Unit – III
Control Statements

Contents

VIGNAN'S
Foundation for Science, Technology & Research
(Deemed to be UNIVERSITY)
-Estd. u/s 3 of UGC Act 1956

Statements & Blocks

Statement:

In C an expression is called a statement if it is terminated by semicolon.

Ex:

```
        x=0;        (or)   i++;
        printf("Hello World!");
```

Block:

Braces { and } are used to group the several statements is called compound statement or block.

Ex:

```
        {       x=0;
                printf("Hello World!");
        }
```

Control Statements

- ✓ The order in which the program statements are executed is known as '**Flow of Program control**'.
- ✓ In general, program control flows **sequentially from top to bottom**.
- ✓ Many practical situations like decision making, repetitive execution of a certain task, etc. require deviation from the default flow of program control. This can be altered by using:
  - ❖Branching Statements
    - ✓Selection Statements
    - ✓Jump Statements
  - ❖Iteration Statements

Control Statements

Branching statements are used to transfer the program control from one point to another.

- ✓ **Conditional Branching:**
  - ➢ It is also called as Selection Statements.
  - ➢ Program control is transferred from one point to another based upon the outcome of a certain condition.

    Ex: if statement, if – else statement, switch statement.
- ✓ **Unconditional Branching:**
  - ➢ It is also called as Jumping, program control is transferred from one point to another without checking any condition.

    Ex: goto statement, break statement, continue statement and return statement

if-else Statements

The if-else statement in C is used to perform the operations based on some specific condition.

The operations specified in if block are executed if and only if the given condition is true.

There are the following variants of if statement in C language.

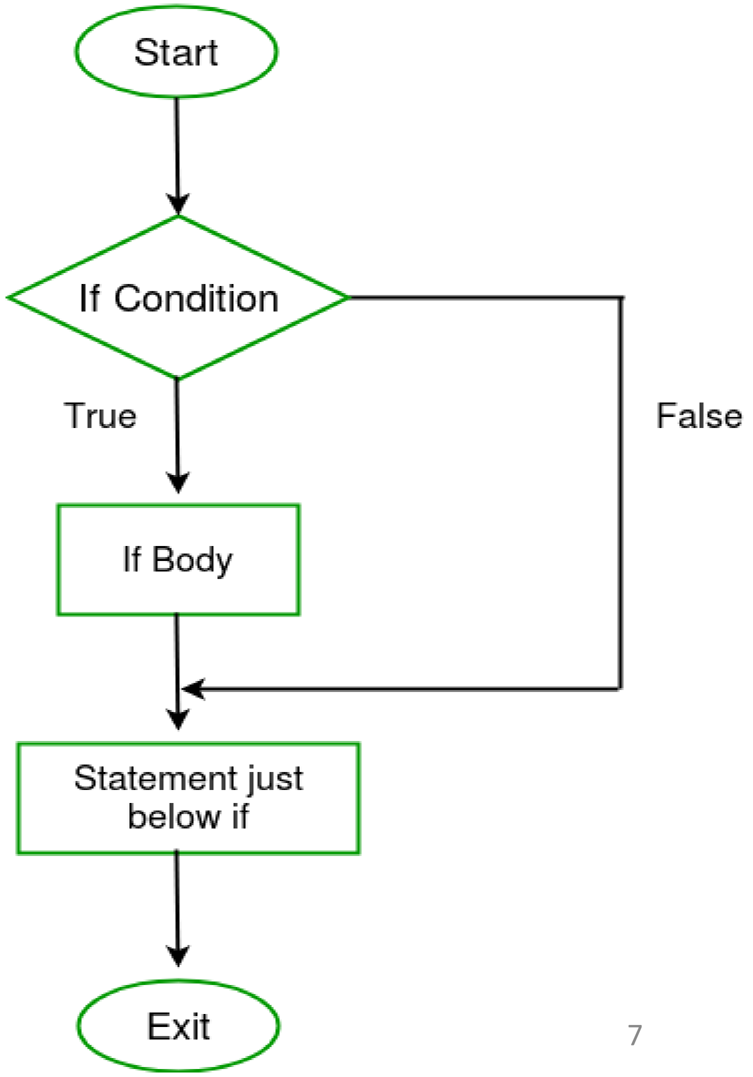
- ✓ If statement
- ✓ If-else statement
- ✓ If else-if ladder
- ✓ Nested if

if Statemens

The if statement is used to check some given condition and perform some operations depending upon the correctness of that condition.

