“C Programming”

3. Control Statements

Control Statements

- **Definition:** The control-flow of a language specify the order in which computations are performed.
- **Types:**
  - **Decision Making (Branching)**
    - if else statement, if..else if..else Statement (Multi-way decision making)
    - Nested if statement
    - Switch statement
  - **Loops**
    - For Loop
    - While Loop
    - Do while Loop

if control statements

- **Definition:** The if-else statement is used to express decisions.
- The else part is optional.

Syntax:

```
if (expression)
    statement;
```

Example :

```
#define MAX_LINES 100
if(line_num == MAX_LINES)
  line_num=0;
```

Syntax compound statements:

```
if (expression)
{
   statement-1;
   statement-2;
   statement-3;
}
```

Example :

```
#define MAX_LINES 100
if(line_num == MAX_LINES )
{
  line_num=0;
  page_num++;
}
```

if..else

- **Definition:** The if-else statement is used to express decisions when one is true and other is not.

Syntax:

```
if (expression)
    statement;
else
     statement;
```

Example :

```
if(a>b)
   printf( " a is greater\n);
else
   printf(" b is greater\n");
```

Syntax:

```
if (expression){
    statement;
}
else{
     statement;
}
```

Example :

```
if(a>b) {
   printf( " a is greater\n);
   printf(" b is smaller\n");
}
else{
   printf(" b is greater\n");
}
```

if..else

Example : Write a C program to check given number is odd or even?

```
#include <stdio.h>
int main()
{
      int num;
      printf("Enter a number to check.\n");
      scanf("%d",&num);

      if(num%2 == 0)
      {
            printf(“ Number %d is even\n",num);
      }
      else
             printf(“ Number %d is odd\n",num);
    return 0;
}
```

If..else..if..

- Multi Way decision making: If any expression is true, the corresponding statement is executed, and chain terminates
- Last “else” part handles “none of the above”

Syntax:

```
if (expression)
    statement 1;
else if (expression)
     statement 2;
else if  (expression)
     statement 3;
else
      statement 4;
```

Example :

```
if(a>b)
   printf( “ a is greater\n);
else if (b>a)
   printf(“ b is greater\n”);
else
   printf(“ a & b are equal”);
```

Nested if

- Nested if: You can use one **if** or **else if** statement inside another **if** or **else if** statement(s).

Syntax:

```
if (expression)
    statement 1;
    if (expression)
     statement 2;
    else
      statement 3;
```

Example :

```
if( a == 100 )
  {
        if( b == 200 )
        {
                printf("Value of a is 100\n"";
                printf("Value of b is 200\n" );
        }
         else
                printf("Value of a is 100\n");
  }
```

If statement

- Whenever condition is evaluated and Results in Non-Zero Value then Condition is Considered as TRUE
- Condition is treated as FALSE , if Result of Expression is Zero
- Generally 1 indicates TRUE , 0 indicates FALSE value
- Opening and Closing braces are not required if there is only one Statement after if.
- Opening and Closing braces are not required if there is only one Statement after else.
- Block followed after if and before else is called "if block"
- if there is no statement after else then else can be dropped
- More than one Conditions can be Combined inside if
- Nested Control Statements are allowed
- ! Operator Reveres the Value of Expression
- Logical && operator indicates "both Conditions must be TRUE"
- Logical || operator indicated "Either Conditions must be TRUE"

Class Room Work

1. Write a program to find out the smallest and the largest of three integers using
   (a) Only if
   (b) Only if else
   (c) Nested if

