્ટ કરવા માટે પ્રોગ્રામ C લખો.

જવાબ

Listing 1. Sum and Average Program

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

```

4  int n, i, sum = 0;
5  float avg;
6
7  printf("Enter n: ");
8  scanf("%d", &n);
9
10 for(i = 1; i <= n; i++) {
11     sum += i;
12 }
13
14 avg = (float)sum / n;
15
16 printf("Sum = %d\n", sum);
17 printf("Average = %.2f\n", avg);
18
19 return 0;
20 }

```

મુખ્ય બિંદુઓ:

- ઇનિશિયલાઇઝેશન: sum = 0
- પુનરાવર્તન: 1 થી n સુધી for લૂપ
- ટાઇપ કાસ્ટિંગ: સાચા સરેરાશ માટે (float)

મેમરી ટ્રીક

“SIAP: Sum Initialize, Add in loop, Print results”

પ્રશ્ન 2(c) [7 ગુણ]

એરિથમેટિક ઓપરેટર અને રિલેશનલ ઓપરેટરને ઉદાહરણ સાથે સમજાવો.

જવાબ

1. એરિથમેટિક ઓપરેટર:

કોષ્ટક 6. Arithmetic Operators

| ઓપરેટર | ઓપરેશન | ઉદાહરણ | પરિણામ |
|--------|----------------|--------|--------------|
| + | સરવાળો | 5 + 3 | 8 |
| - | બાદબાકી | 5 - 3 | 2 |
| * | ગુણાકાર | 5 * 3 | 15 |
| / | ભાગાકાર | 5 / 2 | 2 (પૂર્ણાંક) |
| % | મોડ્યુલો (શેષ) | 5 % 2 | 1 |

2. રિલેશનલ ઓપરેટર:

કોષ્ટક 7. Relational Operators

| ઓપરેટર | અર્થ | ઉદાહરણ | પરિણામ |
|--------|-----------------|--------|-----------|
| < | કરતાં નાનું | 5 < 3 | 0 (ખોટું) |
| > | કરતાં મોટું | 5 > 3 | 1 (સાચું) |
| <= | નાનું અથવા સમાન | 5 <= 5 | 1 (સાચું) |
| >= | મોટું અથવા સમાન | 3 >= 5 | 0 (ખોટું) |
| == | સમાન | 5 == 5 | 1 (સાચું) |
| != | અસમાન | 5 != 3 | 1 (સાચું) |

કોડ ઉદાહરણ:

```

1 int a = 5, b = 3;
2 printf("a + b = %d\n", a + b); // Output: 8
3 printf("a > b is %d\n", a > b); // Output: 1 (True)

```

મેમરી ટ્રીક

``ASMDR for Arithmetic, LEGENE for Relational"

OR

પ્રશ્ન 2(a) [3 ગુણ]

get(S) અને scanf("%s",S) વચ્ચેનો તફાવત સમજાવો.

જવાબ

કોષ્ટક 8. Difference between gets(S) and scanf()

| લક્ષણ | gets(S) | scanf("%s",S) |
|-----------------------|------------------------|----------------------------|
| વ્હાઇટસ્પેસ હેન્ડલિંગ | સ્પેસ વાંચે છે | વ્હાઇટસ્પેસ પર બંધ થાય છે |
| બફર ઓવરફ્લો | બાઉન્ડરી ચેક નથી | વિડ્યુ લિમિટ સાથે સલામત |
| રિટર્ન ટાઇપ | char* | વાંચેલા આઇટમની સંખ્યા |
| ઉપયોગ સુરક્ષા | ડેપ્રિકેટેડ, અસુરક્ષિત | ફોર્મેટ કંટ્રોલ સાથે સલામત |

મેમરી ટ્રીક

``WBRU: Whitespace-Boundary-Return-Usage"

OR

પ્રશ્ન 2(b) [4 ગુણ]

બે નંબરોના મૂલ્યની અદલાબદલી (એક્સચેન્જ) કરવા માટે C પ્રોગ્રામ લખો.

જવાબ

Listing 2. Swap Two Numbers

```

1 #include <stdio.h>
2
3 int main() {
4     int a, b, temp;
5
6     printf("Enter two numbers: ");
7     scanf("%d %d", &a, &b);
8
9     printf("Before swap: a = %d, b = %d\n", a, b);
10
11     temp = a;
12     a = b;

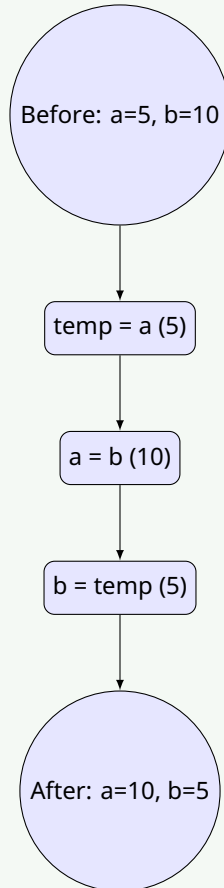
```

```

13  b = temp;
14
15  printf("After swap: a = %d, b = %d\n", a, b);
16
17  return 0;
18  }

```

ડાયાગ્રામ:



આકૃતિ 2. Swapping Logic using Temp Variable

મેમરી ટ્રીક

“TAB: Temp-Assign-Backfill”

OR

પ્રશ્ન 2(c) [7 ગુણ]

બુલિયન ઓપરેટર અને લોજિકલ ઓપરેટર ઉદાહરણ સાથે સમજાવો.

