**Practical No 5**

**Aim**

**Decision control structure(if-else, nested if-else)**

* 1. Write C program to find maximum and minimum number between two numbers given by user using if-else and conditional operator.
  2. Write C program to find even or odd number between two numbers given by user using if-else
  3. Write C program to find grade of students using nested if-else

**Theory**

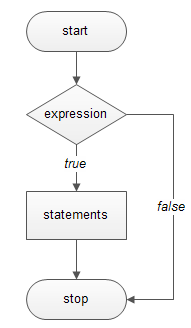
**C *if* statement**

The if statement allows you to control the execution of code based on a  particular condition. The syntax of the if statement is as follows:

|  |  |
| --- | --- |
| 1  2  3 | if(expression){  /\* unit of code to be executed \*/  } |

The body of the *if* statement only executes if the expression evaluates to a non-zero value or *true*. If the expression evaluates to *false*, the statements inside the body of the *if* statement will be ignored.

The following flowchart illustrates the *if* statement:



|  |  |
| --- | --- |
|  |  |

The program asked users to enter a number. If the number is greater than zero, it displays a message showing that otherwise, it does nothing.

**C *if else* statement**

Sometimes you want to execute a piece of code in case of the *expression* in the *if* statement evaluates to *false*. You can use the second form of the if statement which is known as *if else* statement. The following illustrates the syntax of the if else statement:

|  |  |
| --- | --- |
|  | if(expression){    /\* code block of if statement \*/  }else{    /\* code block of else statement \*/  } |

This program asks users to enter a number. It displays the corresponding message based on the input value.

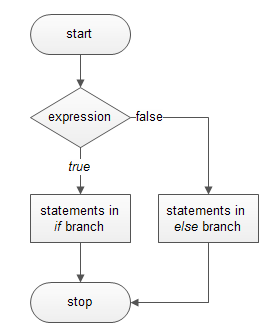
**C *else if* statement**

If you want to use more than one condition, you can use *if else-if* statement. The syntax of the *if else if*statement is as follows:

|  |  |
| --- | --- |
|  | if(condition-1){  /\* code block if condition-1 is true \*/  }else if (condition-2){  /\* code block if condition-2 is true \*/  }else if (condition-3){  /\* code block if condition-3 is true \*/  }else{  /\* code block all conditions above are false \*/  } |

The *else-if* statement is used to make multiway decisions. The conditions in the corresponding *if* and *else if* branch evaluates in sequence from top to bottom. If a condition evaluates to *true*, the statements associated with it executes and terminates the whole chain.

The following flowchart illustrates the *if else* statement:



**Program:**

1. Write C program to find maximum and minimum number between two numbers given by user using if-else and conditional operator.

/\*\*

\* C program to find maximum between two numbers

\*/

#include <stdio.h>

int main()

{

int num1, num2;

/\* Input two numbers from user \*/

printf("Enter two numbers: ");

scanf("%d%d", &num1, &num2);

/\* Compare num1 with num2 \*/

if(num1 > num2)

{

/\* True part means num1 > num2 \*/

printf("%d is maximum", num1);

}

else

{

/\* False part means num1 < num2 \*/

printf("%d is maximum", num2);

}

return 0;

}

/\*\*

\* **C program to find maximum between two numbers using conditional operator**

\*/

#include <stdio.h>

int main()

{

int num1, num2, max;

/\*

\* Input two number from user

\*/

printf("Enter two numbers: ");

scanf("%d%d", &num1, &num2);

/\*

\* If num1 > num2 then

\* assign num1 to max

\* else

\* assign num2 to max

\*/

max = (num1 > num2) ? num1 : num2;

printf("Maximum between %d and %d is %d", num1, num2, max);

return 0;

}

1. Write C program to find even or odd number between two numbers given by user using if-else

#include <stdio.h>

int main()

{

int number;

printf("Enter an integer: ");

scanf("%d", &number);

// True if the number is perfectly divisible by 2

if(number % 2 == 0)

printf("%d is even.", number);

else

printf("%d is odd.", number);

return 0;

}

**Output**

Enter an integer: -7

-7 is odd.

