# C Programming Decision Making and Loops Examples

# \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# If ..Else

## Example #1: Program to Check Even or Odd

1. #include <stdio.h>
2. int main()
3. {
4. int number;
5. printf("Enter an integer: ");
6. scanf("%d", &number);
7. // True if the number is perfectly divisible by 2
8. if(number % 2 == 0)
9. printf("%d is even.", number);
10. else
11. printf("%d is odd.", number);
12. return 0;
13. }

**Output**

Enter an integer: -7

-7 is odd.

In the program, integer entered by the user is stored in variable number.

Then, whether the number is perfectly divisible by 2 or not is checked using modulus operator.

If the number is perfectly divisible by 2, test expression number%2 == 0 evaluates to 1 (true) and the number is even.

However, if the test expression evaluates to 0 (false), the number is odd.

## Example #2: Program to Check Odd or Even Using Conditional Operator

1. #include <stdio.h>
2. int main()
3. {
4. int number;
5. printf("Enter an integer: ");
6. scanf("%d", &number);
7. (number % 2 == 0) ? printf("%d is even.", number) : printf("%d is odd.", number);
8. return 0;
9. }

## Example #3: Program to Check Vowel or consonant

1. #include <stdio.h>
2. int main()
3. {
4. char c;
5. int isLowercaseVowel, isUppercaseVowel;
6. printf("Enter an alphabet: ");
7. scanf("%c",&c);
8. // evaluates to 1 (true) if c is a lowercase vowel
9. isLowercaseVowel = (c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u');
10. // evaluates to 1 (true) if c is an uppercase vowel
11. isUppercaseVowel = (c == 'A' || c == 'E' || c == 'I' || c == 'O' || c == 'U');
12. // evaluates to 1 (true) if either isLowercaseVowel or isUppercaseVowel is true
13. if (isLowercaseVowel || isUppercaseVowel)
14. printf("%c is a vowel.", c);
15. else
16. printf("%c is a consonant.", c);
17. return 0;
18. }

**Output**

Enter an alphabet: G

G is a consonant.

The character entered by the user is stored in variable c.

The isLowerCaseVowel evaluates to 1 (true) if c is a lowercase vowel and 0 (false) for any other character.

Similarly, isUpperCaseVowel evaluates to 1(true) if c is an uppercase vowel and 0 (false) for any other character.

If both isLowercaseVowel and isUppercaseVowel is equal to 0, the test expression evaluates to 0 (false) and the entered character is a consonant.

However, if either isLowercaseVowel or isUppercaseVowel is 1 (true), the test expression evaluates to 1 (true) and the entered character is a vowel.

The program above assumes that the user always enters an alphabet. If the user enters any other character other than an alphabet, the program will not work as intended.

## Example #4 : Largest of 3 Numbers

1. #include <stdio.h>
2. int main()
3. {
4. double n1, n2, n3;
5. printf("Enter three numbers: ");
6. scanf("%lf %lf %lf", &n1, &n2, &n3);
7. if (n1>=n2)
8. {
9. if(n1>=n3)
10. printf("%.2lf is the largest number.", n1);
11. else
12. printf("%.2lf is the largest number.", n3);
13. }
14. else
15. {
16. if(n2>=n3)
17. printf("%.2lf is the largest number.", n2);
18. else
19. printf("%.2lf is the largest number.",n3);
20. }
22. return 0;
23. }

This program uses nested if...else statement to find the largest number.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# For Loop/While and Do While Loop

## Example #1: Sum of Natural Numbers Using for Loop

1. #include <stdio.h>
2. int main()
3. {
4. int n, i, sum = 0;
6. printf("Enter a positive integer: ");
7. scanf("%d",&n);
8. for(i=1; i <= n; ++i)
9. {
10. sum += i; // sum = sum+i;
11. }
12. printf("Sum = %d",sum);
13. return 0;
14. }

The above program takes the input from the user and stores in variable n. Then, for loop is used to calculate the sum upto the given number.

## Example #2: Sum of Natural Numbers Using while Loop

1. #include <stdio.h>
2. int main()
3. {
4. int n, i, sum = 0;
6. printf("Enter a positive integer: ");
7. scanf("%d",&n);
8. i = 1;
9. while ( i <=n )
10. {
11. sum += i;
12. ++i;
13. }
14. printf("Sum = %d",sum);
15. return 0;
16. }

**Output**

Enter a positive integer: 100

Sum = 5050

In both programs, the loop is iterated n number of times. And, in each iteration, the value of i is added to sum and i is incremented by 1.

Though both programs are technically correct, it is better to use for loop in this case. It's because the number of iteration is known.