જવાબ

1. બુલિયન ઓપરેટર:

કોષ્ટક 9. Boolean (Bitwise) Operators

| ઓપરેટર | ઓપરેશન | ઉદાહરણ | પરિણામ |
|--------|-------------|--------|--------|
| & | બિટવાઇઝ AND | 5 & 3 | 1 |
| | બિટવાઇઝ OR | 5 3 | 7 |
| ^ | બિટવાઇઝ XOR | 5 ^ 3 | 6 |
| ~ | બિટવાઇઝ NOT | ~5 | -6 |
| << | લેફ્ટ શિફ્ટ | 5 << 1 | 10 |
| >> | રાઇટ શિફ્ટ | 5 >> 1 | 2 |

2. લોજિકલ ઓપરેટર:

કોષ્ટક 10. Logical Operators

| ઓપરેટર | અર્થ | ઉદાહરણ | પરિણામ |
|--------|------------|----------------|-----------|
| && | લોજિકલ AND | (5>3) && (2<4) | 1 (સાચું) |
| | લોજિકલ OR | (5<3) (2<4) | 1 (સાચું) |
| ! | લોજિકલ NOT | !(5>3) | 0 (ખોટું) |

ઉદાહરણ:

```
1 int a = 5, b = 3;
2 printf("a & b = %d\n", a & b); // Output: 1 (bitwise AND)
3 printf("a > b && b < 10 is %d\n", a > b && b < 10); // Output: 1 (true)
```

બિટ રિપ્રેઝન્ટેશન (5 & 3):

```
1 5 = 101
2 3 = 011
3 & = 001 (1 in decimal)
```

મેમરી ટ્રીક

“BOXNRL for Boolean, AON for Logical”

પ્રશ્ન 3(a) [3 ગુણ]

Entry controlled અને exit controlled લૂપની ઉદાહરણ સાથે સરખામણી કરો.

જવાબ

કોષ્ટક 11. Entry vs Exit Controlled Loop

| લક્ષણ | Entry Controlled | Exit Controlled |
|--------------------|--------------------|----------------------------------|
| શરત ચકાસણી | અમલ પહેલા | અમલ પછી |
| ન્યૂનતમ પુનરાવર્તન | શૂન્ય | એક |
| ઉદાહરણ | while, for | do-while |
| ઉપયોગ | પ્રી-ચેક જરૂરી હોય | ઓછામાં ઓછું એક વાર અમલ જરૂરી હોય |

મેમરી ટ્રીક

“BCME: Before-Check-Multiple-Examples”

પ્રશ્ન 3(b) [4 ગુણ]

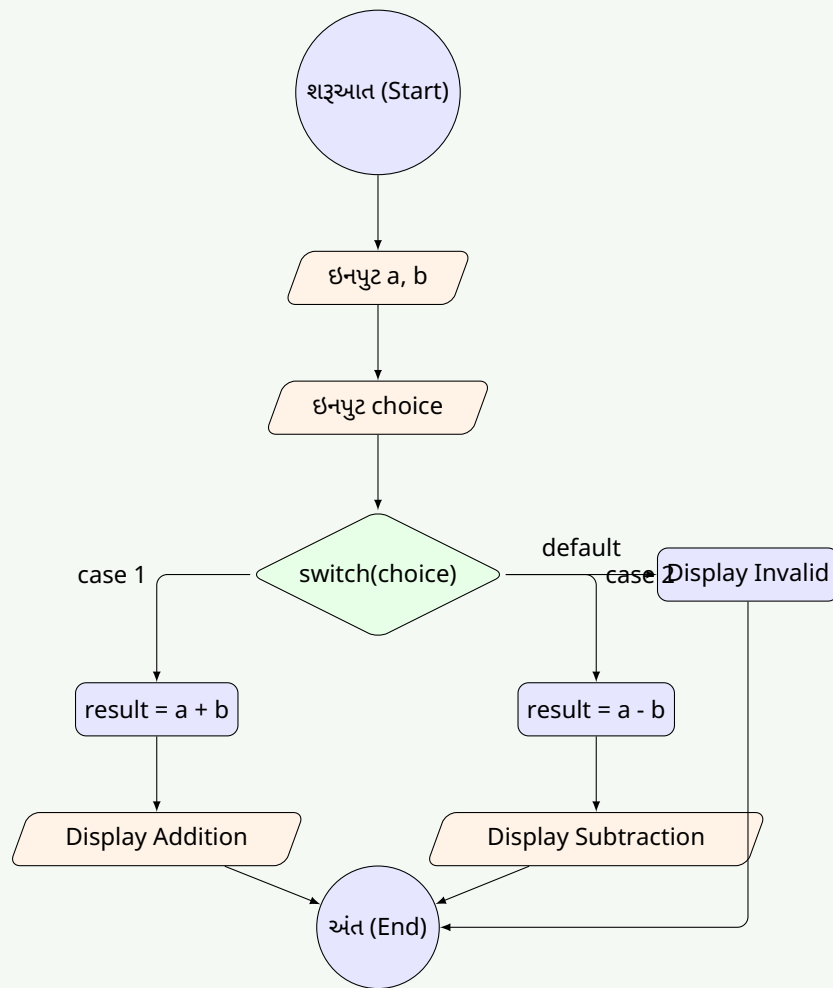
સ્વીચ કેસનો ઉપયોગ કરીને બે નંબરોના સરવાળા અને બાદબાકી દર્શાવવા માટેનો C પ્રોગ્રામ લખો.

જવાબ

Listing 3. Switch Case Calculator

```
1 #include <stdio.h>
2
3 int main() {
4     int a, b, choice, result;
5
6     printf("Enter two numbers: ");
7     scanf("%d %d", &a, &b);
8
9     printf("1. Addition\n2. Subtraction\n");
10    printf("Enter choice (1/2): ");
11    scanf("%d", &choice);
12
13    switch(choice) {
14        case 1:
15            result = a + b;
16            printf("Addition: %d\n", result);
17            break;
18        case 2:
19            result = a - b;
20            printf("Subtraction: %d\n", result);
21            break;
22        default:
23            printf("Invalid choice\n");
24    }
25
26    return 0;
27 }
```

ફોટોગ્રાફ:



આકૃતિ 3. Flowchart for Switch Case Operation

મેમરી ટ્રીક

"CIRCA: Choice-Input-Result-Calculate-Action"

પ્રશ્ન 3(c) [7 ગુણ]

સિન્ટેક્સ, ફ્લોચાર્ટ અને ઉદાહરણ સાથે multiple if-else સ્ટેટમેન્ટ સમજાવો.