Class Room Work

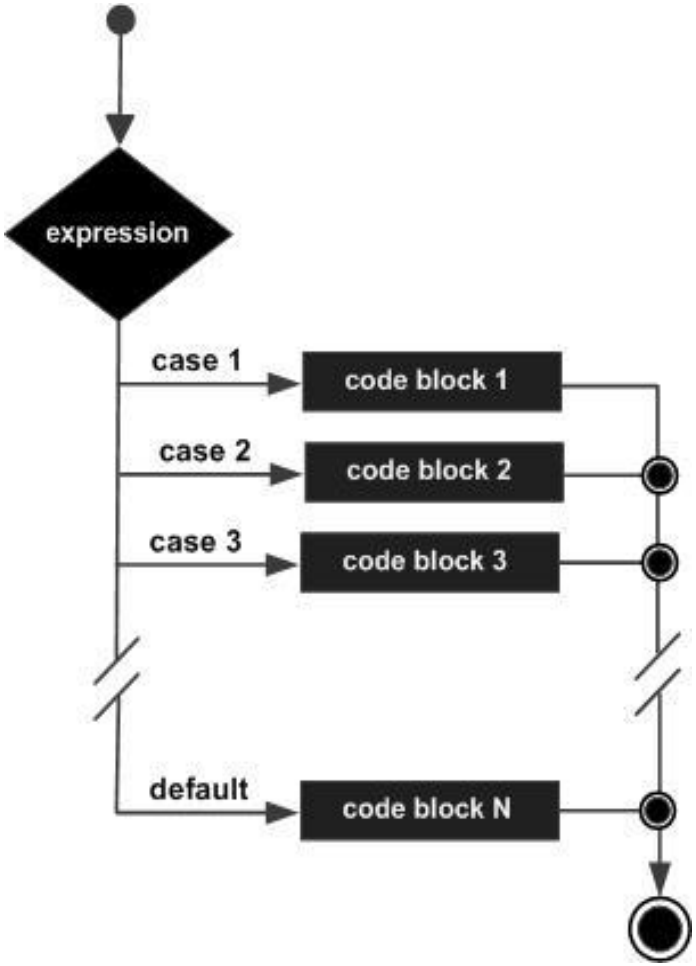
1. Write logical expressions that tests whether a given character is
    - lower case letter
    - upper case letter
    - digit
    - white space (includes space, tab,new line)

Switch

- Definition: The switch statement is a multi-way decision that tests whether an expression matches one of a number of *constant* integer values, and branches accordingly.

Syntax:

```
switch(expression)
{
    case const-expr :
                        statements;
                        break;
    case const-expr :
                        statements;
                        break;
    ……
    ……
    default             :
                        statements;
}
```



Switch - Rules

Rules:

- The **expression used in a switch statement** must have an integral or enumerated type.
- When the variable being switched on is equal to a case, the statements following that case will execute until a **break** statement is reached.
- break statement terminates the switch.
- You can have any number of case statements within a switch.
- A switch statement can have an optional default case. We should have only one default case.
- Default case can be kept any where in the switch.
- Not every case needs to contain a break.

Switch – rules..

- Case label must be unique.
- Empty Switch case is allowed.
- Case label must be a constant or constant expression

```
case 1+1:
case 'A' :
case 67:
```

- Case label must be an integer or character, cannot be float

```
case 10.12:
case 7.5:
```

- Two or more cases can share a single break.
- Nesting of switch statements is allowed.
- Macro Identifier (Symbolic constant) is allowed as Switch Case Label.

Switch – rules..

- Case labels must be of integral type (integer,character)

```
case 10:
case 20+20:
case 'A':
case 'a':
```

- Variable not allowed in switch case labels

```
case var :
case num1 :
case n1+n2 :
```

Switch – Example1

Example1: */* Simple Calculator – Add, Sub, Mul, Div */*

```
#include <stdio.h>
int main()
{
    char  op;
    float num1, num2;

   printf("Select an operator either + or - or * or / \n");
    scanf("%c",&op);

    printf("Enter two operands: ");
    scanf("%f%f",&num1,&num2);

   switch(op)
   {
       // case constant-expression
       case '+':
           printf("%f + %f = %f",num1, num2, num1+num2);
           break;
```