The above programs doesn't work properly if the user enters a negative integer. Here's a little modification of the above program to take input from the user until positive integer is entered.

## Example #3: Program to Read Input Until User Enters a Positive Integer

1. #include <stdio.h>
2. int main()
3. {
4. int n, i, sum = 0;
5. do {
6. printf("Enter a positive integer: ");
7. scanf("%d",&n);
8. }
9. while (n <= 0);
10. for(i=1; i <= n; ++i)
11. {
12. sum += i; // sum = sum+i;
13. }
14. printf("Sum = %d",sum);
15. return 0;
16. }

## Example #4: Multiplication Table Up to 10

1. #include <stdio.h>
2. int main()
3. {
4. int n, i;
5. printf("Enter an integer: ");
6. scanf("%d",&n);
7. for(i=1; i<=10; ++i)
8. {
9. printf("%d \* %d = %d \n", n, i, n\*i);
10. }
12. return 0;
13. }

**Output**

Enter an integer: 9

9 \* 1 = 9

9 \* 2 = 18

9 \* 3 = 27

9 \* 4 = 36

9 \* 5 = 45

9 \* 6 = 54

9 \* 7 = 63

9 \* 8 = 72

9 \* 9 = 81

9 \* 10 = 90

The Fibonacci sequence is a series where the next term is the sum of pervious two terms. The first two terms of the Fibonacci sequence is 0 followed by 1.

The Fibonacci sequence: 0, 1, 1, 2, 3, 5, 8, 13, 21

## Example #5: Fibonacci Series up to n number of terms

1. #include <stdio.h>
2. int main()
3. {
4. int i, n, t1 = 0, t2 = 1, nextTerm;
5. printf("Enter the number of terms: ");
6. scanf("%d", &n);
7. printf("Fibonacci Series: ");
8. for (i = 1; i <= n; ++i)
9. {
10. printf("%d, ", t1);
11. nextTerm = t1 + t2;
12. t1 = t2;
13. t2 = nextTerm;
14. }
15. return 0;
16. }

**Output**

Enter the number of terms: 10

Fibonacci Series: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34,

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Break and Continue

## Example: Program to Check Prime Number

1. #include <stdio.h>
2. int main()
3. {
4. int n, i, flag = 0;
5. printf("Enter a positive integer: ");
6. scanf("%d", &n);
7. for(i = 2; i <= n/2; ++i)
8. {
9. // condition for nonprime number
10. if(n%i == 0)
11. {
12. flag = 1;
13. break;
14. }
15. }
16. if (n == 1)
17. {
18. printf("1 is neither a prime nor a composite number.");
19. }
20. else
21. {
22. if (flag == 0)
23. printf("%d is a prime number.", n);
24. else
25. printf("%d is not a prime number.", n);
26. }
28. return 0;
29. }

**Output**

Enter a positive integer: 29

29 is a prime number.

If the for loop terminates when the test expression of loop i <= n/2 is false, the entered number is a prime number. The value of flag is equal to 0 in this case.

If the loop terminates because of break statement inside the if statement, the entered number is a nonprime number. The value of flag is 1 in this case.

# Switch case

This program takes an arithmetic operator +, -, \*, / and two operands from the user and performs the calculation on the two operands depending upon the operator entered by the user.

## Example: Simple Calculator using switch Statement

1. // Performs addition, subtraction, multiplication or division depending the input from user
2. # include <stdio.h>
3. int main() {
4. char operator;
5. double firstNumber,secondNumber;
6. printf("Enter an operator (+, -, \*,): ");
7. scanf("%c", &operator);
8. printf("Enter two operands: ");
9. scanf("%lf %lf",&firstNumber, &secondNumber);
10. switch(operator)
11. {
12. case '+':
13. printf("%.1lf + %.1lf = %.1lf",firstNumber, secondNumber, firstNumber + secondNumber);
14. break;
15. case '-':
16. printf("%.1lf - %.1lf = %.1lf",firstNumber, secondNumber, firstNumber - secondNumber);
17. break;
18. case '\*':
19. printf("%.1lf \* %.1lf = %.1lf",firstNumber, secondNumber, firstNumber \* secondNumber);
20. break;
21. case '/':
22. printf("%.1lf / %.1lf = %.1lf",firstNumber, secondNumber, firstNumber / secondNumber);
23. break;
24. // operator doesn't match any case constant (+, -, \*, /)
25. default:
26. printf("Error! operator is not correct");
27. }
29. return 0;
30. }

**Output**

Enter an operator (+, -, \*,): \*

Enter two operands: 1.5

4.5

1.5 \* 4.5 = 6.8