જવાબ

multiple if-else સિન્ટેક્સ:

```

1  if (condition1) {
2      // code block 1
3  }
4  else if (condition2) {
5      // code block 2
6  }
7  else if (condition3) {
8      // code block 3
9  }
10 else {

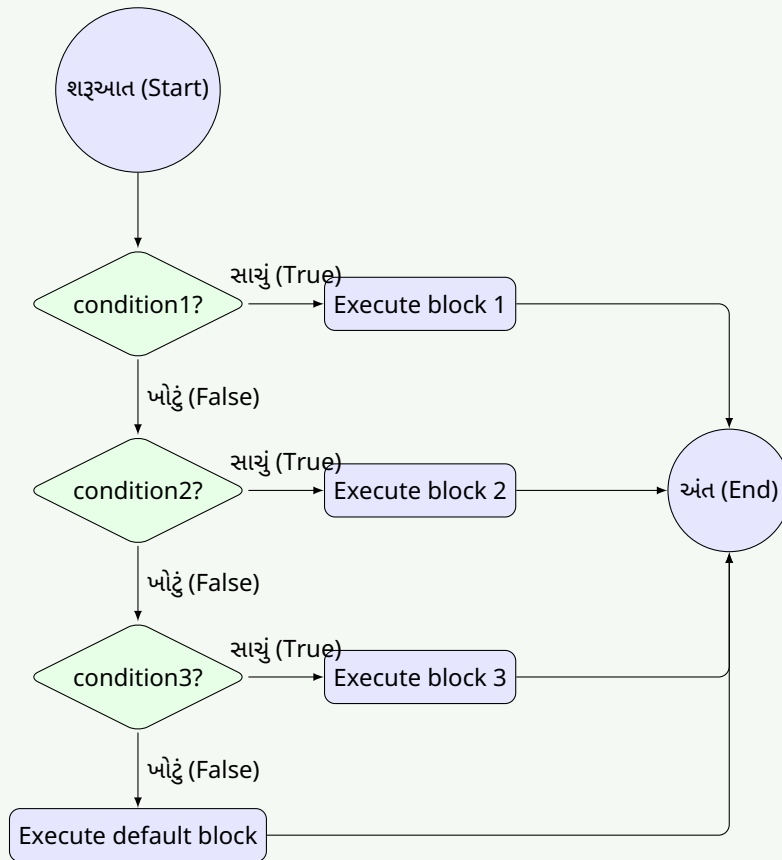
```

```

11 // default code block
12 }

```

ફ્લોચાર્ટ:



આકૃતિ 4. Multiple If-Else Flowchart

ઉદાહરણ:

```

1 #include <stdio.h>
2
3 int main() {
4     int marks;
5
6     printf("Enter marks: ");
7     scanf("%d", &marks);
8
9     if (marks >= 80) {
10         printf("Grade: A\n");
11     }
12     else if (marks >= 70) {
13         printf("Grade: B\n");
14     }
15     else if (marks >= 60) {
16         printf("Grade: C\n");
17     }
18     else {
19         printf("Grade: F\n");
20     }
21
22     return 0;
23 }

```

મેમરી ટ્રીક

``TEST: Try Each Statement Then default"

OR

પ્રશ્ન 3(a) [3 ગુણ]

break અને continue કીવર્ડનો ઉપયોગ જણાવો.

જવાબ

કોષ્ટક 12. Break vs Continue

| કીવર્ડ | હેતુ | અસર | સામાન્ય ઉપયોગ |
|----------|------------------------|---------------------------|---------------------------------|
| break | લૂપ/સ્વિચ ટર્મિનેટ કરે | વર્તમાન લૂપ/સ્વિચ છોડી દે | શરત પૂરી થાય ત્યારે બહાર નીકળવા |
| continue | ઇટરેશન છોડે | આગામી ઇટરેશન પર જાય | ચોક્કસ મૂલ્યો છોડવા |

ઉદાહરણ કોડ:

```

1 // break ઉદાહરણ
2 for(i=1; i<=10; i++) {
3     if(i == 5) break; // i=5 પર લૂપ છોડે
4     printf("%d ", i); // 1 2 3 4 પ્રિન્ટ કરે
5 }
6
7 // continue ઉદાહરણ
8 for(i=1; i<=5; i++) {
9     if(i == 3) continue; // i=3 છોડે
10    printf("%d ", i); // 1 2 4 5 પ્રિન્ટ કરે
11 }

```

મેમરી ટ્રીક

``EXIT-SKIP: EXit IT or SKIP iteration"

OR

પ્રશ્ન 3(b) [4 ગુણ]

આપેલ સંખ્યા એકી છે કે બેકી તે ચકાસવા માટે C પ્રોગ્રામનો લખો.

જવાબ

Listing 4. Even Odd Check

```

1 #include <stdio.h>
2
3 int main() {
4     int num;
5
6     printf("Enter a number: ");
7     scanf("%d", &num);
8 }

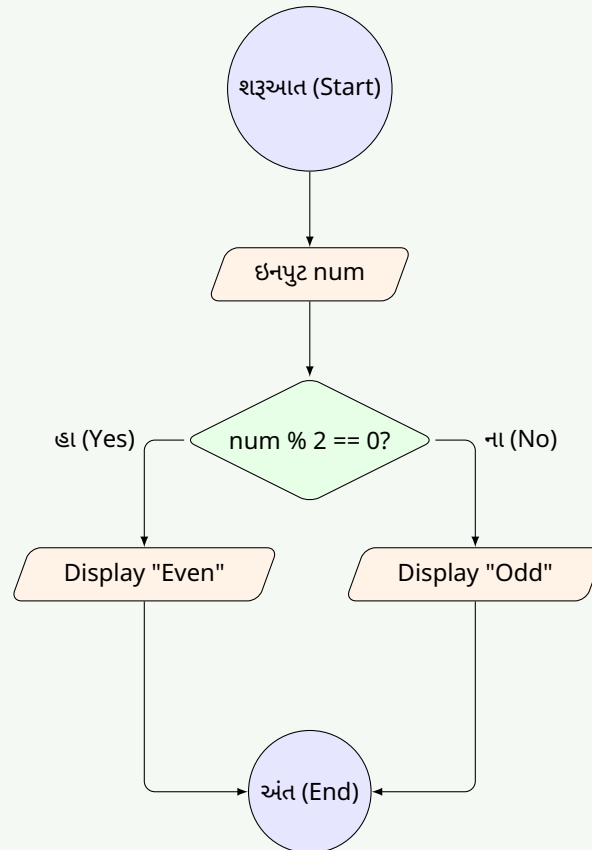
```

```

9  if (num % 2 == 0) {
10     printf("%d is even.\n", num);
11 }
12 else {
13     printf("%d is odd.\n", num);
14 }
15
16 return 0;
17 }