Switch – Example1…

Example1: */* Simple Calculator – Add, Sub, Mul, Div */*

```
        case '-':
            printf("%f - %f = %f",num1, num2, num1-num2);
            break;
        case '*':
            printf("%f * %f = %f",num1, num2, num1*num2);
            break;
        case '/':
            printf("%f /  %f = %f",num1, num2, num1/num2);
            break;
        default:
            /* If operator is other than +, -, * or /, error message */
            printf("Error! operator is not correct");
            break;
    }
    return 0;
}
```

Switch…QUIZ

```
#include <stdio.h>
int main ()
{
        int value = 3;
        switch(value)
        {
        case 1:
            printf(“Value is 1 \n” );
            break;
        case 2:
            printf(“Value is 2 \n” );
            break;
        case 3:
           printf(“Value is 3 \n” );
            break;
        case 4:
           printf(“Value is 4 \n” );
           break;
```

```
  default :
   printf(“Value is other than 1,2,3,4 \n” );

  }//end of switch
return 0;
}//end of main
```

Output:
Value is 3

More on switch

```
# include <stdio.h>
int main() {
        int id=3;
        switch(id)
       {
       case 1:
              printf("C Programming Language");
              break;
       case 2:
              printf("C++ Programming Language");
              break;
       case 2:
              printf("Web Technology");
              break;
       default :
              printf("No student found");
              break;
        }
        return 0;
}
```

Case Label must be unique

Switch case with 2 defaults?

```
# include <stdio.h>
int main() {
int id=3;
        switch(id)
       {
       case 1:
              printf("C Programming Language");
              break;
       case 2:
              printf("C++ Programming Language");
              break;
       case 3:
              printf("Web Technology");
              break;
       default :
              printf("No student found");
              break;
       default :
              printf("No student found");
              break;
        }
    return 0;
}
~
```

Switch case should have atmost one default label

Default can be anywhere in the switch ?

```
# include <stdio.h>
int main() {
int id=3;
        switch(id)
       {
       case 1:
              printf("C Programming Language");
              break;
       default :
              printf("No student found");
              break;
       case 2:
              printf("C++ Programming Language");
              break;
       case 3:
              printf("Web Technology");
              break;
        }
    return 0;
}
```

Yes, Default can be anywhere in the switch

More on switch

```
#include <stdio.h>
int main() {
        int id=1;
        switch(id)
       {
       case 1:
              printf("C Programming Language");
       case 2:
              printf("C++ Programming Language");
       case 3:
              printf("Web Technology");
              break;
       default :
              printf("No student found");
              break;
        }
        return 0;
}
~
```

Break statement takes control out of switch

2 or more case may share one break statement

More on switch

```
# include <stdio.h>
#define NUM 1
int main() {
        int id=1;
        switch(id)
       {
       case NUM:
              printf("C Programming Language");
              break;
       case 2:
              printf("C++ Programming Language");
              break;
       case 3:
              printf("Web Technology");
              break;
       default :
              printf("No student found");
              break;
        }
    return 0;
}
```

Macro identifiers are allowed as switch case labels

if else if else Vs switch

if else if

```
if (grade == 4)
        printf(“excellent\n”);
else if (grade == 3)
        printf(“good\n”);
else if (grade == 2)
        printf(“average\n”);
else if (grade == 1)
        printf(“poor\n”);
else if (grade == 0)
        printf(“failed\n”);
else
         printf(“Invalid grade\n”);
```

switch

```
switch(grade) {
        case 4: printf(“excellent\n”);
                break;
        case 3: printf(“good \n”);
                break;
        case 2: printf(“average \n”);
                break;
        case 1: printf(“poor\n”);
                break;
        case 0: printf(“failed \n”);
                break;
        default:
                printf(“Invalid grade\n”);
                 break;
}
```

dangling else

```
if (y != 0)
    if(x != 0 )
        result = x/y;
else
    printf(“error y equal to zero;
```

```
if (y != 0)
    if(x != 0 )
        result = x/y;
    else
        printf(“error y equal to zero;
```