1. Write C program to find grade of students using nested if-else

#**include** <stdio.h>

**int** main()

{

**int** phy, chem, bio, math, comp;

**float** per;

/\* Input marks of five subjects from user \*/

printf("Enter five subjects marks: ");

scanf("%d%d%d%d%d", &phy, &chem, &bio, &math, &comp);

/\* Calculate percentage \*/

per = (phy + chem + bio + math + comp) / 5.0;

printf("Percentage = %.2f\n", per);

/\* Find grade according to the percentage \*/

**if**(per >= 90)

{

printf("Grade A");

}

**else** **if**(per >= 80)

{

printf("Grade B");

}

**else** **if**(per >= 70)

{

printf("Grade C");

}

**else** **if**(per >= 60)

{

printf("Grade D");

}

**else** **if**(per >= 40)

{

printf("Grade E");

}

**else**

{

printf("Grade F");

}

**return** 0;

}

**Practical No 6**

**Aim**

**Loop control structure(While Loop)**

1. Write C program to print number 1 to 20 using while loop.
2. Write C program to print square, cube and factorial of an entered number using while loop.
3. Write C program to check number is palindrome number or not.

**Theory**

**while loop**

The syntax of a while loop is:

while (testExpression)

{

//codes

}

where, testExpression checks the condition is true or false before each loop.

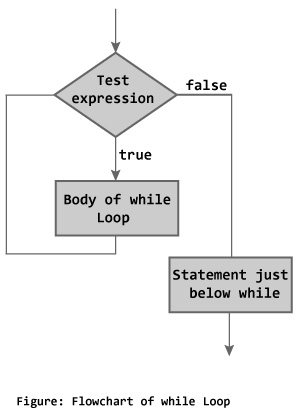
**How while loop works?**

The while loop evaluates the test expression.

If the test expression is true (nonzero), codes inside the body of while loop are exectued. The test expression is evaluated again. The process goes on until the test expression is false.

When the test expression is false, the while loop is terminated.

**Flowchart of while loop**



**Program:**

1. Write C program to print number 1 to 20 using while loop.

#include <stdio.h>

int main () {

/\* local variable definition \*/

int a = 1;

/\* while loop execution \*/

while( a < 20 ) {

printf("value of a: %d\n", a);

a++;

}

return 0;

}

1. Write C program to print square, cube and factorial of an entered number using while loop.

PROGRAM:

#include<stdio.h>

#include<conio.h>

void main()

{

int n,i,f,sqr,cube;

f=i=1;

clrscr();

printf("Enter a Number to Find Factorial: ");

scanf("%d",&n);

sqr=n\*n;

cube=n\*n\*n;

while(i<=n)

{

f\*=i;

i++;

}

printf("The Square of %d is : %d",n,sqr);

printf("The cune of %d is : %d",n,cube);

printf("The Factorial of %d is : %d",n,f);

getch();

}

 OUTPUT:

Enter a Number to Find Factorial: 5

The Factorial of 5 is : 120

1. Program to check a number is Palindrome number or not.

#include <stdio.h>  
int main()  
{  
 int n, reversedInteger = 0, remainder, originalInteger;  
  
 printf("Enter an integer: ");  
 scanf("%d", &n);  
  
 originalInteger = n;  
  
 // reversed integer is stored in variable   
 while( n!=0 )  
 {  
 remainder = n%10;

reversedInteger = reversedInteger\*10 + remainder;

n =n/ 10;

}  
  
 // palindrome if orignalInteger and reversedInteger are equal  
 if (originalInteger == reversedInteger)  
 printf("%d is a palindrome.", originalInteger);  
 else  
 printf("%d is not a palindrome.", originalInteger);  
   
 return 0;  
}

OUTPUT:-

Enter an integer: 121

121 is palindrome

**Practical No 7**

**Aim**

**Loop control structure(Do-While Loop)**

1. Write C program to check number is Armstrong number or not.
2. Write C program to find sum of digits of integer.
3. Write C program to check a number entered by user is perfect number or not.

**Theory**

**Do-While**

Unlike for and while loops, which test the loop condition at the top of the loop, the do...while loop in C programming checks its condition at the bottom of the loop.

