CONTROL STATEMENTS

Simple if statement

Syntax:

```
if (condition)
{
 Statement(s);
}
        Statement-x;
```

Explanation:

Here the test expression is evaluated first. If the condition is true the statement(s) are executed and then control is passed to the statement immediately after the **if** statement. If the condition is false, control is passed directly to the statement following the **if** statement.

Program:

```
#include<iostream.h>
#include<conio.h>
main( )
{
 int a,b,big;
 clrscr( );
cin>>a>>b;
if(a>b)
{
  big=a;
}
if(b>a)
{
 big=b;
}
cout<<"Big="<<big;
getch();
}

OUTPUT
       78
       34
Big=78
```

if else statement

Syntax:

```
if(condition)
{
 true block statement(s)
}
else
{
 false block statement(s);
}
statement-x;
```

Explanation

Here the text expression evaluated first. If the condition is true, the true block statement(s) are executed and if the condition is false, the false block statement(s) are executed.

Program:

```
#include<iostream.h>
#include<conio.h>
main()
{
 int a,b,big;
 clrscr();
 cin>>a>>b;
 if(a>b)
 {
  big=a;
 }
else
{
 big=b;
}
cout<<"Big="<<big;
getch();
}

OUTPUT
78
34
Big=78
```

Nested if statements

```
Syntax:
if(condition1)
{
 ..............
 ..............
 if(condition2)
 {
  ............
  ............
  if(condition3)
  {
   ...........
   ...........
  }
  else
  {
   ...........
   ...........
  }
 }
  else
  {
   ..........
```

```
 ……..
 }
}
else
{
 ……..
 ……..
}
```

Here, **condition1** is evaluated first. If it is true, then **condition2** is evaluated otherwise control is passed to the last else statement and all the statements in between are skipped. Similarly result of **condition2** determines whether **condition3** is to be evaluated and which set of statements is to be executed.

Program:

```
#include<iostream.h>
#include<conio.h>
void main()
{
 int a,b,c,big;
 clrscr();
 cin>>a>>b>>c;
 if(a>b)
 {
  if(a>c)
   big=a;
  else
   big=c;
 }
 else
 {
  if(b>c)
   big=b;
  else
   big=c;
 }
 cout<<"Big="<<big;
 getch();
}

OUTPUT
67
89
45
Big=89
```

Else if ladder

```
Syntax
 If(condition1)
 {
  statement1
 }
 else if(condition2)
 {
```

```
  statement2
 }
else if(condition3)
{
 statement3
 }
---------------------
---------------------
else if(condition n)
{
        statement n
}
else
{
 default statement
}
statement-x;
```

The conditions are evaluated from the top downward. As soon as a true condition is found, the statement associated with it is executed and the rest of the ladder is bypassed. If none of the conditions are true, the final else is executed. That is, if all other conditional tests fail, the last else statement is performed. If the final else is not present, no action takes place if all other conditions are false.

Program:

```
#include<iostream.h>
#include<conio.h>
main()
{
 int m1,m2,m3,m4,m5;
 float per;
 cout<<"Enter marks in five subjects";
 cin>>m1>>m2>>m3>>m4>>m5;
 per=(m1+m2+m3+m4+m5)/5;
 if(per>=60)
 {
  cout<<"First division";
 }
 else if(per>=50)
 {
  cout<<"Second division";
 }
 else if(per>=40)
 {
  cout<<"Third division";
 }
 else
 {
  cout<<"Fail";
 }
 getch();
}

OUTPUT
56
89
89
```

```
First division
```

Switch case statement

Syntax

```
switch(expression)
{
 case constant1:
        statement1
        break;
 case constant2:
        statement2
        break;
 case constant3:
        statement3
        break;
 -------------
 -------------
 case constant n:
        statement n
        break;
 default:
        default statement
        break;
}
```

In switch statement, expression is compared with each of the constants **constant1**, **constant2**, **constant3**, and so on. When a match is found, the statements following the case are executed. The break statement is used to break out of the switch statement and pass control to the statement following the switch statement.