If else .vs. Switch

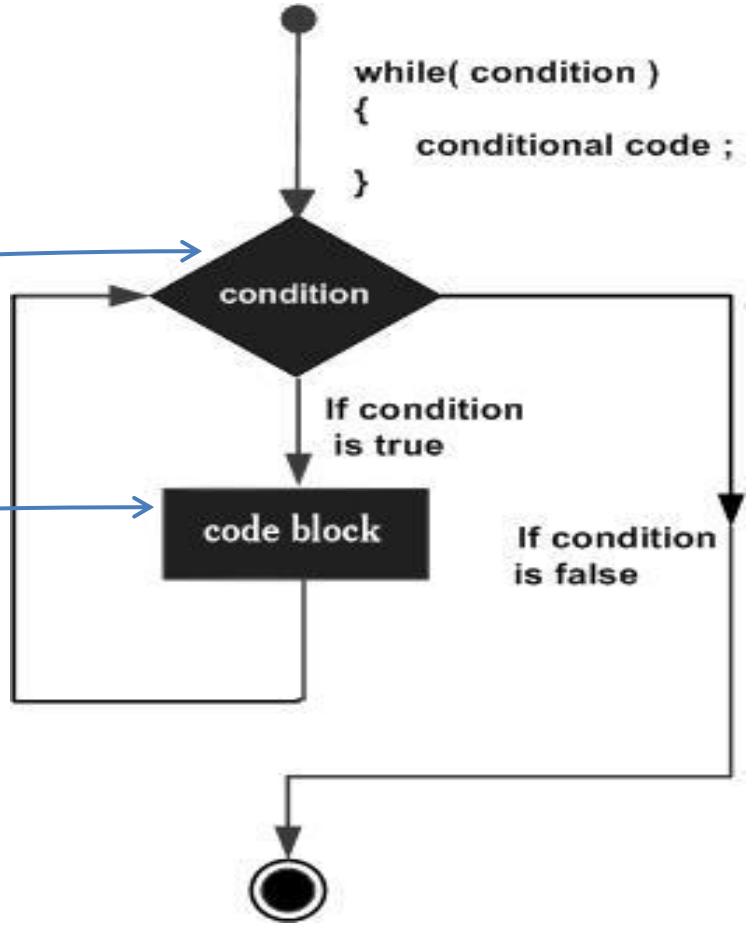
else if Ladder	Switch
In else if ladder, the control goes through the every else if statement until it finds true value of the statement or it comes to the end of the else if ladder.	In case of switch case, as per the value of the switch, the control jumps to the corresponding case.
Less compact, Not readable	More compact than lot of nested else if. More readable.
Less efficient (slow), if the size of the ladder is big	More efficient (Fast) – Jump Table or Branch Table *(Because compiler generates a jump table for a switch during compilation. Consequently, during execution, instead of checking which case is satisfied, it only* decides *which case has to be executed.)*
Can use float expressions	Cannot use float expressions

While loop

- A **while loop** is a repetition control structure that allows you to execute a block of statements as long as a given condition is true.
- When we do not know the number of iterations in advance.

Syntax:

```
while (expression)
{
   statement(s);
}
```




```
while ( i < n)
    i = i * 2;
```

While loop

- A **while loop** is a repetition control structure that allows you to execute a block of statements as long as a given condition is true.
- When we do not know the number of iterations in advance.

```
#include <stdio.h>
int main()
{
        int i=3;
         while(i<10)
        {
                printf("%d\n",i);
                i++;
         }
}
```

Output:

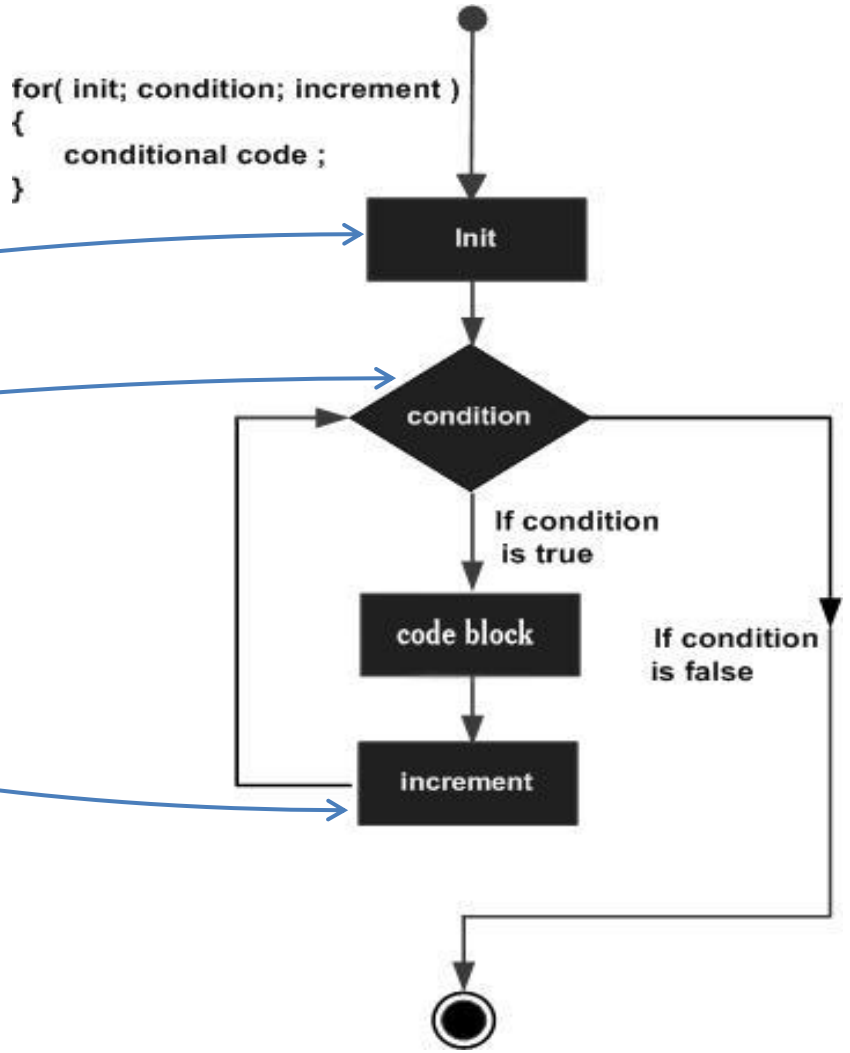
3 4 5 6 7 8 9

Loops – for loop

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

Syntax:

```
for (expr1; expr2 ; expr 3)
{
    statement(s);
}
```

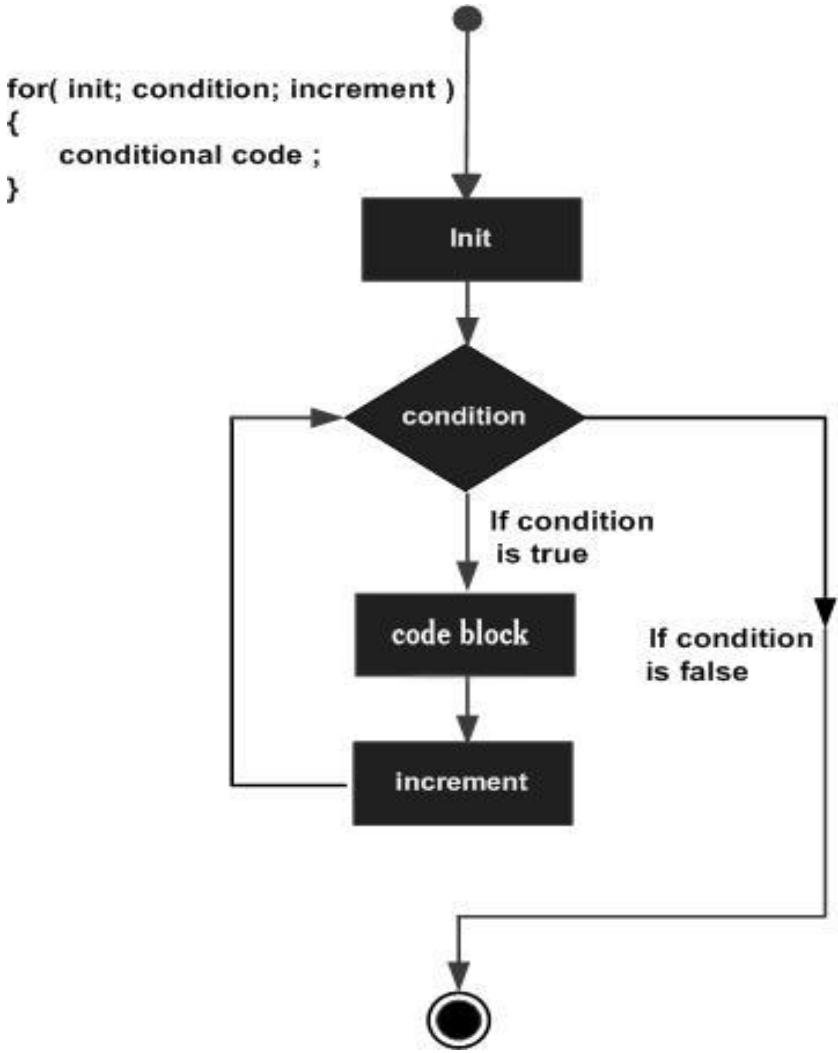


Loops – for loop

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

```
#include <stdio.h>
int main()
{
        int i;
         for(i=0;i<10;i++)
        {
                printf("%d ",i);
         }
}
```

Output:
0 1 2 3 4 5 6 7 8 9



