

Compiled by ab

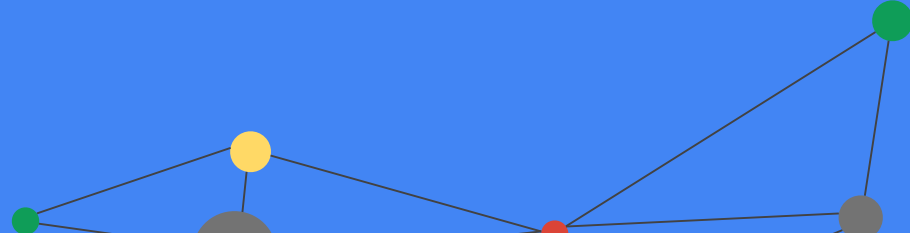


# Computer Concept and Programming

I semester, B.Sc.CSIT

Ambition Guru College

**Compiled by :**  
**Ankit Bhattarai**



# Unit III

(6 hrs.)

## Control Structure

- Introduction,
- Decision Making Statements,
- Looping Statements,
- Branching Statements,
- Common Pitfalls: Infinite loops, Misplacement of Conditionals.

- Control structure enables, us to specify the order in which the various instructions in a program are to be executed by the computer.
- In other words control instructions determine the “flow of control” in a program.
- There are 3 types of control instructions in C. They are:
  1. Sequential Control instruction
  2. Selection or decision control instruction
  3. Repetition or looping control instruction

# Control Structure : Sequential Control Structure

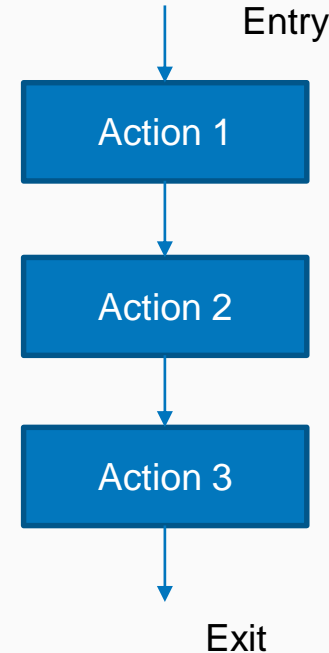
Compiled by ab

- Instructions are executed in the same order in which they appear
- Each instruction is executed exactly once.
- No any condition is evaluated.

## Example:

**Program to find the sum of two numbers.**

```
#include<stdio.h>
#include<conio.h>
int main()
{
    int num1,num2,sum;
    printf("Enter the two numbers");
    scanf("%d%d",&num1,&num2);
    sum=num1+num2;
    printf("Sum of two numbers is %d",sum);
    getch();
    return 0
}
```



## **Selection/Branching/Decision Control instructions**

# Control Structure : Selection/Branching/Decision Control instructions

Compiled by ab

- They are used when we have a number of situations where we may need to change the order of execution of statements based on certain conditions.
  - I. simple if statement
  - II. if else statement
  - III. nested if else statement
  - IV. else if ladder