```

ડાયાગ્રામ:



આકૃતિ 5. Flowchart for Even/Odd Check

મુખ્ય બિંદુઓ:

- ચકાસણી: મોડ્યુલો (%) ઓપરેટરનો ઉપયોગ
- નિર્ણય: 2 સાથેના શેષ પર આધારિત
- આઉટપુટ: શેષ 0 હોય તો બેકી, અન્યથા એકી

મેમરી ટ્રીક

``MODE: MODulo Equals zero for even``

OR

પ્રશ્ન 3(c) [7 ગુણ]

સિન્ટેક્સ, ફ્લોચાર્ટ અને ઉદાહરણ સાથે switch-case statement સ્ટેટમેન્ટ સમજાવો.

જવાબ

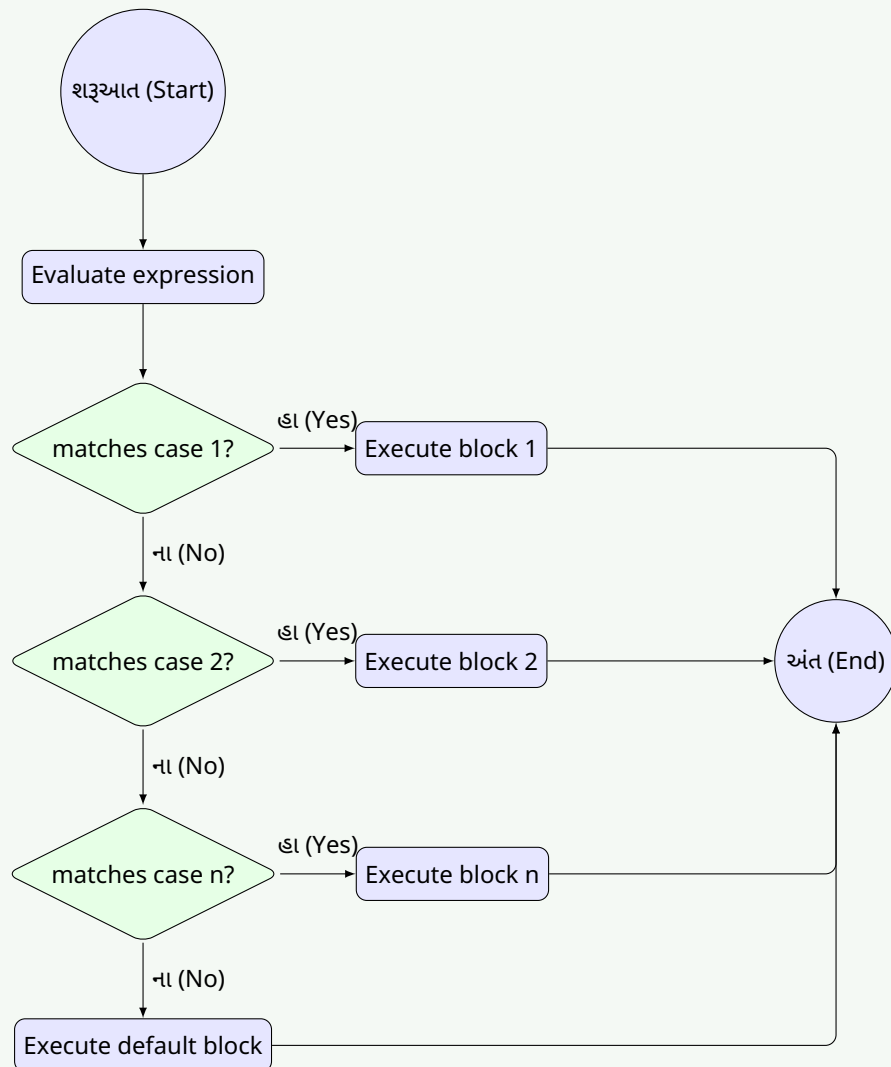
switch-case સિન્ટેક્સ:

```

1 switch (expression) {
2   case constant1:
3     // code block 1
4     break;
5   case constant2:
6     // code block 2
7     break;
8   ...
9   default:
10    // default code block
11 }

```

ફ્લોચાર્ટ:



આકૃતિ 6. Switch Case Flowchart

ઉદાહરણ:

```

1 #include <stdio.h>
2
3 int main() {
4   char grade;

```

```

5
6 printf("Enter grade (A-D): ");
7 scanf(" %c", &grade);
8
9 switch (grade) {
10     case 'A':
11         printf("Excellent!\n");
12         break;
13     case 'B':
14         printf("Good job!\n");
15         break;
16     case 'C':
17         printf("Satisfactory!\n");
18         break;
19     case 'D':
20         printf("Needs improvement!\n");
21         break;
22     default:
23         printf("Invalid grade!\n");
24 }
25
26 return 0;
27 }

```

મેમરી ટ્રીક

“CEBID: Compare-Execute-Break-If-Done”

પ્રશ્ન 4(a) [3 ગુણ]

સ્ટ્રિંગ વ્યાખ્યાયિત કરો. સ્ટ્રિંગ પર કરી શકાય તેવા વિવિધ ઓપરેશન્સની યાદી આપો.

જવાબ

સ્ટ્રિંગ: કેરેક્ટર્સનો એરે જે NULL કેરેક્ટર '\0' થી સમાપ્ત થાય છે.