Nested For loop - Example

Example: /* Print Floid Triangle */

```
#include <stdio.h>
int main()
{
  int n, i,  c, a = 1;
  printf("Enter the number of rows of Floyd's triangle to print\n");
  scanf("%d", &n);

 for (i = 1; i <= n; i++)
  {
    for (c = 1; c <= i; c++)
    {
      printf("%d ",a);
      a++;
    }
    printf("\n");
  }
   return 0;
}
```

Floyd's Triangle

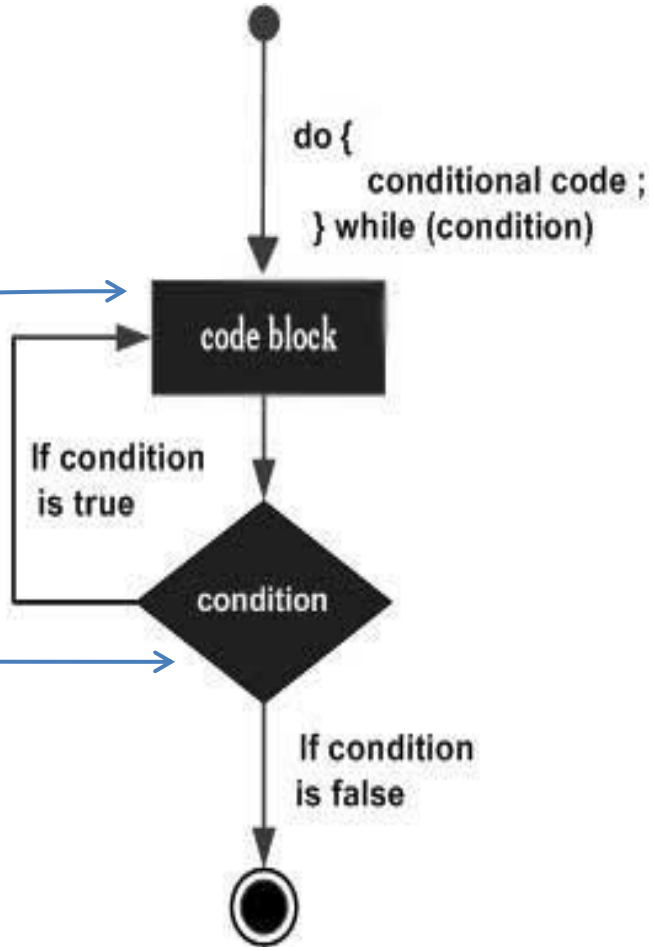
```
1
2  3
4  5  6
7  8  9  10
11 12 13 14 15
```

