UNIT -III | INTRODUCTION TO C++ | CHAPTER 10

Flow of Control

1.Define statements.

- A computer program is a set of statements or instructions to perform a specific task.

There are two kinds of statements used in C++.

(i) Null statement
(ii) Compound statement

2.What is Null statement? What is the use of null or empty Statements?

- The " null or empty statement" is a statement containing only a **semicolon (;**).
- Null statements are commonly used as **placeholders** in **iteration statements**.

3.What is Compound statement ?

- C++ allows a **group of statements** enclosed by pair of braces **{}.**
- This **group of statements** is called as a compound statement or a block.

{
Statement 1;
Statement 2;
…
}

4. Differentiate between sequential flow and control flow?

Sequential flow	**control flow**.
The Statements are executed **sequentially**	The statements are executed like branching, Iteration, jumping and function calls

5.What is control statement.?

- Control statements are statements that **alter** the sequence of flow of instructions.
- There are three kinds, They are
  1)Sequence Statement
  2) Selection Statement.
  3)Iteration (Loop) Statement.

6.What is Sequential Statements?

- The sequential statement are executed one after another only once from top to bottom.
- These statements do not alter the flow of execution.
- They are always end with a semicolon (;).

7.What is Selection Statements?

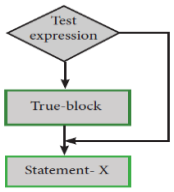
- Here, the statement (s) are executed depends upon a condition.
- If a condition is **true**, a **true block** (a set of statements) is executed otherwise a **false** block is executed.
- This statement is also called **decision** statement.
- Two types of selection statements,
  i) if ii)switch …case

8.What is Iteration statement?

- It is also called **Looping Statement**.
- An **Iteration statement** is a **set of statements** that are **repeatedly** executed **until a condition** is TRUE.
- These statements are also called as **control flow statements.**
- C++ supports three types of iteration statements.
- They are, i) **for** ii) **while** iii) **do…while**

9.Explain if statement with an example.

Syntax:
if (expression)
{
True – Block;
}
Statement x ;



- It is a selection statement
  In **if** statement,
- if the condition is **true** then a **true block-** is executed, otherwise the true-block is **skipped**.
- Statement x is **executed.**

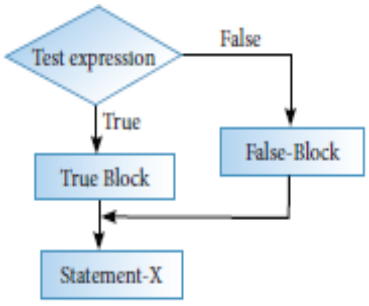
```
#Include<iostream.>
Using namespace std;
int main()
{
clrscr();
int a;
cin>>a;
if(a>=18)
cout<<"Eligible for Vote..";
cout<<"\nGood Bye";
getch();
}
```

Here,
if condition is True(Nonzero), **Eligible for Vote Good Bye** will be Display.
Otherwise control jumps to **Good Bye**.

10.Explain if …else statement with an example.

Syntax:

if (expression)
{
True – Block;
}
else
{
False- block;
}
Statement – x ;



- if the condition is **true** then a **true block-** is executed, **False**-block is **skipped**.
- if the condition is **False** then a **true block-** is **skipped**, **False**-block is **executed**.

```
#Include<iostream>
Using namespace std;
int main()
{
int a
cin>>a;
    if(a>=18)
    cout<<"Eligible for Vote";
    else
    cout<<"Not Eligible for Vote";
cout<<"\nGood Bye";
}
```

Here,if condition is True(Nonzero), **Eligible for Vote Good Bye** will be Display. Otherwise control jumps to **else**and **Not Eligible for Vote Good Bye**.

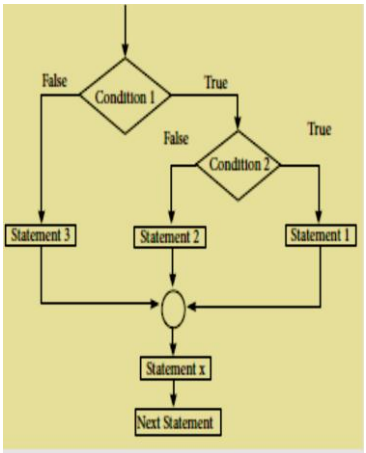
11.Explain Nested if statements with suitable example.

An if statement contains another if statement is called nested if statement. It has three forms.