It is mostly used in the scenario where we need to perform the different operations for the different conditions.

```
if(expression){
   //code to be executed
   }
```



if Statemens

```
#include<stdio.h>
main()
{
    int number=0;
    printf("Enter a number:");
    scanf("%d",&number);
    if(number%2==0)
        printf("%d is even number",number);
}
```

```
#include <stdio.h>
main()
{
    int a=5, b=10;
    if(a>b&&a>10)
    {
        printf("a is greater than 10");
        printf("a is greater than b");
    }
}
```

If-else Statements

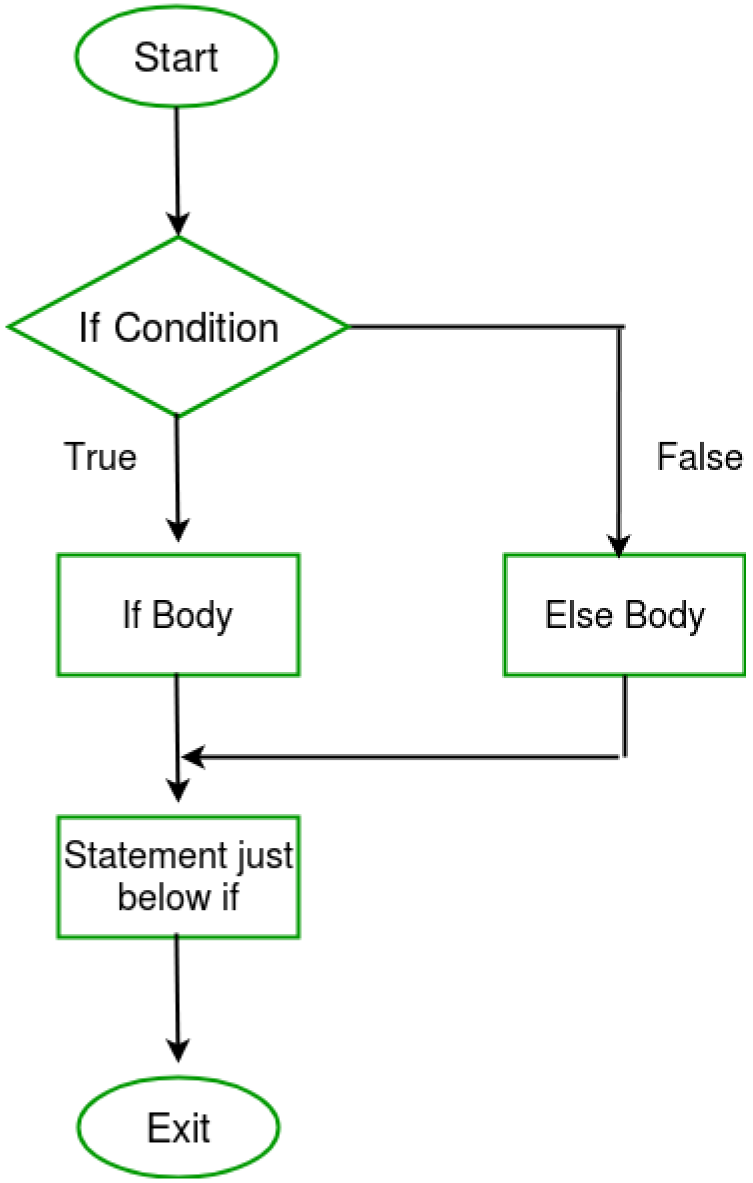
VIGNAN'S Foundation for Science, Technology & Research (Deemed to be UNIVERSITY) -Estd. u/s 3 of UGC Act 1956

- ✓ The if statement alone tells us that if a condition is true it will execute a block of statements and if the condition is false it won't.
- ✓ But what to do something else if the condition is false.
- ✓ Here comes the C else statement.
- ✓ Else statement with if statement to execute a block of code when the condition is false.

```
if (condition)
{
    // Executes this block if
    // condition is true
}
else
{
    // Executes this block if
    // condition is false
}
```

If-else Statements

VIGNAN'S Foundation for Science, Technology & Research (Deemed to be UNIVERSITY) -Estd. u/s 3 of UGC Act 1956



```
#include <stdio.h>
main()
{
    int i = 20;
    if (i < 15)
            printf("i is smaller than 15");
    else
            printf("i is greater than 15");
}
```

If-else Statements

```
#include<stdio.h>
main()
{
    int a=11;
    if(a>10)
        printf("The value of a is %d",a);
    printf("Value a is greater than 10");
    else
        printf("Value a is less than 10");
}
```

O/P:

Compilation error “Misplaced else in function main”

If-else Statements

Check whether the given number is even or not

```
#include <stdio.h>
main()
{
    int number;
    printf("Enter an integer: ");
    scanf("%d", &number);
    if(number % 2 == 0)
        printf("%d is even.", number);
    else
        printf("%d is odd.", number);
}
```

If-else Statements

Check the eligibility for voting using if-else.