Selection/Branching/Decision Control instructions : **if statement**

# Selection/Branching/Decision Control instructions : if statement

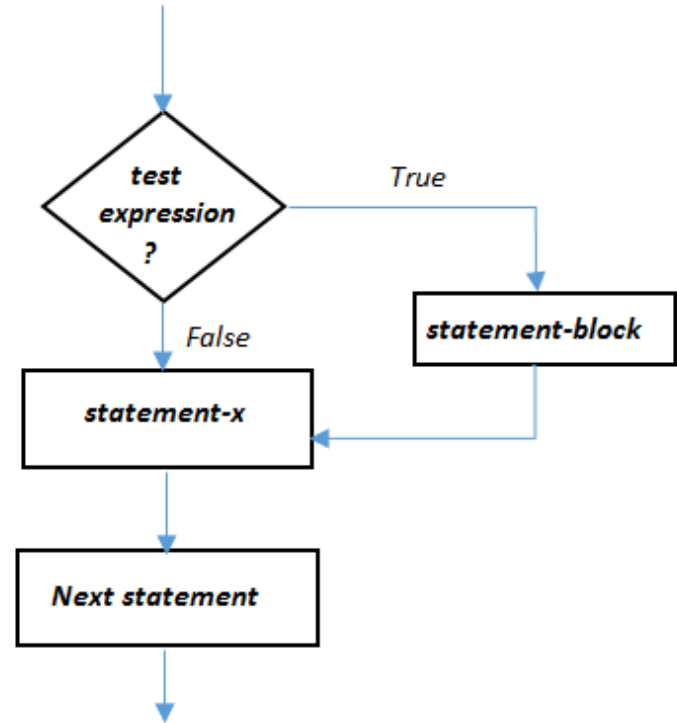
Compiled by ab

Evaluates the expression first then do the following:

- If the value of expression is true , it executes the statement within the block
- Otherwise it skips the statements within its block and continues from the first statement outside the block

## Syntax:

```
if(test expression)
{
    statement-block;
}
statement-x;
```





## if statement example

WAP that reads an integer from the user and checks if the number is positive.

```
#include <stdio.h>
```

```
int main() {
```

```
    int number;
```

```
    printf("Enter an integer: ");
```

```
    scanf("%d", &number);
```

```
    if (number > 0) {
```

```
        printf("The number is positive.\n");
```

```
    }
```

```
    return 0;
```

```
}
```

## if statement example

WAP to input the average marks of a student and add 10% bonus marks if his/her average marks is greater than or equal to 65.

```
#include<stdio.h>
#include<conio.h>
int main()
{
    float marks;
    printf("Enter the marks\n") ;
    scanf("%f",&marks);
    if(marks>=65)
    {
        marks=marks+marks*0.1;
    }
    printf("Final marks=%f",marks);
    getch();
    return 0;
}
```

Selection/Branching/Decision Control instructions : **if-else statement**

## Extension of simple if statement.

- If test expression is true,
  - then true block statement(s) (*immediately following the if statements are executed*)
- otherwise false block statements are executed

## Syntax:

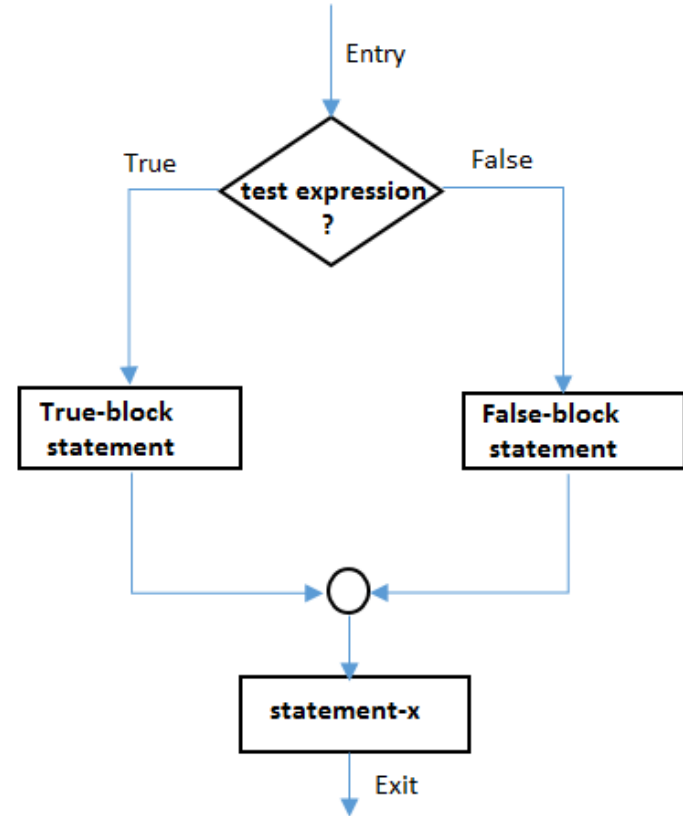
```
if(test_expression)
{
    true-block statements
}
else
{
    false-block statements
}
statement-x;
```