કોષ્ટક 13. String Operations

| ઓપરેશન | વિગત | ફંક્શન |
|--------------|--|----------------|
| ઇનપુટ/આઉટપુટ | સ્ટ્રિંગ વાંચવી/લખવી | gets(), puts() |
| કોપી | એક સ્ટ્રિંગને બીજી સ્ટ્રિંગમાં કોપી કરવી | strcpy() |
| જોડાણ | બે સ્ટ્રિંગ જોડવી | strcat() |
| સરખામણી | બે સ્ટ્રિંગની સરખામણી | strcmp() |
| લંબાઈ | સ્ટ્રિંગની લંબાઈ શોધવી | strlen() |
| શોધ | સબસ્ટ્રિંગ શોધવી | strstr() |

મેમરી ટ્રીક

“ICCLS: Input-Copy-Concatenate-Length-Search”

પ્રશ્ન 4(b) [4 ગુણ]

અપરકેસ આલ્ફાબેટને લોઅરકેસ આલ્ફાબેટમાં કન્વર્ટ કરવા માટે C પ્રોગ્રામ લખો.

જવાબ

Listing 5. Uppercase to Lowercase

```

1 #include <stdio.h>
2
3 int main() {
4     char ch;
5
6     printf("Enter an uppercase letter: ");
7     scanf("%c", &ch);
8
9     if (ch >= 'A' && ch <= 'Z') {
10         char lowercase = ch + 32; // ASCII difference is 32
11         printf("Lowercase: %c\n", lowercase);
12     }
13     else {
14         printf("Not an uppercase letter\n");
15     }
16
17     return 0;
18 }
```

ASCII ટેબલનો અંશ:

કોષ્ટક 14. ASCII Values

| કેરેક્ટર | ASCII મૂલ્ય |
|----------|-------------|
| A | 65 |
| a | 97 |
| Z | 90 |
| z | 122 |
| તફાવત | 32 |

મેમરી ટ્રીક

``COOL: Character Offset Of Lowercase"

પ્રશ્ન 4(c) [7 ગુણ]

for લૂપ માટેનો ફ્લોચાર્ટ દોરો અને ઉદાહરણ સાથે સમજાવો.

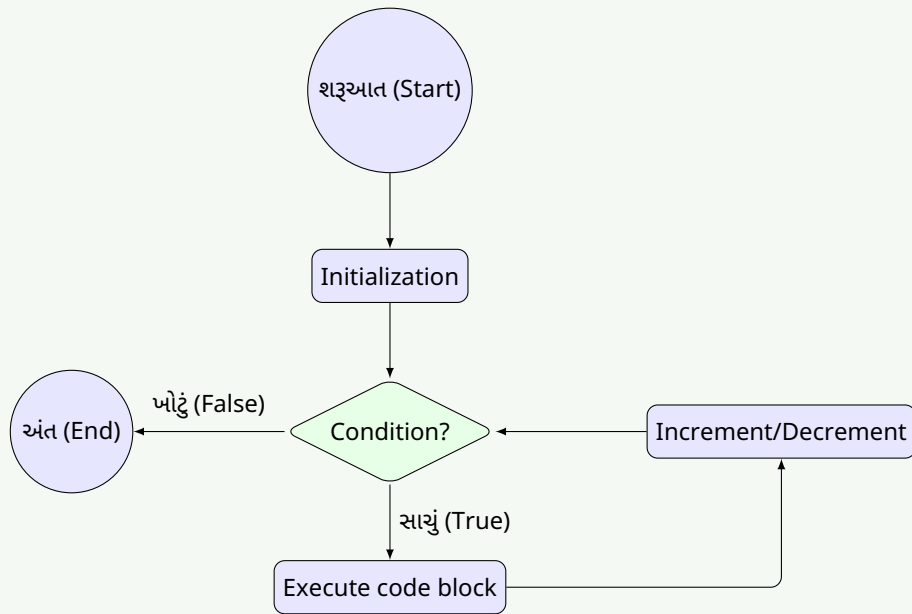
જવાબ

For Loop સિન્ટેક્સ:

```

1 for (initialization; condition; increment/decrement) {
2     // code block
3 }
```

ફ્લોચાર્ટ:



આકૃતિ 7. For Loop Flowchart

For Loop ઘટકો:

1. ઇનિશિયલાઇઝેશન: શરૂઆતમાં એક વખત અમલ
2. શરત: દરેક પુનરાવર્તન પહેલાં ચકાસાય
3. વૃદ્ધિ/ઘટાડો: દરેક પુનરાવર્તન પછી અમલ
4. બોડી: શરત સાચી હોય તો અમલ

ઉદાહરણ:

```

1 #include <stdio.h>
2
3 int main() {
4     int i;
5
6     for (i = 1; i <= 5; i++) {
7         printf("%d ", i);
8     }
9     // Output: 1 2 3 4 5
10
11     return 0;
12 }
  
```

અમલીકરણનો પ્રવાહ:

1. $i = 1$ પ્રારંભિક
2. શરત ચકાસો ($1 \leq 5$) - સાચું
3. બોડી અમલ - 1 પ્રિન્ટ
4. i ને 2 માં વૃદ્ધિ
5. શરત ચકાસો ($2 \leq 5$) - સાચું
6. અને એ રીતે આગળ $i = 6$ થાય ત્યાં સુધી

મેમરી ટ્રીક

“ICE-T: Initialize, Check, Execute, Then increment”

OR

પ્રશ્ન 4(a) [3 ગુણ]

એરે વ્યાખ્યાયિત કરો. એરે પર કરી શકાય તેવા વિવિધ ઓપરેશન્સની યાદી આપો.

જવાબ

એરે: સમાન ડેટા પ્રકારના તત્વો જે ક્રમિક મેમરી સ્થાનોમાં સંગ્રહિત થાય છે.