1.Nested if inside **if part**
2. Nested if inside **else part**
3. Nested if inside **both** if part and else part

1. Nested if inside **if part**

If(condition-1)
{
    if **(condition-2)**
    { True block 2}
    else
    { False block 2}
}
else
{
False block 1;
}

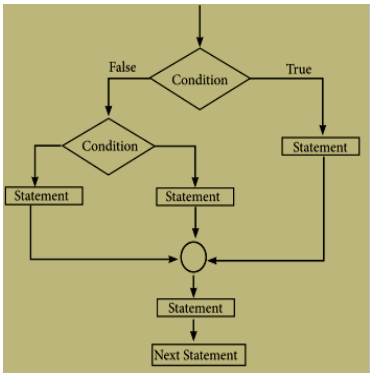


Case 1
If condition 1 & condition 2 are TRUE
True Block 2 will be executed
Case 2
If condition 1 is TRUE and Condition 2 is FALSE
False block 2 will be executed
Case3
If Condition 1 is FALSE
False block 1 will be executed.

2. Nested if inside **else part**

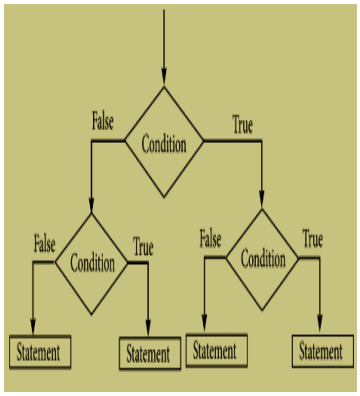
Syntax:

if(condition-1)
{
TRUE block 1;
}
else
{
    if **(condition-2)**
    { True block 2 }
    else
    { False block 2 }
}



3. Nested if inside **both** if part and else part

if(condition-1)
{
    if **(condition-2)**
    { True block 2 }
    else
    { False block 2 }
}
else
{
    if **(condition-3)**
    { True block 3 }
    else
    { False block 3 }
}



12.Explain if -else-if ladder statement with an example.

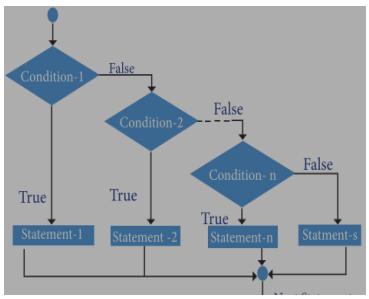
- The if-else ladder is a **multi-path decision** making statement.

In this type of statement,

- **'if'** is followed by one or more **else if** statements and finally end with an **else** statement.

Syntax:

if(**condition-1**)
    { block – 1}
else if (**condition – 2)**
    { block – 2}
else if (**condition – 3)**
    { block – 3}
else
    { default block}



- When the respective Condition becomes **TRUE**, the respective block is executed and **skipped** from ladder.
- If **none** of the conditions is **true**, then the **final else** statement will be executed.

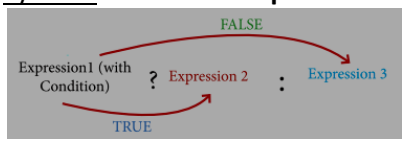
Example.

```
#Include<iostream>
Using namespace std;
int main()
{
int a,b,c;
cin>>a>>b>>c;
if (( a>b)&& (a>c))
  cout<< " A is greater " ;
    else if  (( b>a) &&( b>c))
     cout<< " B is greater " ;
      else
        cout<< " C is greater " ;
}
```

13.Explain conditional operator with an example.

- The conditional operator**(?:)** (or Ternary operator) is an alternative for '**if else** statement'.
- It takes **three** operands.

Syntax: **Condition ? expression 2 : expression 3**



- if the **condition** is **true** (Non-zero),
- then the control is transferred to **expression 2**, otherwise, the control passes to **expression 3**.

Example:

```
#Include<iostream>
Using namespace std;
int main()
{
  int a, b, l;
  cout << "\n Enter any two numbers: ";
  cin >> a >> b;
l = (a>b)? a : b;
  cout << "\n Largest number : " << l;
}
```

14.Explain switch statement with an example.

- The switch statement is a **multi-way** branch statement.
- Based on a condition, the control is transferred to one of the **many possible points**.
- The switch statement replaces **multiple if-else** statement.

Syntax:

```
switch (expression/variable)
{
case 1 : action block 1; break;
case 2 : action block 2; break;
default : action block 3;
}
```

In the above syntax,

- When the expression is evaluated and if its value matches against the **case value**,
- that respective set of statements are **executed**.
- Otherwise, the **default** statements are **executed**.