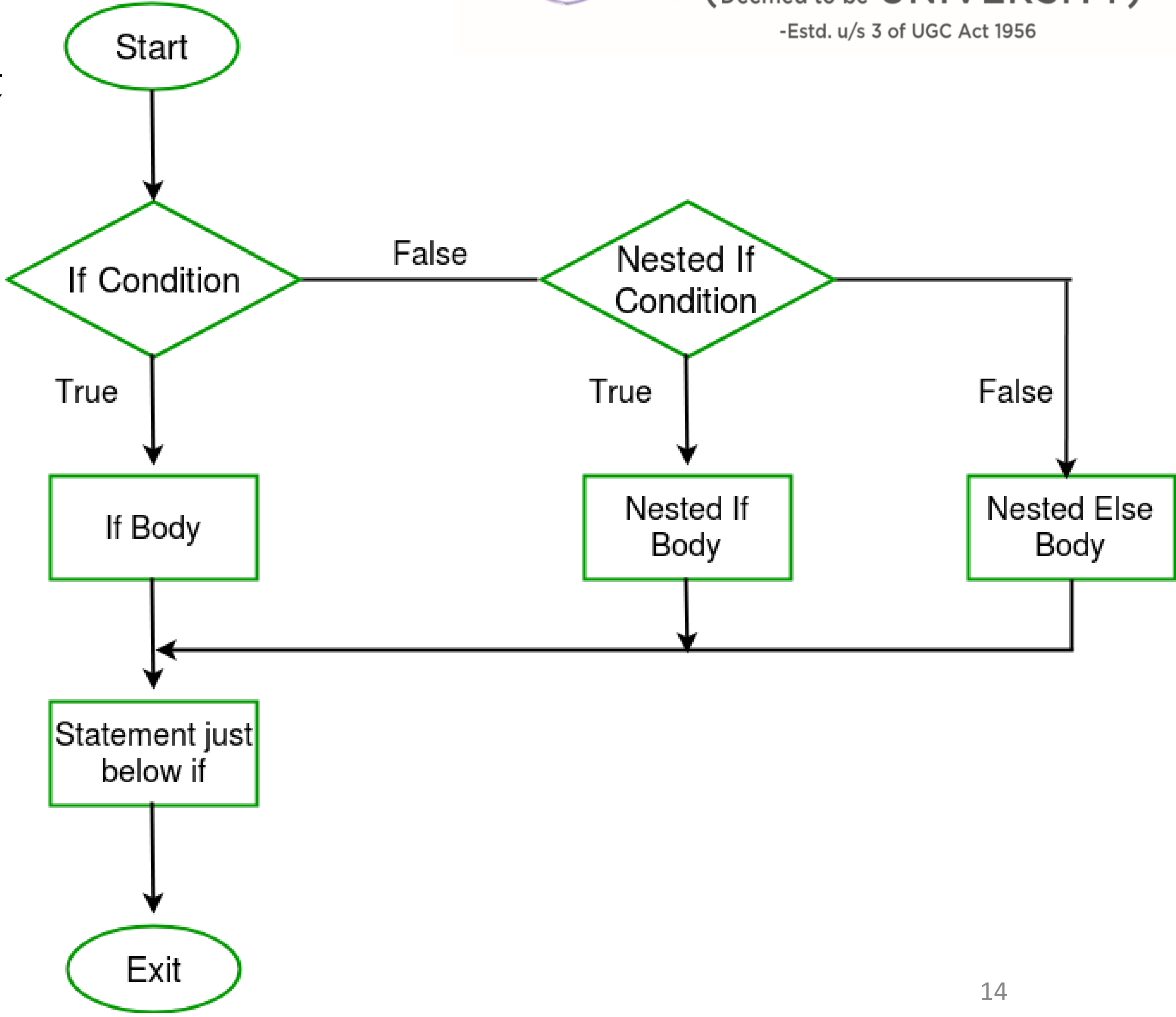
```
#include<stdio.h>
main()
{
    int a;
    printf("Enter the age of the person:");
    scanf("%d", &a);
    if(a>=18)
        printf("Eligible for voting");

    else
        printf("Not eligible for voting");
}
```

nested-if Statements

VIGNAN'S Foundation for Science, Technology & Research (Deemed to be UNIVERSITY) -Estd. u/s 3 of UGC Act 1956

- ✓ A nested if in C is an if statement that is the target of another if statement.
- ✓ Nested if statements mean an if statement inside another if statement.

```
if (condition1)
{
   // Executes when condition1 is true
   if (condition2)
   {
      // Executes when condition2 is true
   }
}
```