કોષ્ટક 15. Array Operations

| ઓપરેશન | વિગત | ઉદાહરણ |
|---------------|-----------------------|--------------------|
| ઘોષણા | એરે બનાવો | int arr[5]; |
| ઇનિશિયલાઇઝેશન | મૂલ્ય આપો | arr[0] = 10; |
| ટ્રેવર્સલ | તમામ તત્વો એક્સેસ કરો | for loop |
| ઇન્સર્શન | નવું તત્વ ઉમેરો | arr[pos] = value; |
| ડિલીશન | તત્વ દૂર કરો | તત્વો શિફ્ટ કરો |
| સર્ચિંગ | તત્વ શોધો | લિનિયર/બાઇનરી શોધ |
| સોર્ટિંગ | તત્વો ગોઠવો | બબલ/સિલેક્શન સોર્ટ |

મેમરી ટ્રીક

``DITIDSS: Declare-Initialize-Traverse-Insert-Delete-Search-Sort``

OR

પ્રશ્ન 4(b) [4 ગુણ]

pointer વ્યાખ્યાયિત કરો. ઉદાહરણ સાથે સમજાવો.

જવાબ

પોઇન્ટર: એવું ચલ જે અન્ય ચલના મેમરી સરનામાને સંગ્રહિત કરે છે.

કોષ્ટક 16. Pointer Concepts

| કન્સેપ્ટ | વિગત | સિન્ટેક્સ |
|----------------|----------------------------|------------------|
| ઘોષણા | પોઇન્ટર બનાવો | int *ptr; |
| એડ્રેસ ઓપરેટર | સરનામું મેળવો | &variable |
| ડિરેક્ટરન્સિંગ | સરનામા પર મૂલ્ય એક્સેસ કરો | *ptr |
| એસાઇનમેન્ટ | પોઇન્ટરમાં સરનામું સંગ્રહો | ptr = &variable; |

ઉદાહરણ:

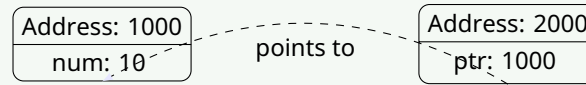
```
1 #include <stdio.h>
2
3 int main() {
4     int num = 10;
5     int *ptr;
6
7     ptr = &num; // num નું સરનામું ptr માં સંગ્રહો
8
9     printf("Value of num: %d\n", num);           // 10
10    printf("Address of num: %p\n", &num);         // num નું સરનામું
11    printf("Value of ptr: %p\n", ptr);             // સમાન સરનામું
12    printf("Value pointed by ptr: %d\n", *ptr);    // 10
```

```

13
14 *ptr = 20; // પોઇન્ટર દ્વારા મૂલ્ય બદલો
15 printf("New value of num: %d\n", num); // 20
16
17 return 0;
18 }

```

ડાયાગ્રામ:



આકૃતિ 8. Pointer Memory Layout

મેમરી ટ્રીક

“SAVD: Store Address, Value through Dereferencing”

OR

પ્રશ્ન 4(c) [7 ગુણ]

while લૂપ માટેનો ફ્લોચાર્ટ દોરો અને ઉદાહરણ સાથે સમજાવો.

જવાબ

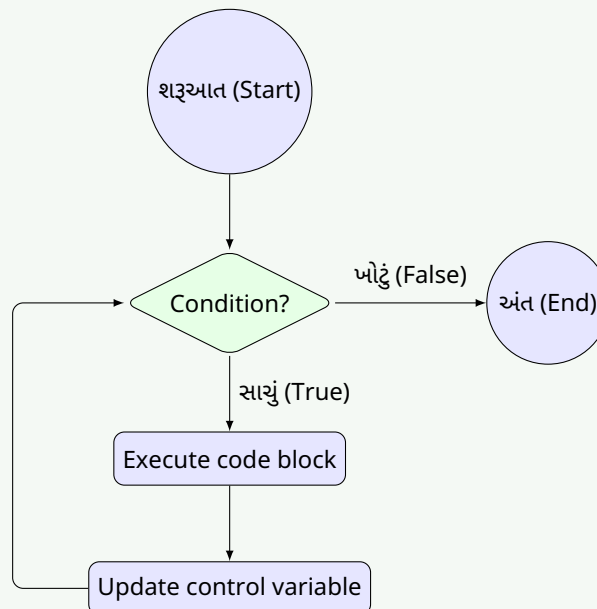
While Loop સિન્ટેક્સ:

```

1 while (condition) {
2   // code block
3 }

```

ફ્લોચાર્ટ:



આકૃતિ 9. While Loop Flowchart

While Loop ઘટકો:

1. ઇનિશિયલાઇઝેશન: લૂપની પહેલાં
 2. શરત: દરેક પુનરાવર્તન પહેલાં ચકાસાય
 3. બોડી: શરત સાચી હોય તો અમલ
 4. અપડેટ: બોડીની અંદર કરવું જરૂરી
- ઉદાહરણ:

```

1 #include <stdio.h>
2
3 int main() {
4     int i = 1;
5
6     while (i <= 5) {
7         printf("%d ", i);
8         i++;
9     }
10    // Output: 1 2 3 4 5
11
12    return 0;
13 }
```

અમલીકરણનો પ્રવાહ:

1. i = 1 પ્રારંભિક (લૂપ પહેલાં)
2. શરત ચકાસો (1 <= 5) - સાચું
3. બોડી અમલ - 1 પ્રિન્ટ
4. i ને 2 માં અપડેટ
5. શરત ચકાસો (2 <= 5) - સાચું
6. અને એ રીતે આગળ i = 6 થાય ત્યાં સુધી

મેમરી ટ્રીક

``CHECK-UPDATE: CHECK before entering, UPDATE before repeating"

પ્રશ્ન 5(a) [3 ગુણ]

નીચેના functions નો ઉપયોગ જણાવો. (1) strcat() (2) strlen() (3) strcpy()

જવાબ

કોષ્ટક 17. String Functions

| ફંક્શન | હેતુ | સિન્ટેક્સ | ઉદાહરણ |
|----------|-------------------------|-------------------|----------------------------------|
| strcat() | સ્ટ્રિંગ્સ જોડે છે | strcat(dest, src) | "Hello" + "World" → "HelloWorld" |
| strlen() | સ્ટ્રિંગની લંબાઈ આપે છે | strlen(str) | "Hello" → 5 |
| strcpy() | સ્ટ્રિંગ કોપી કરે છે | strcpy(dest, src) | src → dest |

કોડ ઉદાહરણ:

```

1 #include <string.h>
2
3 char str1[20] = "Hello";
4 char str2[20] = "World";
5 char str3[20];
6
7 strcat(str1, str2); // str1 "HelloWorld" બને છે
8 int len = strlen(str1); // len 10 બને છે
9 strcpy(str3, str1); // str3 "HelloWorld" બને છે
```

મેમરી ટ્રીક

``CLS: Concatenate-Length-Source copy``

પ્રશ્ન 5(b) [4 ગુણ]

પુસ્તકની માહિતી સંગ્રહિત કરવા માટે એક સ્ટ્રક્ચર બનાવો: book_no, book_title, book_author, book_price.