Example:

```
#Include<iostream>
Using namespace std;
int main()
{
int n;
cin >> n;
        switch (n)
        {
        case 1 : cout << "\nONE"; break;
        case 2 : cout << "\n TWO"; break;
        default : cout << "\nEnter only 1 & 2";
        }
}
```

Working of a program

* If n=1

* case 1 statement will execute directly and terminated by break;

* If switch expression gets **other then the case** value **default** statement will execute

15.Differentiate between switch and if…else statement.

if…else	**switch**
Expression decide whether if block or else block is execute.	Expression decide which case to execute
Uses **multiple** expression for multiple choices.	Uses **single** expression for multiple choices
It checks equality or logical expression	It checks only for expression
It evaluates **integer** ,**char** ,**float**,**pointer** or **Boolean** data type	It evaluate only **char,int and enum** data type
If expression is **false, else** statement will be execute.	If expression is **false, default** statement will be execute.
Difficult to edit	Easy to Edit

16.Explain Nested switch statement.

- When a switch statement contains another switch statement is called as nested switch statement.
- The inner switch and the outer switch constant may or may not be the same.

Syntax

```
switch( Expression1)
{
case 1 : {
        switch(Expression 2)
        case 1:   statements1; break;
        case 2:  statement 2; break;
        ……..
        default : statement;
        }; break:
case 2: statement; break;
…….
default: statement;
}
```

17.Explain an Iteration or Loop statements (or) Define Iteration Statements

- It is also called **Looping Statement**.
- An **Iteration statement** is a **set of statements** that are **repeatedly** executed **until a condition** is TRUE.
- These statements are also called as **control flow statements.**
- C++ supports three types of iteration statements.
- They are, i) **for** ii) **while** iii) **do…while**

18.Explain the Parts of a loop.

Every loop has four elements .They are

- Initialization expression
- Test expression
- Update expression
- The body of the loop

Initialization expression(s):

- The control variable(s) must be **initialized before** enters into loop.
- The initialization expression is executed **only once** in the beginning of the loop.

Test Expression:

- The test expression is an **expression** or **condition.**
- If condition is **TRUE**, the loop-body will be execute. otherwise the loop is terminated.

Update expression:

- It is used to change the value of the control variable.

The body of the loop:

- A statement or set of statements forms a body of the loop that are executed repetitively.

19.What are the difference between Entry controlled loop and Exit controlled loop?.

Entry controlled	Exit controlled
Test-expression is placed at the beginning of the body of the loop	Test-expression is placed at the end of the body of the loop
First the test-expression is evaluated.	First the body of the loop is executed
The body of loop will be executed only when condition is true	The body of loop will be executed **at least** one time
Ex.for , while	do…while

20.Explain for loop with an example

- **It is** an **Entry** controlled loop.
- The **condition** (**Test –Expression)** placed at the **beginning** of the body of the loop.
- for loop contains initialization , test expression and update expression but these are optional.

Syntax:

```
for(Initialization ; test-Expression ; update expression)
        {
        Statements;
        }
Statement – x;
```

General working for loop

1.First the **control variable** is **initialized**
2. Then to **condition**.
3. If the condition is **false**, the control transferred to **statement-x**.
4. If the condition is **true**, the **body of the loop** is executed,
5. Next the control is to **update** expression.
6.After this, the control is again transferred to the **condition**.
7.Next the **steps 3** to **5** is repeated.

Example:

```
#Include<iostream>
Using namespace std;
int main()
)
{
        for(int n=1;n<5;n++)
        {
        cout <<n
        }
}
```

In the above program,

Step 1 .Control Variable **n** gets**1**
Step 2.Next n is compared with **5,**
Step 3 conditions true so **1** is printed on screen
Step 4 n incremented by **1** so n is **2**
Seep **2** to **4** is repeated. Till condition gets false.

21.Why always prefer prefix increment/decrement operator over postfix when to be used alone?

- Because prefix operators are executed faster than postfix.

23.Give an example of infinity loop and empty loop

for(int i=0 ;; i++) - infinity loop
for(; ;) - infinity loop

24.What is an empty loop?

- Empty loop means a loop has no statement in its body is called an empty loop.

Ex.for(int i=0 ; i<4 ; i++);

25.What is the output of the following code?

```
int i;
for(i=0;i<=5;i++);
{ cout<< " We are Indians";  }
```

- It is an infinity loop
- In the above code, the body of a for loop enclosed by braces is not executed.

26.Explain While loop with an example.

- **It is** an Entry controlled loop.
- The **condition** (**Test –Expression)** placed at the **beginning** of the body of the loop.
- A while loop may contain several variations.
- It can be an empty loop or an infinite loop.