# Selection/Branching/Decision Control instructions : **if-else statement**

Compiled by ab

## Syntax:

```
if(test_expression)
{
    true-block statements
}
else
{
    false-block statements
}
statement-x;
```



```
#include <stdio.h>
```

```
int main() {
```

```
    int age;
```

```
    printf("Enter your age: ");
```

```
    scanf("%d", &age);
```

```
    if (age >= 18) {
```

```
        printf("You are eligible to vote.\n");
```

```
    } else {
```

```
        printf("You are not eligible to vote.\n");
```

```
    }
```

```
    return 0;
```

```
}
```

## if-else example

- Write a C program that reads the age of a person and checks if they are eligible to vote. A person is eligible to vote if they are 18 years or older. If they are eligible, display a message indicating that they can vote; otherwise, display a message indicating that they are not eligible.

## if-else example

- WAP to find the maximum number between two numbers

```
#include<stdio.h>
#include<conio.h>
int main()
{
    int num1,num2;
    printf("Enter the first number\n");
    scanf("%d",&num1);
    printf("Enter the second number\n");
    scanf("%d",&num2);
    if(num1>num2)
    {
        printf("Maximum number=%d",num1);
    }
    else
    {
        printf("Maximum number=%d",num2);
    }
    getch();
    return 0;
}
```

Selection/Branching/Decision Control instructions : **Nested if-else statement**



# Selection/Branching/Decision Control instructions : Nested if-else statement

When a series of decisions are involved, we may have to use more than one if. . . else statement in nested form as shown below.

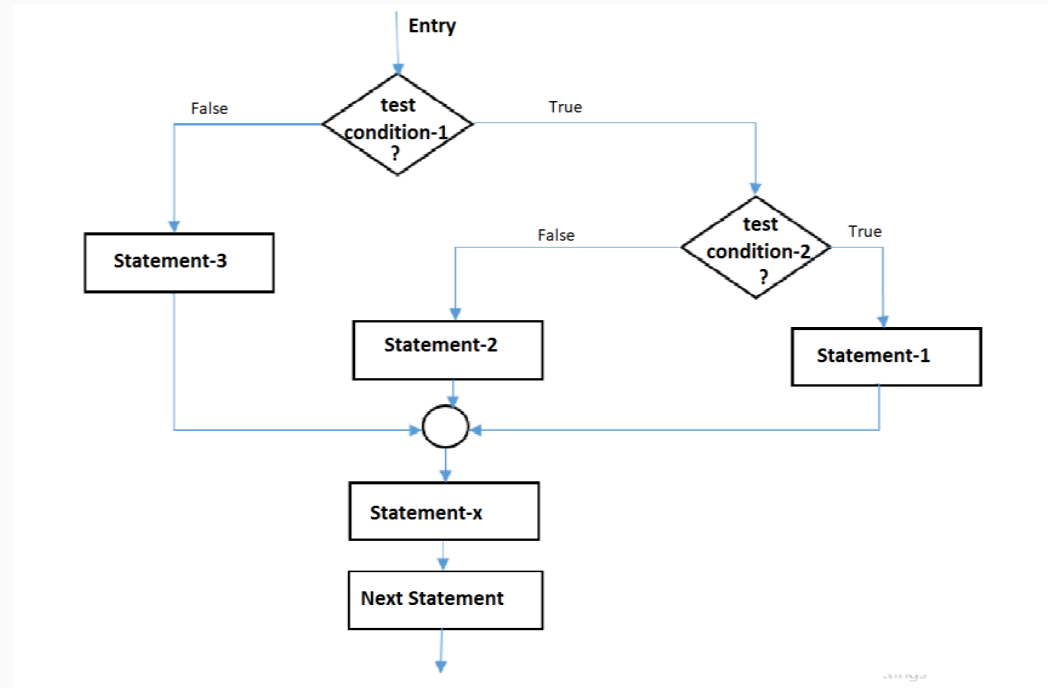
**Syntax:**

```
if(test condition-1)
{
    if(test condition-2)
    {
        Statement -1;
    }
    else
    {
        Statement-2;
    }
}
else
{
    Statement-3;
}
Statement x;
```