જવાબ

Listing 6. Book Structure

```

1 #include <stdio.h>
2 #include <string.h>
3
4 struct Book {
5     int book_no;
6     char book_title[50];
7     char book_author[30];
8     float book_price;
9 };
10
11 int main() {
12     struct Book book1;
13
14     // Assign values
15     book1.book_no = 101;
16     strcpy(book1.book_title, "Programming in C");
17     strcpy(book1.book_author, "Dennis Ritchie");
18     book1.book_price = 450.75;
19
20     // Display book information
21     printf("Book No: %d\n", book1.book_no);
22     printf("Title: %s\n", book1.book_title);
23     printf("Author: %s\n", book1.book_author);
24     printf("Price: %.2f\n", book1.book_price);
25
26     return 0;
27 }

```

સ્ટ્રક્ચર મેમરી લેઆઉટ:

| book_no | book_title | book_author | book_price |
|---------|------------|-------------|------------|
| 4 bytes | 50 bytes | 30 bytes | 4 bytes |

આકૃતિ 10. Structure Memory Representation

મેમરી ટ્રીક

``NTAP: Number-Title-Author-Price``

પ્રશ્ન 5(c) [7 ગુણ]

એરે અને એરે initialization સમજાવો. ઉદાહરણ આપો.

જવાબ

એરે: સમાન ડેટા પ્રકારના તત્વોનો સમૂહ જે કમિક મેમરી સ્થાનોમાં સંગ્રહિત થાય છે.
એરે ઇનિશિયલાઇઝેશન પદ્ધતિઓ:

કોષ્ટક 18. Initialization Methods

| પદ્ધતિ | સિન્ટેક્સ | ઉદાહરણ |
|--------------|---------------------------|------------------------------------|
| ધોષણા સમયે | type name[size] = {vals}; | int arr[5] = {10, 20, 30, 40, 50}; |
| આંશિક | type name[size] = {vals}; | int arr[5] = {10, 20}; // બાકીના 0 |
| બધા શૂન્ય | type name[size] = {0}; | int arr[5] = {0}; |
| તત્વ દર તત્વ | name[idx] = val; | arr[0] = 10; arr[1] = 20; |
| સાઇઝ અનુમાન | type name[] = {vals}; | int arr[] = {10, 20}; // સાઇઝ 2 |

ઉદાહરણ:

```

1 #include <stdio.h>
2
3 int main() {
4     // Method 1: Complete initialization
5     int arr1[5] = {10, 20, 30, 40, 50};
6
7     // Method 2: Partial initialization
8     int arr2[5] = {10, 20}; // Remaining are 0
9
10    // Method 4: Size inference
11    int arr4[] = {1, 2, 3, 4, 5};
12
13    // Accessing elements
14    printf("arr1[2] = %d\n", arr1[2]); // Output: 30
15
16    return 0;
17 }
```

મેમરી રિપ્રેઝન્ટેશન:

arr[0] arr[1] arr[2] arr[3] arr[4]

| | | | | |
|----|----|----|----|----|
| 10 | 20 | 30 | 40 | 50 |
|----|----|----|----|----|

આકૃતિ 11. Array Memory Layout

મેમરી ટ્રીક

“CAPES: Complete, Automatic, Partial, Element, Size-inferred”

OR

પ્રશ્ન 5(a) [3 ગુણ]

array અને structure ઉદાહરણ સાથે સરખામણી કરો.

જવાબ

કોષ્ટક 19. Array vs Structure

| લક્ષણ | Array | Structure |
|------------|---------------------------|-------------------------------|
| ડેટા ટાઇપ | સમાન પ્રકારના તત્વો | અલગ પ્રકારના તત્વો |
| એક્સેસ | ઇન્ડેક્સનો ઉપયોગ (arr[i]) | ડોટ ઓપરેટરનો ઉપયોગ (s.member) |
| મેમરી | ક્રમિક, ફિક્સ સાઇઝ | ક્રમિક, પેડિંગ હોઈ શકે |
| એસાઇનમેન્ટ | તત્વ દર તત્વ | સીધું જ સુસંગત સ્ટ્રક્ચર સાથે |
| હેતુ | સમાન આઇટમોનો સંગ્રહ | સંબંધિત ડેટાનો સમૂહ |

ઉદાહરણ:

```

1 // Array
2 int marks[5] = {85, 90, 78, 92, 88};
3 printf("%d", marks[2]);
4
5 // Structure
6 struct Student { int roll; char name[20]; };
7 struct Student s1 = {101, "Raj"};
8 printf("%s", s1.name);

```

મેમરી ટ્રીક

“DAMPA: Datatype-Access-Memory-Purpose-Assignment”

OR

પ્રશ્ન 5(b) [4 ગુણ]

User Defined Function વ્યાખ્યાયિત કરો. ઉદાહરણ સાથે સમજાવો.

જવાબ

User Defined Function: પ્રોગ્રામર દ્વારા લખાયેલ કોડનો બ્લોક જે ચોક્કસ કાર્ય કરે છે અને તેને વારંવાર કોલ કરી શકાય છે.