Nested If Statements

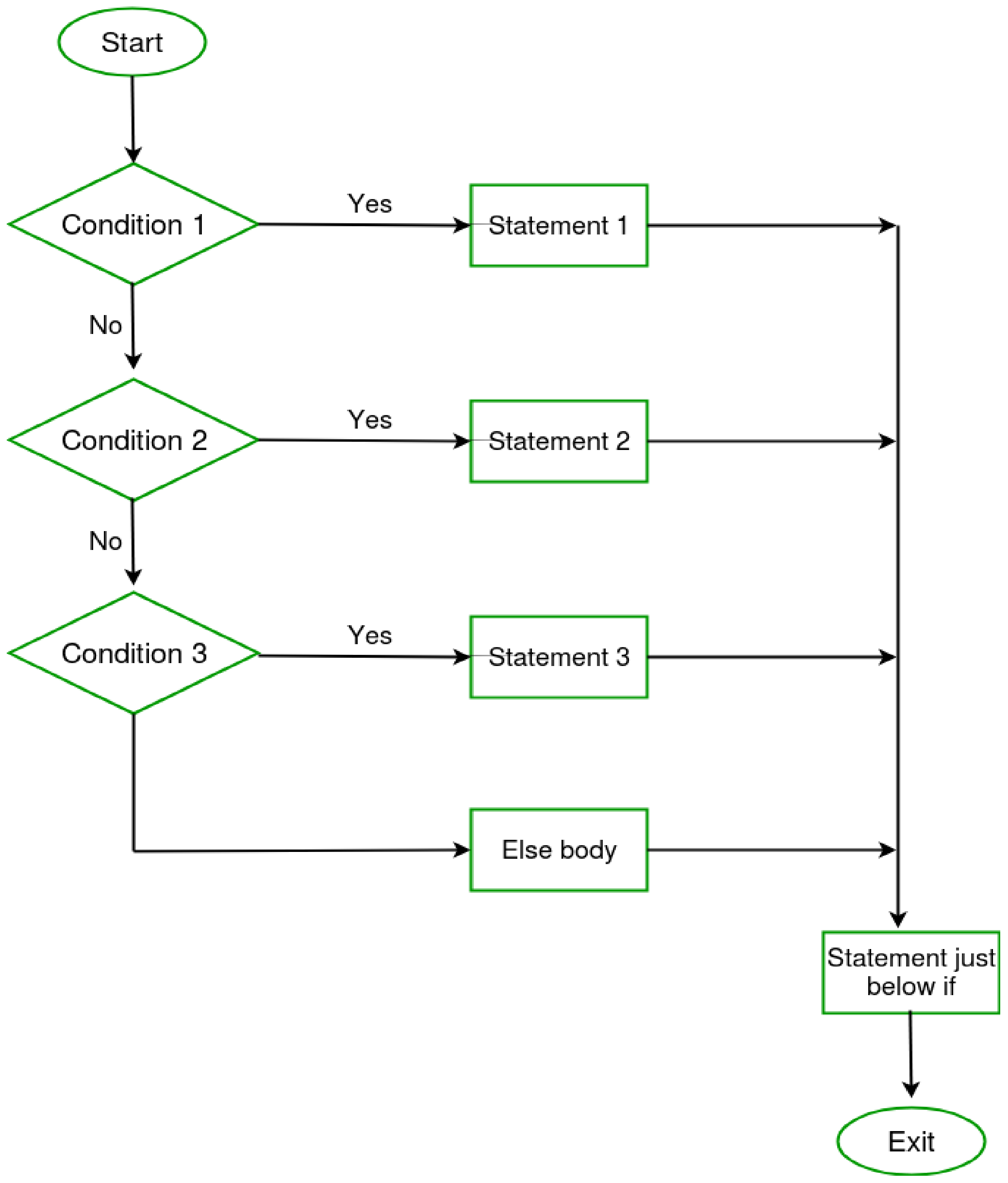
```
#include <stdio.h>
main()
{
    int i = 10;
    if (i == 10)
    {
        if (i < 15)
                printf("i is smaller than 15\n");
        if (i < 12)
                printf("i is smaller than 12 too\n");
        else
        printf("i is greater than 15");
    }
}
```

if-else-if ladder Statements

- ✓ Here, a user can decide among multiple options.
- ✓ The C if statements are executed from the top down.
- ✓ As soon as one of the conditions controlling the if is true, the statement associated with that if is executed, and the rest of the C else-if ladder is bypassed.
- ✓ If none of the conditions are true, then the final else statement will be executed.

```
if (condition)
    statement;
else if (condition)
    statement;
.
.
else
    statement;
```

if-else-if ladder Statements



```
#include <stdio.h>
int main()
{
int i = 20;
if (i == 10)
        printf("i is 10");
else if (i == 15)
        printf("i is 15");
else if (i == 20)
        printf("i is 20");
else
        printf("i is not present");
}
```

Exercise Problems

VIGNAN'S
Foundation for Science, Technology & Research
(Deemed to be UNIVERSITY)
-Estd. u/s 3 of UGC Act 1956

Finding the greatest among the three given numbers.

```
#include <stdio.h>
main()
{
int a, b,c;
printf("Enter Three different values\n");
scanf("%d %d %d", &a, &b, &c);
if(a > b)
{
        if(a>c)
                printf("%d is Largest\n", a);
        else
                printf("%d is Largest\n", c);
}
else
{
        if(b>c)
                printf("%d is Largest\n", b);
        else
                printf("%d is Largest\n", c);
}
}
```

Exercise Problems

In Nested if – else statement, ambiguity is referred as **Dangling else ambiguity.**

```
#include<stdio.h>
main()
{
    int a=10,b=20;
    if(a==100)
    if(b==20)
    printf("Match - I");
    else
    printf("Match - II");
}
```

Exercise Problems

VIGNAN'S
Foundation for Science, Technology & Research
(Deemed to be UNIVERSITY)
-Estd. u/s 3 of UGC Act 1956