Program:

```
#include<iostream.h>
#include<conio.h>
main()
{
 int n;
 clrscr();
 cin>>n;
 switch(n)
 {
  case 1:
   cout<<"The number is one";
   break;
  case 2:
   cout<<"The number is two";
   break;
  case 3:
   cout<<"The number is three";
   break;
  default:
   cout<<"The number is not between 1 and 3";
   break;
 }
```

```
 cout<<endl<<"End of program";
 getch();
}

OUTPUT
2
The number is two
```

While Loop

syntax
while(condition)
{
statement(s)
}

This loop is executed as long as the condition is true. When the condition becomes false control is passed to the statement following the **while** statement.

Program:

```
#include<iostream.h>
#include<conio.h>
main()
{
 int n,i,s=0;
 clrscr();
 cin>>n;
 i=1;
 while(i<=n)
 {
  s=s+i;
  i++;
 }
 cout<<s;
 getch();
}

OUTPUT
10
55
```

do-while loop

do
{
statement(s)
}while(condition);

The **do**-**while** loop is very similar to the while loop. The difference is that in a do-while loop, the condition is evaluated at the end of the loop. This means that these loops are executed at least once. The general format of the statement is

Program:

```
#include<iostream.h>
#include<conio.h>
main()
{
 int n,i,s=0;
 clrscr();
 cin>>n;
 i=1;
 do
 {
  s=s+i;
  i++;
 }while(i<=n);
 cout<<s;
 getch();
}

OUTPUT
10
55
```

For loop

```
for(initialization; condition; increment)
{
 statement(s)
}
```

Initialization

Statement that assigns an initial value to the control variable. The control variable is a variable that is used to keep track of the number of times the loop has to be repeated. This statement is executed once at the beginning of the loop.

Condition

A condition that compares the control variable with a value to find out if the loop has been repeated the required number of times. If the condition is true, the loop is repeated again, If the condition is false, control is passed to statement after the loop. The test is performed at the start of every iteration.

Increment

A statement that increases (or decreases) the value of the control variable by a specified number every time the loop is executed. This statement in executed at the end of every iteration.

Program:

```
#include<iostream.h>
#include<conio.h>
main()
{
```

```
 int n,i,s=0;
 clrscr();
 cin>>n;
 for(i=1;i<=n;i++)
 {
  s=s+i;
 }
 cout<<s;
 getch();
}

OUTPUT
10
55
```

The break statement

The break statement is used to exit out of a loop, by bypassing the normal exit condition. When a break statement is encountered, the execution of the loops stops and control is given to the statement after the loop.

The continue statement

The continue statement is another statement that forces control from a loop. When this statement is executed, all the remaining statements in the body of the loop are ignored and control is passed to the beginning of the loop.

Program:

Write a program to check whether the number is prime or not.

```
#include<iostream.h>
#include<conio.h>
main()
{
 int n,i;
 clrscr();
 cout<<"Enter the number";
 cin>>n;
 for(i=2;i<n;i++)
 {
  if(n%i==0)
   break;
  else
   continue;
 }
 if(i==n)
  cout<<<"Given number is prime";
 else
  cout<<"Given number is not prime";
 getch();
}
OUTPUT
5
Given number is prime
```

goto statement

The **goto** statement is used to alter the program execution sequence by transferring control to some other part of the program.

The general syntax of the **goto** statement is:

goto label;

where label is a valid c identifier used to label the destination such that control could be transferred. There are two ways of using the **goto** statements in a program, namely, as a conditional **goto** and as an unconditional **goto**.

Unconditional goto:

The unconditional **goto** statement is used just to transfer the control from one part of the program to the other part without checking any condition.

Program:

```
#include<iostream.h>
#include<conio.h>
main( )
{
 start:
 cout<<"Welcome to c++"<<endl;
 goto start;
}
```

Conditional goto:

The conditional **goto** is used to transfer the control of the execution from one part of the program to the other in certain conditional cases.

Program:

```
#include<iostream.h>
#include<conio.h>
main()
{
 int n,f=1,i=1;
 clrscr();
 cin>>n;
 fact:
  f=f*i;
  i++;
 if(i<=n)
  goto fact;
 cout<<f;
 getch();
}

OUTPUT
5
```