કોષ્ટક 20. Function Components

| ઘટક | વિગત | ઉદાહરણ |
|-------------|-----------------------|------------------|
| રિટર્ન ટાઇપ | પરત થતા ડેટાનો પ્રકાર | int, float, void |
| ફંક્શન નેમ | અનન્ય ઓળખકર્તા | sum, findMax |
| પેરામીટર્સ | ઇનપુટ ડેટા | (int a, int b) |
| બોડી | સ્ટેટમેન્ટ સમૂહ | { return a+b; } |

ઉદાહરણ:

```

1 #include <stdio.h>
2
3 // Function declaration
4 int sum(int a, int b);
5
6 int main() {
7     int num1 = 10, num2 = 20, result;
8
9     // Function call
10    result = sum(num1, num2);
11
12    printf("Sum = %d\n", result);
13

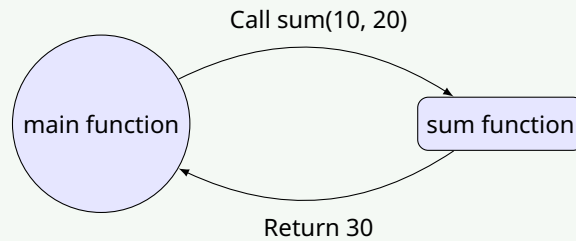
```

```

14     return 0;
15 }
16
17 // Function definition
18 int sum(int a, int b) {
19     return a + b;
20 }

```

ફંક્શન ફ્લો:



આકૃતિ 12. Function Call Flow

મેમરી ટ્રીક

“DPCR: Declaration-Parameters-Call-Return”

OR

પ્રશ્ન 5(c) [7 ગુણ]

એરેમાંથી મહત્તમ અને લઘુત્તમ element શોધવા માટે C પ્રોગ્રામનો લખો.

જવાબ

Listing 7. Min Max Array

```

1  #include <stdio.h>
2
3  int main() {
4      int arr[100], n, i;
5      int max, min;
6
7      printf("Enter number of elements: ");
8      scanf("%d", &n);
9
10     printf("Enter %d integers: ", n);
11     for(i = 0; i < n; i++) {
12         scanf("%d", &arr[i]);
13     }
14
15     // Initialize max and min
16     max = min = arr[0];
17
18     // Find max and min
19     for(i = 1; i < n; i++) {
20         if(arr[i] > max)
21             max = arr[i];
22         if(arr[i] < min)
23             min = arr[i];

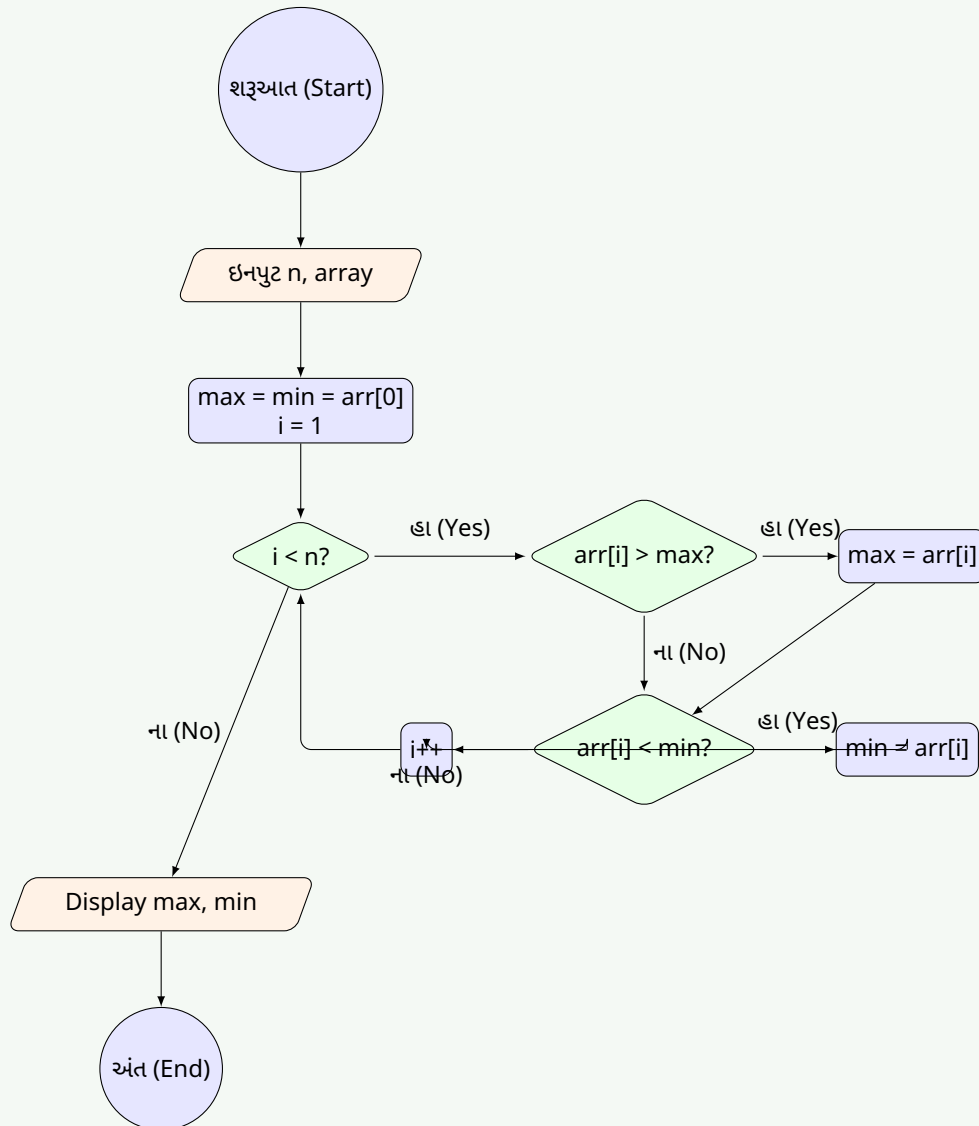
```

```

24 }
25
26 printf("Maximum: %d\n", max);
27 printf("Minimum: %d\n", min);
28
29 return 0;
30 }

```

ફ્લોચાર્ટ:



આકૃતિ 13. Flowchart for Min Max

મેમરી ટ્રીક

“FILLS: First Initialize, Loop through, Look for Small/large”