In Nested if – else statement, ambiguity is referred as **Dangling else ambiguity.**

```
#include<stdio.h>
main()
{
    int a=10,b=20;
    if(a==100)
    {
        if(b==20)
                printf("Match - I");
    }
    else
        printf("Match - II");
}
```

Exercise Problems

VIGNAN'S Foundation for Science, Technology & Research (Deemed to be UNIVERSITY) -Estd. u/s 3 of UGC Act 1956

```
#include<stdio.h>
main()
{
        int a=10,b=20;
        if(a==100)
                if(b==20)
                        printf("Match - I");
                else
                        printf("Match - II");
        else
                printf("Match - III");
}
O/P: Match - III
```

Exercise Problems

Program to calculate the grade of the student according to the marks.

```
#include <stdio.h>
int main()
{
        int marks;
        printf("Enter your marks?");
        scanf("%d",&marks);
        if(marks > 85 && marks <= 100)
                printf("Congrats ! you scored grade A ...");
        else if (marks > 60 && marks <= 85)
                printf("You scored grade B + ...");
        else if (marks > 40 && marks <= 60)
                printf("You scored grade B ...");
        else if (marks > 30 && marks <= 40)
                printf("You scored grade C ...");
        else
                printf("Sorry you are fail ...");
}
```

Exercise Problems

In C Is it possible to write else clause without any body?

Exercise Problems

VIGNAN'S
Foundation for Science, Technology & Research
(Deemed to be UNIVERSITY)
-Estd. u/s 3 of UGC Act 1956

What will be output when you will execute following c code?

```
#include<stdio.h>
void main()
{
    int a=5,b=10;
    if(++a||++b)
        printf("%d  %d",a,b);
    else
        printf("Doraemon");

}
```

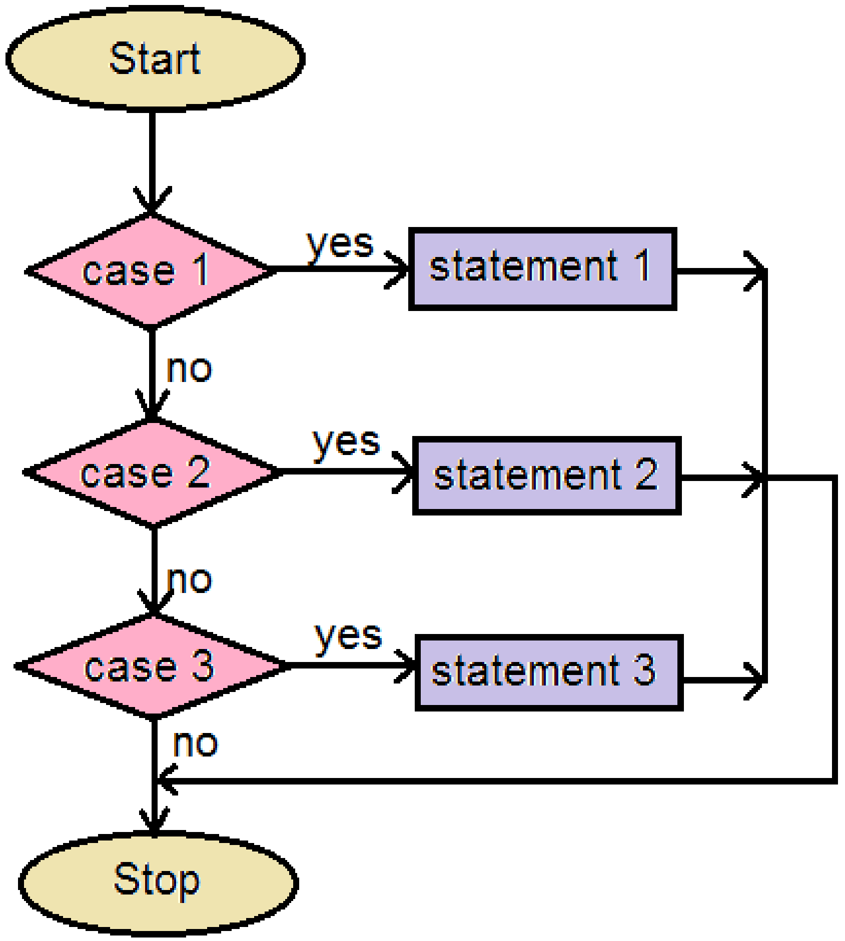
What will be output when you will execute following c code?

```
#include<stdio.h>
void main()
{
    int a=5,b=10;
    if(++a && ++b)
        printf("%d  %d",a,b);
    else
        printf("Doraemon");

}
```

Switch Statements

VIGNAN'S Foundation for Science, Technology & Research (Deemed to be UNIVERSITY) -Estd. u/s 3 of UGC Act 1956

- ✓***Switch statements*** are also decision-making statements which choose to execute a particular block of statements from many cases that has been provided in the code by the programmer.
- ✓It is also known as **switch-case-default**.
- ✓These programs are **menu driven**, i.e., it compares the value to the case values and executes the statements of the case that matches.
- ✓The three keywords used are *switch*, *case* and *default* and they are used to make up the whole **switch-case-default structure**.