A do...while loop is similar to a while loop, except the fact that it is guaranteed to execute at least one time.

**Syntax**

The syntax of a do...while loop in C programming language is −

do {

statement(s);

} while( condition );

Notice that the conditional expression appears at the end of the loop, so the statement(s) in the loop executes once before the condition is tested.

If the condition is true, the flow of control jumps back up to do, and the statement(s) in the loop executes again. This process repeats until the given condition becomes false.

**Flow Diagram**



| BASIS FOR COMPARISON | WHILE | DO-WHILE |
| --- | --- | --- |
| General Form | while ( condition) { statements; //body of loop } | do{ . statements; // body of loop. . } while( Condition ); |
| Controlling Condition | In 'while' loop the controlling condition appears at the start of the loop. | In 'do-while' loop the controlling condition appears at the end of the loop. |
| Iterations | The iterations do not occur if, the condition at the first iteration, appears false. | The iteration occurs at least once even if the condition is false at the first iteration. |

**Difference Between While and Do\_while**

**Program**

1. Write C program to check number is Armstrong number or not.

A positive integer is called an Armstrong number of order n if

abcd... = an + bn + cn + dn + ...

In case of an Armstrong number of 3 digits, the sum of cubes of each digits is equal to the number itself. For example:

153 = 1\*1\*1 + 5\*5\*5 + 3\*3\*3 // 153 is an Armstrong number

#include <stdio.h>  
int main()  
{  
 int number, originalNumber, remainder, result = 0;  
  
 printf("Enter a three digit integer: ");  
 scanf("%d", &number);  
  
 originalNumber = number;

do  
 {  
 remainder = originalNumber%10;   
 result =result+ remainder\*remainder\*remainder;

originalNumber *=* originalNumbe/10;  
 } while (originalNumber != 0);  
  
  
 if(result == number)  
 printf("%d is an Armstrong number.",number);  
 else  
 printf("%d is not an Armstrong number.",number);  
  
 return 0;  
}

1. Write C program to find sum of digits of integer.

#include <stdio.h>

int main()

{

  int Number, Reminder, Sum=0;

  printf("\n Please Enter any number\n");

  scanf("%d", &Number);

do

{

     Reminder = Number % 10;

     Sum = Sum+ Reminder;

     Number = Number / 10;

  }   while(Number > 0);

  printf("\n Sum of the digits of Given Number = %d", Sum);

  return 0;

}

1. Write C program to check a number entered by user is perfect number or not.

#include <stdio.h>

int main()

{

int i, num, sum = 0;

/\* Input a number from user \*/

printf("Enter any number to check perfect number: ");

scanf("%d", &num);

/\* Calculate sum of all proper divisors \*/

for(i=1; i<num; i++)

{ /\* If i is a divisor of num \*/

if(num%i == 0)

{

sum =sum + i;

}

}

/\* Check whether the sum of proper divisors is equal to num \*/

if(sum == num)

{

printf("%d is PERFECT NUMBER", num);

}

else

{ printf("%d is NOT PERFECT NUMBER", num);

}

return 0;

}

**Practical No 8**

**Aim**

**Loop control structure (For Loop)**

1. Write C program to check and generate prime number up to n.
2. Write C program to find GCD of two entered number.
3. Write C program to print the given star patterns.
4. Write C program to print the given number patterns.
5. Write C program to print Floyd’s triangle up to n rows.

**Theory**

A **for** loop is a repetition control structure that allows you to efficiently write a loop that needs to execute a specific number of times.

## Syntax

The syntax of a **for** loop in C programming language is −

for ( init; condition; increment ) {

statement(s);

}

Here is the flow of control in a 'for' loop −

* The **init** step is executed first, and only once. This step allows you to declare and initialize any loop control variables. You are not required to put a statement here, as long as a semicolon appears.
* Next, the **condition** is evaluated. If it is true, the body of the loop is executed. If it is false, the body of the loop does not execute and the flow of control jumps to the next statement just after the 'for' loop.
* After the body of the 'for' loop executes, the flow of control jumps back up to the **increment** statement. This statement allows you to update any loop control variables. This statement can be left blank, as long as a semicolon appears after the condition.
* The condition is now evaluated again. If it is true, the loop executes and the process repeats itself (body of loop, then increment step, and then again condition). After the condition becomes false, the 'for' loop terminates.