# Selection/Branching/Decision Control instructions : Nested if-else statement

## Syntax:

```
if(test condition-1)
{
    if(test condition-2)
    {
        Statement -1;
    }
    else
    {
        Statement-2;
    }
}
else
{
    Statement-3;
}
Statement x;
```



## Nested if-else example

- WAP to find the largest number among three numbers

```
#include<stdio.h>
#include<conio.h>
int main()
{
    int a,b,c;
    printf("Enter three number\n");
    scanf("%d%d%d",&a,&b,&c);
    if(a>b)
    {
        if(a>c)
        {
            printf("The largest number is %d",a);
        }
        else
        {
            printf("The largest number is %d",c);
        }
    }
}
```

## Nested if-else example

- WAP to find the largest number among three numbers

```
else
{
    if(b>c)
    {
        printf("Largest number is %d",b);
    }
    else
    {
        printf("largest number is %d",c);
    }
}
```

## Nested if-else example

- WAP to find the largest number among three numbers

```
else
{
    if(b>c)
    {
        printf("Largest number is %d",b);
    }
    else
    {
        printf("largest number is %d",c);
    }
}
```

## Nested if-else example

- Write a C program that reads an integer from the user and checks if it is positive, negative, or zero using nested if-else statements.

```
#include <stdio.h>

int main() {
    int number;

    printf("Enter an integer: ");
    scanf("%d", &number);

    if (number > 0) {
        printf("The number is positive.\n");
    } else {
        if (number < 0) {
            printf("The number is negative.\n");
        } else {
            printf("The number is zero.\n");
        }
    }

    return 0;
}
```

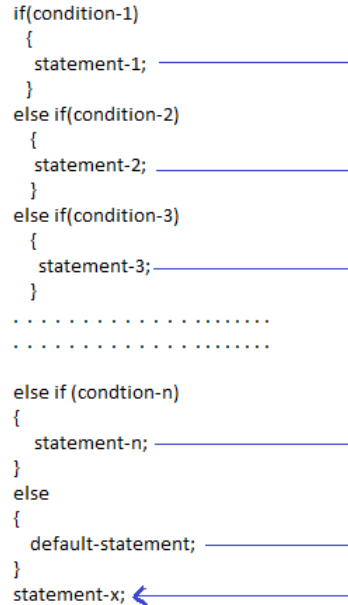
Selection/Branching/Decision Control instructions : **else-if ladder statement**

## else-if ladder statement

- It is used when multiple decisions are involved. Here, the condition expression is evaluated in order.
- ✓ If any of these expression is true, the statement associated with it is executed and this terminates the whole chain.
- ✓ If none of the expression is true then statement associated with final else is executed.

### Syntax:

```
if(condition-1)
{
    statement-1;
}
else if(condition-2)
{
    statement-2;
}
else if(condition-3)
{
    statement-3;
}
.....
else if (condition-n)
{
    statement-n;
}
else
{
    default-statement;
}
statement-x;
```