Switch Statements

```
#include <stdio.h>
int main() {
 int ch, a = 16, b = 5;
 double result = 0.0;
 printf("Enter 1 for addition, 2 for subtraction, 3 for multiplication and 4 for division :\n");
 scanf("%d", &ch);
 switch (ch) {
 case 1:
   result = 16 + 5;
   printf("The sum is: %f \n", result);
   break;
 case 2:
   result = 16 - 5;
   printf("The difference is: %f \n", result);
   break;
```

Switch Statements

```
case 3:
   result = 16 * 5;
   printf("The product is: %f \n", result);
   break;
 case 4:
   result = 16 / 5;
   printf("The quotient is: %f \n", result);
   break;
 default:
   printf("Wrong Choice\n");
 }
 return 0;
}
```

Output:-

```
Enter 1 for addition, 2 for subtraction, 3
for multiplication and 4 for division :
2
The difference is : 11.0000
```

Switch Statements

VIGNAN'S Foundation for Science, Technology & Research (Deemed to be UNIVERSITY) -Estd. u/s 3 of UGC Act 1956

```
#include <stdio.h>
int main() {
int ch;
printf("Enter 1 for square, 2 for triangle, 3
for rectangle and 4 for circle:\n");
scanf("%d", &ch);
switch (ch) {
case 1:
        printf("You chose Square\n");
        break;
case 2:
        printf("You chose Triangle\n");
case 3:
        printf("You chose Rectangle\n");
        break;
case 4:
        printf("You chose Circle\n");
        break;
default:
        printf("Wrong Choice\n");
}
return 0;
}
```

- **What is the output when you enter 1**
- **What is the output when you enter 2**

Iterative Statements

✓**Iteration** is the process where a set of instructions or statements is executed repeatedly for a specified number of time or until a condition is met.

✓These statements also alter the control flow of the program and thus can also be classified as **control statements**.

- **Iteration** statements are most commonly know as ***loops***.

There are three types of looping statements:

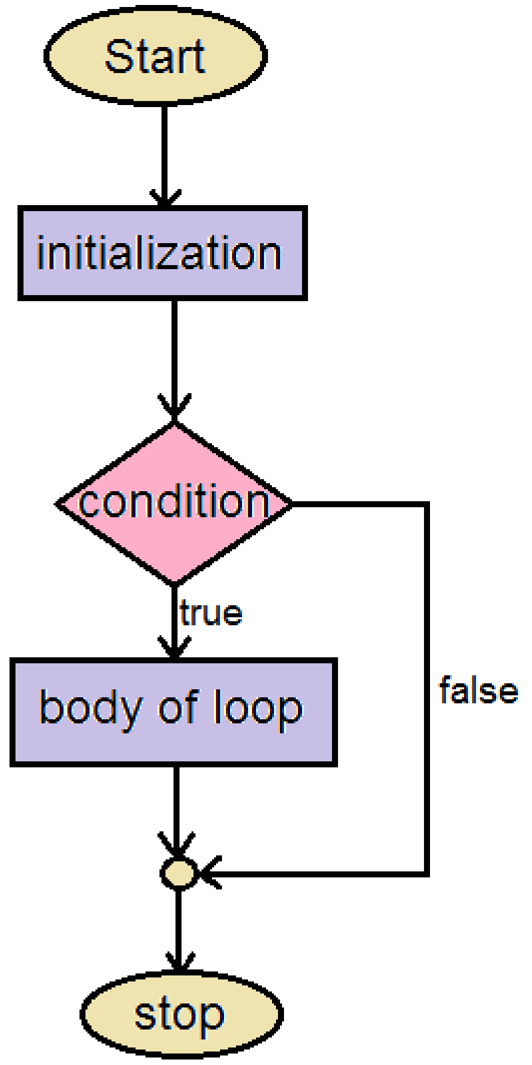
✓For Loop

✓While Loop

✓Do-while loop

Iterative Statements

- A loop basically consists of three parts: **initialization, test expression, increment/decrement or update value**.
- For the different type of loops, these expressions might be present at different stages of the loop.



For Loop

- **For loop** is the most commonly used looping technique.
- It is so popular is because it has all the three parts of the loop: *initialization*, *test expression*, *update expression* in the same line.

```
for(initialization; test expression; update expression)
{
  //body of loop
}
```

```
#include <stdio.h>
int main() {
int i;
for (i = 1; i <= 5; i++)
printf("Hi Vignan\n");
return 0;
}
```

Infinite For Loop

- An infinite for loop is the one which goes on repeatedly for infinite times..