do..While loop

- A **do-while loop** executes a block of statements at least once.
- The remaining behavior is same as while loop.

Syntax:

```
do
{
   statement(s);
} while(expression);
```



do..While loop

- A **do-while loop** executes a block of statements at least once.
- The remaining behavior is same as while loop.

```
#include <stdio.h>
int main()
{
        int i=1;
        do
        {
                printf("Value of i is %d\n",i);
                i++;
        }
        while(i<=4 && i>=2);
}
```

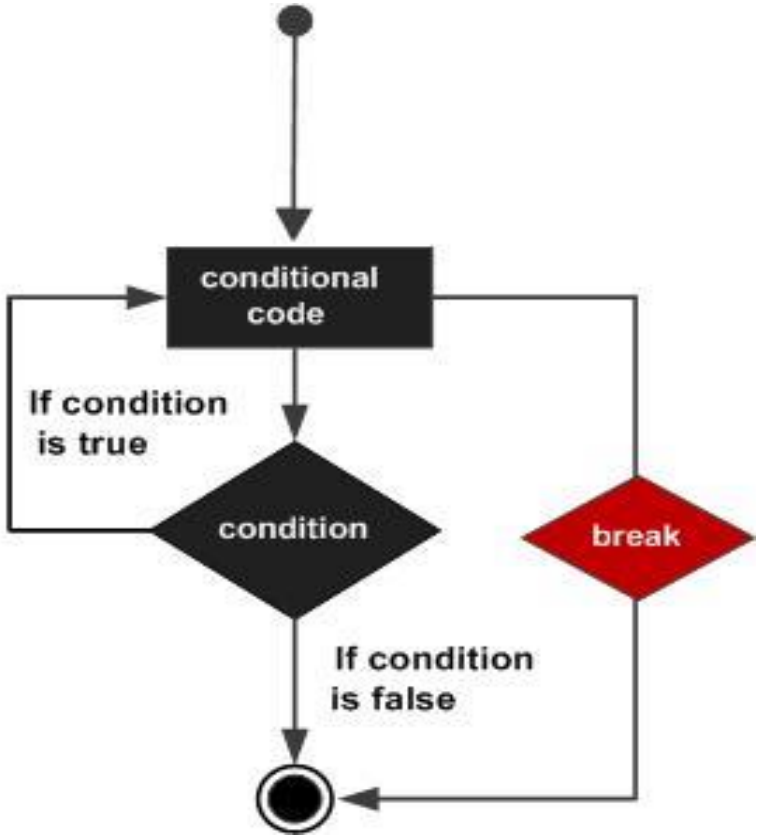
```
Output:
Value of i is 1
Value of i is 2
Value of i is 3
Value of i is 4
```

break

- A **break statement** is used for early exit from a loop.
- It can be used to terminate a case in the **switch** statement
- If you are using nested loops then the break statement will stop the execution of the innermost loop and start executing the next line of code after the block.

Syntax: **break;**

```
while( a < 20 )
   {
      printf("value of a: %d\n", a);
      a++;
      if( a > 15)
      {
          break;
      }
   }
```



break

```
#include <stdio.h>
 int main()
{
        int i;
        for(i=0;i<10;i++)
        {
                if(i==5)
                {
                        printf("\nComing out of for loop when i = 5″);
                        break;
                }
                printf("%d ",i);
        }
        return 0;
}
```

Output:

0 1 2 3 4

Coming out of for loop when i = 5

THANK YOU