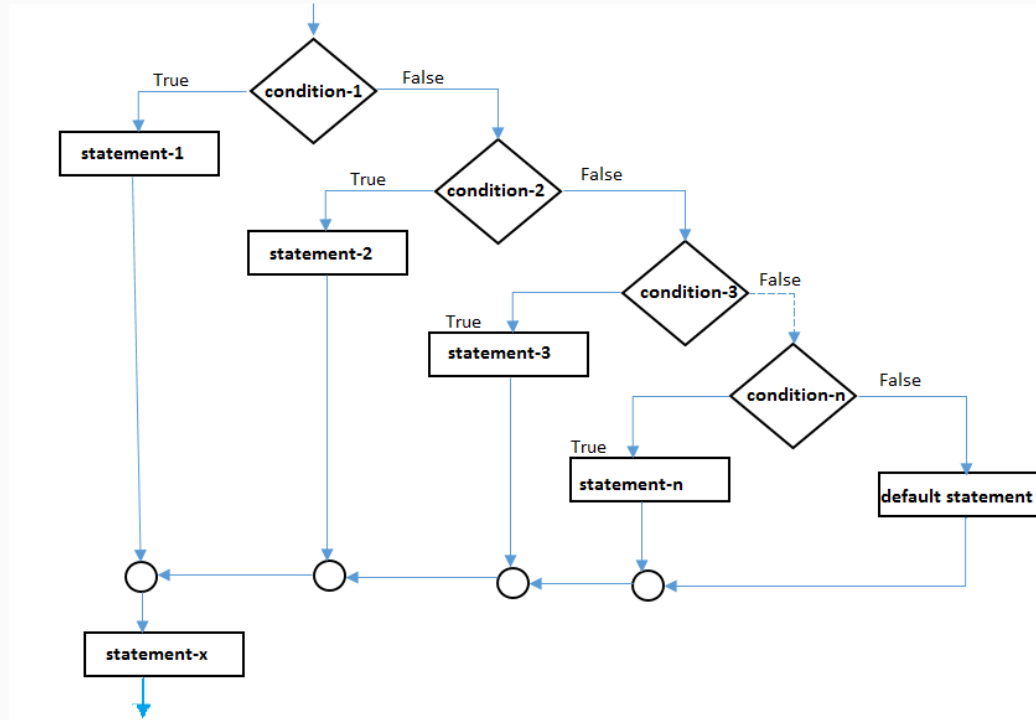




## else-if ladder statement

- It is used when multiple decisions are involved. Here, the condition expression is evaluated in order.
- ✓ If any of these expression is true, the statement associated with it is executed and this terminates the whole chain.
- ✓ If none of the expression is true then statement associated with final else is executed.

### Flowchart:



## else-if ladder statement

- Write a C program to read a person's marks and determine their grade using an **else-if ladder**. The grading criteria are as follows:

Marks  $\geq 90$ : Grade A

Marks  $\geq 80$ : Grade B

Marks  $\geq 70$ : Grade C

Marks  $\geq 60$ : Grade D

Marks  $< 60$ : Fail

```
#include <stdio.h>

int main() {
    int marks;

    printf("Enter your marks: ");
    scanf("%d", &marks);

    if (marks >= 90) {
        printf("Grade: A\n");
    } else if (marks >= 80) {
        printf("Grade: B\n");
    } else if (marks >= 70) {
        printf("Grade: C\n");
    } else if (marks >= 60) {
        printf("Grade: D\n");
    } else {
        printf("Grade: Fail\n");
    }

    return 0;
}
```

## else-if ladder statement

An electricity board charges according to the following rates:

For the first 100 units ..... Rs 40 Per Unit

For the next 200 units.....Rs. 50 Per Unit

For the beyond 300 Units.....Rs.60 Per unit  
All users are also charge meter charge. Which is equal to Rs.50. WAP to read number of units consumed and print out total charges.

```
#include<stdio.h>
#include<conio.h>
#define METER_CHARGE 50
int main()
{
    int units,charge,totalcharge;
    printf("Enter the number of units\n");
    scanf("%d",&units);
    if(units<=100)
    {
        charge=units*40;
    }
    else if(units<=300)
    {
        charge=100*40+(units-100)*50;
    }
    else
    {
        charge=100*40+200*50+(units-300)*60;
    }
```

## else-if ladder statement

An electricity board charges according to the following rates:

For the first 100 units ..... Rs 40 Per Unit

For the next 200 units.....Rs. 50 Per Unit

For the beyond 300 Units.....Rs.60 Per unit  
All users are also charge meter charge. Which is equal to Rs.50. WAP to read number of units consumed and print out total charges.

```
totalcharge=charge+METER_CHARGE;  
printf("Total charge=%d",totalcharge);  
getch();  
return 0;  
}
```

## **Repetition/Iteration/Loop Control Instructions**

Loop control instructions causes a program to execute the certain block of code until some conditions for termination of loop are satisfied.

Basically there are three types of loop control instructions.