Syntax:

```
Initialization;
while ( Test expression )
{
Body of the loop;
Update expression;
}
Statement-x;
```

General working while loop

1.First the **control variable** is **initialized** the **first** time
2. Then to **condition**.
3 . If the condition is **false**, the control transferred to **statement-x**.
4. If the condition is **true**, the **body of the loop** is executed,
5. Next the control is to **update** expression.
6.After this, the control is again transferred to the **condition**.
7.Next the **steps 3** to **5** is repeated

Example:

```
#Include<iostream>
Using namespace std;
int main()
{
int n=1;
        while(n<5)
        {
        cout <<n;
        n++;
        }
}
```

Output

1234

In the above program,
Step 1 .Control Variable **n** gets **1**
Step 2.Next **n** is compared with **5,**
Step 3 condition true so **1** is printed on screen
Step 4 n incremented by **1** so n gets**2**
Seep **2** to **4** is repeated. Till condition gets false

27.Explain do..while loop with an example.

- do.. while is an **Exit controlled** loop,
- The condition placed at the **end** of the body of the loop.
- The body of loop will be executed **at least one** time.

Syntax:

```
Initialization;
        do
        {
        statement;
        update expression;
        } while (condition);
Statement-x;
```

General working do..while loop

1.**First**the **control variable** is **initialized.**
2.The **body of the loop** is executed, and **update** expression.
3.Then to **condition**.
4 . If the condition is **false**, the control transferred to **statement-x**.
4. If the condition is **true**, the **body of the loop** is executed, .

Example

```
#Include<iostream>
Using namespace std;
int main()
{
Int n=1;
do
{ cout <<n
n++;
}while(n <3);
}
getch(); }
```

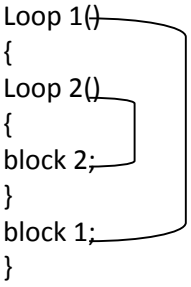
In the above program,

- Control Variable **n** gets **1**
- **1** is printed on screen
- **n** increment by 1
- Next **n** is compared with **3,**
- If it is true ,the body of loop is executed.
- If it is False, exit from the loop.

28.Explain Nesting of Loop with an example.

- A loop which contains another loop is called as a nested loop.
- The **inner loop** must be completely **nested inside** the body of the **outer loop**.
- An outer loop and inner loop **cannot** have the **same control variable**, as it will lead to **logical errors**.

Syntax:




```
for(int i = 1; i <= 3; i++)
{
int j = 1;
while (j <= i)
{
cout << "* ";
j++;
}
cout << '\n';
}
```

working of the above program:

The iterations of the nested loops are as follows;

For loop	While loop
I=1	Is executed once(j<=1)
I=2	Is executed twice (j=1,2)
I=3	Is executed thrice(j=1,2,3)

28A)Define jump statements

- Jump statements are used to interrupt the normal flow of program. Types of Jump Statements are
  - **goto** statement
  - **break** statement
  - **continue** statement

29.Explain goto statement with an example (or) explain unconditional statement in C++.

- The **goto** statement is a control statement.
- It is an **unconditional statement.**
- It is used to transfer the control from one place to another place **without any condition** in a program.

Syntax:

goto label;
……
……
……
Label:

Syntax

Label:
…..
……..
goto Label;

Here ,

- Label is an identifier.
- When **goto label;** is encountered, the control of program jumps to **label:** and executes the code below it.

Example:

```
#Include<iostream>
Using namespace std;
int main()
{
int a,b;
cin>>a>>b;
if(a>b)
        goto true;
else
        cout<<"B is greater";
```

Break	Continue
Break is used to exit the current loop only .	continue statement forces the loop to execute the next iteration following statements will be skipped
Break is used with loops as well as switch case.	Continue is only used in loops, it is not used in switch case.

```
        true:
cout<<"A is greater";
getch()}
```

30.Difference between Break and Continue

Answers to all the questions (2 Marks):

1. What is a null statement and compound statement? **1**
2. What is selection statement? write it's types?**7**
3. Correct the following code segment:

```
 if (x=1)
   p= 100;
 else
   p = 10;
```

correction : (x==1)

4. What will be the output of the following code:

```
 int year;
 cin >> year;
   if (year % 100 == 0)
     if ( year % 400 == 0)
       cout << "Leap";
   else
     cout << "Not Leap year";
```

If the input given is (i) 2000 (ii) 2003 (iii) 2010?

5. What is the output of the following code?

```
 for (int i=2; i<=10 ; i+=2)
   cout << i;
```

6. Write a for loop that displays the number from 21 to 30.
7. Write a while loop that displays numbers 2, 4, 6, 8.....20.
8. Compare an if and a ? : operator.