## Flow Diagram



## Example

#include <stdio.h>

int main () {

int a;

/\* for loop execution \*/

for( a = 10; a < 20; a = a + 1 ){

printf("value of a: %d\n", a);

}

return 0;

}

When the above code is compiled and executed, it produces the following result −

value of a: 10

value of a: 11

value of a: 12

value of a: 13

value of a: 14

value of a: 15

value of a: 16

value of a: 17

value of a: 18

value of a: 19

**Difference Between For and While Loop**

| BASIS FOR COMPARISON | FOR | WHILE |
| --- | --- | --- |
| Declaration | for(initialization; condition; iteration){ //body of 'for' loop } | while ( condition) { statements; //body of loop } |
| Format | Initialization, condition checking, iteration statement are written at the top of the loop. | Only initialization and condition checking is done at the top of the loop. |
| Use | The 'for' loop used only when we already knew the number of iterations. | The 'while' loop used only when the number of iteration are not exactly known. |
| Condition | If the condition is not put up in 'for' loop, then loop iterates infinite times. | If the condition is not put up in 'while' loop, it provides compilation error. |
| Initialization | In 'for' loop the initialization once done is never repeated. | In while loop if initialization is done during condition checking, then initialization is done each time the loop iterate. |
| Iteration statement | In 'for' loop iteration statement is written at top, hence, executes only after all statements in loop are executed. | In 'while' loop, the iteration statement can be written anywhere in the loop. |

**Program**

1. Write C program to check and generate prime number up to n.

#include<stdio.h>

int main(){

    int num,i,count,n;

    printf("Enter max range: ");

    scanf("%d",&n);

    for(num = 1;num<=n;num++){

         count = 0;

         for(i=2;i<=num/2;i++){

             if(num%i==0){

                 count++;

                 break;

             }

        }

         if(count==0 && num!= 1)

             printf("%d ",num);

    }

   return 0;

}

Sample output:

Enter max range: 50

2 3 5 7 11 13 17 19 23 29 31 37 41 43 47

1. Write C program to find GCD of two entered number.

The HCF or GCD of two integers is the largest integer that can exactly divide both numbers (without a remainder).

#include <stdio.h>

int main()

{

int n1, n2, i, gcd;

printf("Enter two integers: ");

scanf("%d %d", &n1, &n2);

for(i=1; i <= n1 && i <= n2; ++i)

{

// Checks if i is factor of both integers

if(n1%i==0 && n2%i==0)

gcd = i;

}

printf("G.C.D of %d and %d is %d", n1, n2, gcd);

return 0;

Program to print half pyramid using \*

\*

\* \*

\* \* \*

\* \* \* \*

\* \* \* \* \*

Source Code

#include <stdio.h>

int main()

{

int i, j, rows;

printf("Enter number of rows: ");

scanf("%d",&rows);

for(i=1; i<=rows; ++i)

{

for(j=1; j<=i; ++j)

{

printf("\* ");

}

printf("\n");

}

return 0;

}

Program to print half pyramid a using numbers

1

1 2

1 2 3

1 2 3 4

1 2 3 4 5

Source Code

#include <stdio.h>

int main()

{

int i, j, rows;

printf("Enter number of rows: ");

scanf("%d",&rows);

for(i=1; i<=rows; ++i)

{

for(j=1; j<=i; ++j)

{

printf("%d ",j);

}

printf("\n");

}

return 0;

}

Print Floyd's Triangle.

1

2 3

4 5 6

7 8 9 10

Source Code

#include <stdio.h>

int main()

{

int rows, i, j, number= 1;

printf("Enter number of rows: ");

scanf("%d",&rows);

for(i=1; i <= rows; i++)

{

for(j=1; j <= i; ++j)

{

printf("%d ", number);

++number;

}

printf("\n");

}

return 0;

}

\* \* \*

\* \* \* \*

\* \* \* \* \*