1. while Loop (*Entry Controlled loop*)
2. do while loop (*Exit Controlled loop*)
3. for loop

## Repetition/Iteration/Loop Control Instructions : `while` loop

# Repetition/Iteration/Loop Control Instructions : while loop

Compiled by ab

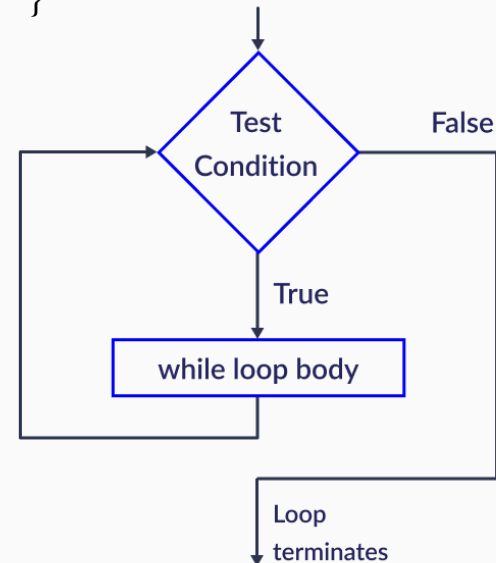
## ✓ Entry Controlled loop

Test condition is evaluated first

- If the condition is true the body of the loop is executed.
- After the execution of the body, test condition is again evaluated and if it is true body is executed once again. This process goes on until test condition is false.
- If test condition is false, the body of the loop will not be executed and control is transferred out of the loop.

## Syntax:

```
while (test condition)
{
    body of the loop
}
```





## while loop

compiled by ab

**WAP to print 10 natural numbers using while loop.**

```
#include<stdio.h>

int main()
{
    int i;
    i=1;
    while(i<=10)
    {
        printf("%d\n",i);
        i++;
    }

    return 0;
}
```

## while loop

compiled by ab

**WAP to print the sum of numbers from 1 to 10 using a while loop.**

```
#include <stdio.h>

int main() {
    int i = 1, sum = 0;

    while (i <= 10) {
        sum += i;
        i++;
    }

    printf("The sum of numbers from 1 to 10 is: %d\n", sum);

    return 0;
}
```

## Repetition/Iteration/Loop Control Instructions : `do...while` loop

# Repetition/Iteration/Loop Control Instructions : do...while loop

Compiled by ab

## ✓ Exit Controlled loop

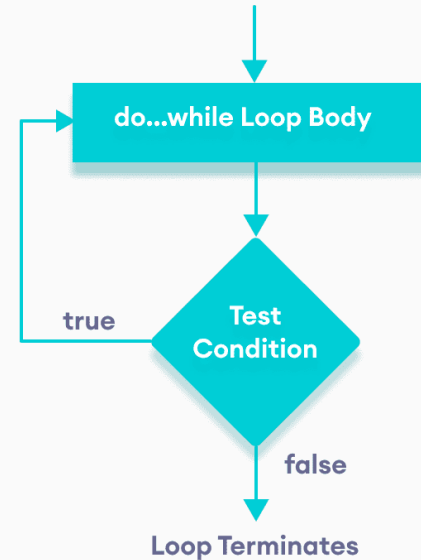
The test is performed at end of the body of the loop and therefore the body of the loop is executed unconditionally for the first time.

Then, test condition is evaluated.

- If test condition is true, the body of the loop is executed again. This process goes on until the test condition is false.
- When test condition is false, loop is terminated and control goes to the statement that appears immediately after the while statement.

## Syntax

```
do  
{  
  Body of the loop  
} while(test condition);
```



## do...while loop

**WAP to print 10 natural  
numbers using while  
do...while loop.**

```
#include<stdio.h>
#include<conio.h>
int main()
{
    int i;
    i=1;
    do
        {
            printf("%d\n",i);
            i++;
        }while(i<=10);

    getch();
    return 0;
}
```

`while loop Vs do...while loop`

## while loop(Entry controlled) Vs do...while loop(Exit Controlled)

Compiled by ab

While loop	Do-while loop(Exit controlled loop)
It is entry controlled loop	It is exit controlled loop.
Test condition is evaluated at beginning of the loop execution.	Test condition is evaluated at the end of the body of the loop.
The body of the loop will execute only if the test condition is true	The body of the loop will execute at least once without depending on the test condition.
<b>Syntax:</b> while(test condition) { Body of the loop }	<b>Syntax:</b> do { Body of the loop }while(test condition);