When the loop has no expressions

```
for(;;)
{
  printf(“Hi Vignan");
}
```

When the test condition never becomes false.

```
for(i=1;i<=5;)
{
  printf("CodinGeek");
}
```

Empty For Loop

- An empty for loop is the one which has got no body.
- In simple words, it contains no statements or expressions in its body.
- It can be used for time-consuming purposes.
- Each iteration takes its own time for compilation and execution.
- Usually, it is too small to be of any significant importance.

```
#include <stdio.h>
int main() {
for(i=1;i<=5;i++)
{
}
return 0;
}
```

Nested For Loops

- Nested for loop refers to the process of having one loop inside another loop.
- Multiple loops can be used inside one another.
- The body of one 'for' loop contains the other and so on.

```
for(initialization; test; update)
{
  for(initialization;test;update)//using another variable
  {
    //body of the inner loop
  }
  //body of outer loop(might or might not be present)
}
```

Nested For Loops

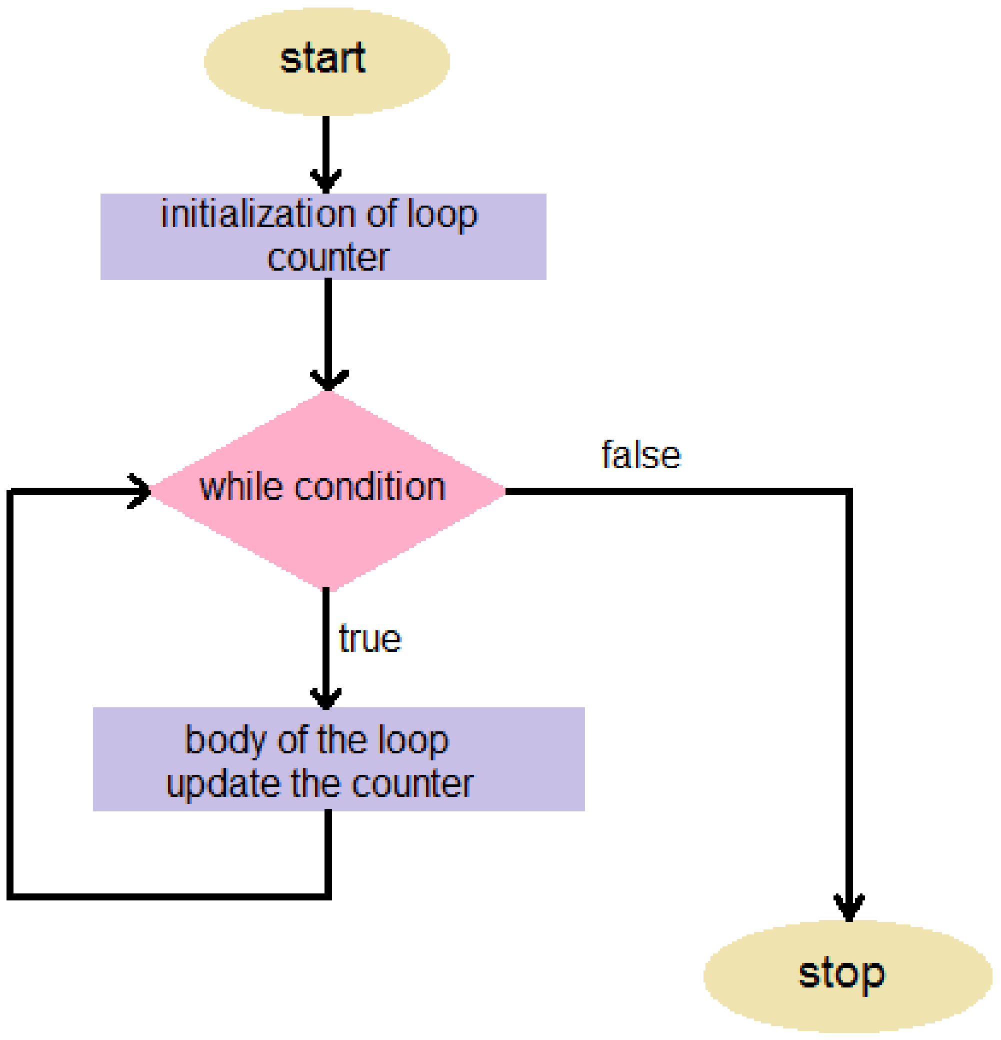
```
#include <stdio.h>
int main()
{
 int i, j;
 for (i = 1; i <= 5; i++)
{
   for (j = 1; j <= i; j++)
     printf("%d", j);
   printf("\n");
 }
 return 0;
}
```

Output:-

```
1
12
123
1234
12345
```

While Loops

- A while loop is very similar to a repeating if statement.
- It executes a certain block of statements based on a certain condition present at the beginning of the loop.
- If the condition returns boolean true the loop block is executed, otherwise not.
- A while loop has its test condition at the beginning of the loop.
- Its initialization is done before the loop and update condition is present in the body of the loop.


start
initialization of loop counter
while condition
false
true
body of the loop update the counter
stop

While Loops

```
initialization;
while(test condition)
{
  //body of the loop
  update expression;
}
```

```
#include <stdio.h>
main()
{
    int i = 1;
    while (i <= 5)
    {
        printf(“Hi Vignan\n");
        i++;
    }
}
```

Infinite While Loops

Infinite while loop: The **while loop** that keeps repeating itself an infinite number of times is known as **infinite while loop**.