# while loop(Entry controlled) Vs do...while loop(Exit Controlled)

Compiled by ab

While loop	Do-while loop(Exit controlled loop)
<pre>graph TD; Entry(( )) --&gt; TC{Test Condition}; TC -- True --&gt; Body[while loop body]; Body --&gt; TC; TC -- False --&gt; Exit[Loop terminates];</pre>	<pre>graph TD; Entry(( )) --&gt; Body[do...while Loop Body]; Body --&gt; TC{Test Condition}; TC -- true --&gt; Body; TC -- false --&gt; Exit[Loop Terminates];</pre>
Write an example.	Write an example.



## Repetition/Iteration/Loop Control Instructions : `for` loop

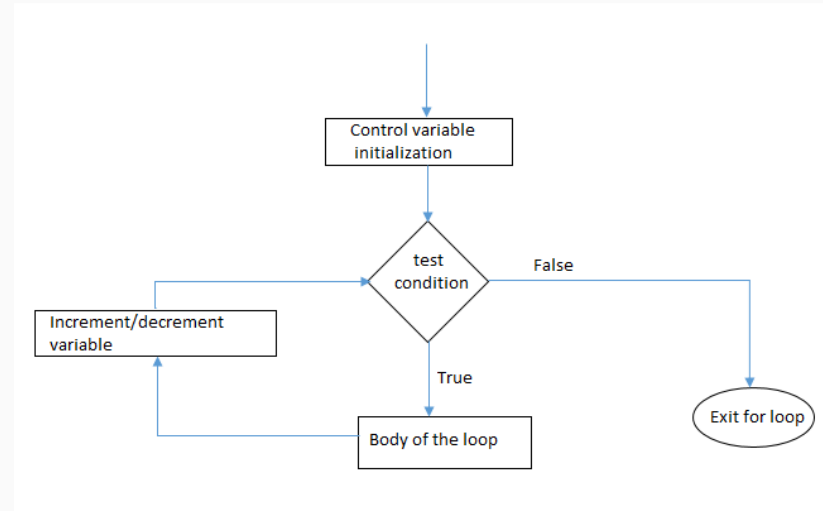
# Repetition/Iteration/Loop Control Instructions : `for` loop

Compiled by ab

- A **for loop** in C is a control flow statement used to repeatedly execute a block of code a specific number of times. It is particularly useful when the number of iterations is known in advance.
- The syntax of a **for loop** includes initialization, a condition to check before each iteration, and an increment/decrement operation.

## Syntax:

```
for(initialization; test condition; increment/decrement)
{
    body of the loop
}
```



## for loop Compiled by ab

**WAP to print 10 natural numbers using for loop.**

```
#include<stdio.h>
#include<conio.h>
int main()
{
    int i;
    for(i=1;i<=10;i++)
    {
        printf("%d\n",i);
    }
    getch();
    return 0;
}
```

## for loop Compiled by ab

**WAP to print numbers  
from 200 to 500 which  
are divisible by 5 and  
find their sum.**

```
#include<stdio.h>
#include<conio.h>
int main()
{
    int i,sum=0;
    for(i=200;i<=500;i++)
    {
        if(i%5==0)
        {
            printf("%d\n",i);
            sum=sum+i;
        }
    }
    printf("The sum is %d",sum);
    getch();
    return 0;
}
```

## Repetition/Iteration/Loop Control Instructions : Nested loop

# Repetition/Iteration/Loop Control Instructions : Nested Loop

Compiled by ab

A loop inside another loop is called nesting of loops. There can be any number of loops inside one another with any combinations depending on the complexity of program.eg. A for loop inside a while loop or while loop inside a for loop.