When the condition of the **while** loop never turns *false*(no termination of loop), it keeps repeating itself infinite times.

Unlike **for loop**, without *test condition* **while loop**, gives an error.

```
#include <stdio.h>
main()
{
    int i=1;
    while(i<=5)
        printf("Hi Vignan\n");
}
```

Empty While Loops

An empty while loop is the one which does not have any statement or expression in its body(except the update expression).

```
#include <stdio.h>
main()
{
        int i=1;
        while(i<=5)
        {

        }
}
```

Nested While Loops

A nested while loop is one which contains one or more than one loop inside its body.

Note: initialization of the inner loop should also be inside the outer loop.

```
#include <stdio.h>
main()
{
    int i = 1, j;
    while (i <= 3)
    {
        j = 1;
        while (j <= 2)
        {
            printf(“Vignan\n");
            j++;
        }
        printf("-----\n");
        i++;
    }
}
```

Break Jump Statement

- ✓A break statement is used to terminate the execution of the rest of the block where it is present and takes the control out of the block to the next statement.
- ✓It is mostly used in loops and switch-case to bypass the rest of the statement and take the control to the end of the loop.
- ✓Break statement when used in nested loops only terminates the inner loop where it is used and not any of the outer loops.

Do-while Loop Statement

VIGNAN'S
Foundation for Science, Technology & Research
(Deemed to be UNIVERSITY)
-Estd. u/s 3 of UGC Act 1956

- ✓The Do-while loop is an **exit controlled loop**.
- ✓In simple words, the test condition is present after the body of the loop.
- ✓The special feature of a do-while loop is that it executes the body of the loop at least once even if the condition is false.
- ✓A do-while loop is very similar to a while loop except that it is an exit controlled loop.

```
initialization;
do
{
    //body of the loop
    update expression;
}while(test condition);
```

Do-while Loop Statements

```
#include <stdio.h>
main()
{
    int i = 1;
    do
    {
        printf(“Hi Vignan\n");
        i--;
    } while (i == 0);
}
```

```
#include <stdio.h>
main()
{
    int i=1;
    do {
        printf(“Hi Vignan\n");
    }while(i<=5);
}
```

Infinite Do-while Loop

Do-while Loop Statements

Empty Do-while

```
#include <stdio.h>
main()
{
    int i=1;
    do {



    }while(i<=5);
}
```

Jump Statements

✓Jump Statement makes the control jump to another section of the program unconditionally when encountered.

✓It is usually used to terminate the loop or switch-case instantly.

✓It is also used to escape the execution of a section of the program.

- Break
- Continue
- Goto
- Return

Break Jump Statement

```
#include <stdio.h>
main()
{
    int i;
    for (i = 1; i <= 15; i++)
    {
        printf("%d\n", i);
        if (i == 10)
            break;
    }
}
```

Output:-

```
1
2
3
4
5
6
7
8
9
10
```

Continue Jump Statement

VIGNAN'S Foundation for Science, Technology & Research (Deemed to be UNIVERSITY) -Estd. u/s 3 of UGC Act 1956

✓The continue jump statement interrupts or changes the flow of control during the execution of a program.

✓Continue is mostly used in loops.

✓Continue stops the execution of the statements underneath and takes control to the next iteration.

✓Similar to a break statement, in the case of a nested loop, the continue passes the control to the next iteration of the inner loop where it is present and not to any of the outer loops.

Continue Jump Statement

```
#include <stdio.h>
main()
{
    int i, j;
    for (i = 1; i < 3; i++)
    {
        for (j = 1; j < 5; j++)
        {
            if (j == 2)
                continue;
            printf("%d\n", j);
        }
     }
}
```

Output:-

```
1
3
4
1
3
4
```

goto Jump Statement

goto jump statement is used to transfer the flow of control to any part of the program desired.

```
#include <stdio.h>
main()
{
    int i, j;
    for (i = 1; i < 5; i++)
    {
        if (i == 2)
                goto there;
        printf("%d\n", i);
    }
    there:
        printf("Two");
}
```

Output:-

```
1
Two
```

Return Jump Statement

- ✓ Return jump statement is usually used at the end of a function to end or terminate it with or without a value.

- ✓ It takes the control from the calling function back to the main function(main function itself can also have a return).

Return Jump Statement

```
#include <stdio.h>
char func(int ascii)
{
        return ((char) ascii);
}
main()
{
    int ascii;
    char ch;
    printf("Enter any ascii value in decimal: \n");
    scanf("%d", & ascii);
    ch = func(ascii);
    printf("The character is : %c", ch);
}
```

Output:

```
Enter any ascii value in decimal:
110
The character is : n
```

VIGNAN'S
Foundation for Science, Technology & Research
(Deemed to be UNIVERSITY)
-Estd. u/s 3 of UGC Act 1956