## Nested while loop

```
while (test-condition1)
{
    Statement (s);
    while (test-condition2)
    {
        statement (s);
        . . . . .
    }
    . . . . .
}
```

## Nested do-while loop

```
do
{
    Statement (s);
    do
    {
        Statement (s);
        . . . . .
    }while (test-condition2);
    . . . . .
}while (test-condition1);
```

# Repetition/Iteration/Loop Control Instructions : Nested Loop

Compiled by ab

A loop inside another loop is called nesting of loops. There can be any number of loops inside one another with any combinations depending on the complexity of program.eg. A for loop inside a while loop or while loop inside a for loop.

## Nested for loop

```
for(initialization;testcondition;increment/decrement)
{
    Statement(s);
    for(initialization;testcondition;increment/decrement)
    {
        Statement(s);
    }
}
```

# Nested loop

WAP to print the  
following output:

```
*
*   *
*   *   *
*   *   *   *
*   *   *   *   *
```

## Example: Nested for loop

```
#include<stdio.h>
#include<conio.h>

int main()
{
    int i,j;

    for(i=1;i<=5;i++)
    {
        for(j=1;j<=i;j++)
        {
            printf("*\t");
        }

        printf("\n");
    }

    getch();
    return 0;
}
```

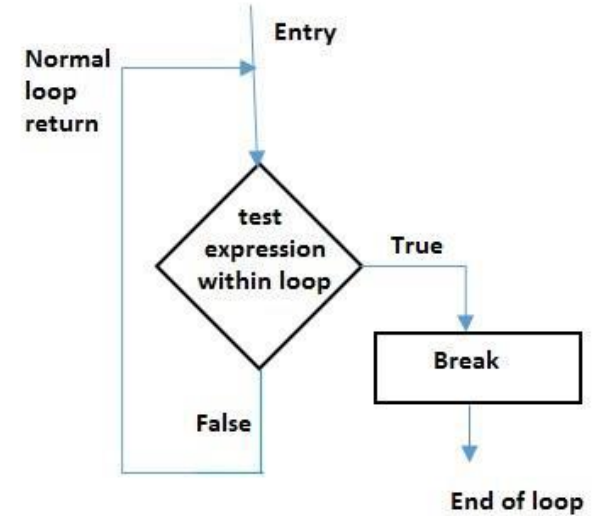


## JUMP statements

- 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.
- There are basically three types of jump statements:
  1. Break
  2. Continue
  3. Goto

## i) Break statement

- The break statement is used inside the loop or switch statement.
- The execution of break statement will abort the loop and continue to execute statements followed by loop.
- Similarly, execution of break statement causes bypass the rest of statement in switch and takes control out of the switch statement.



Flowchart

## Break statement example

```
#include<stdio.h>
#include<conio.h>

int main()
{
    int i;
    for(i=1;i<=5;i++)
    {
        printf("%d\t",i);
        if(i==3)
        {
            break;
        }
    }

    printf("\nloop terminates here.");
    getch();
    return 0;
}
```

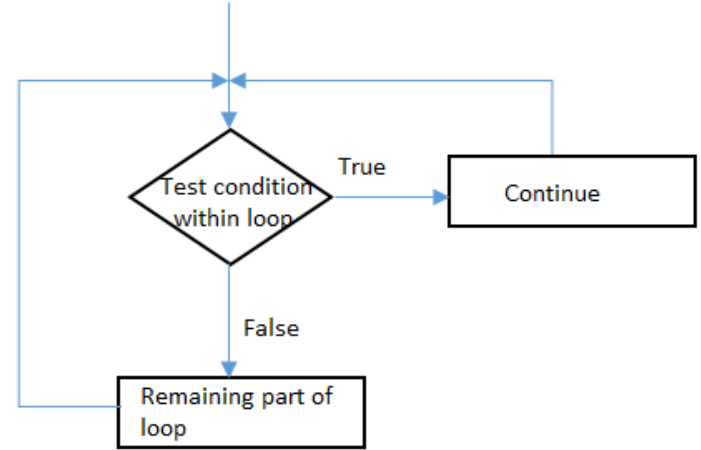
### Output:

1 2 3

Loop terminates here.

## ii) Continue statement

When continue statement is encountered inside the loop, it skips the execution of statements specified within the body of the loop and control automatically passes to the next iteration of the loop.



Flowchart

## Continue statement example

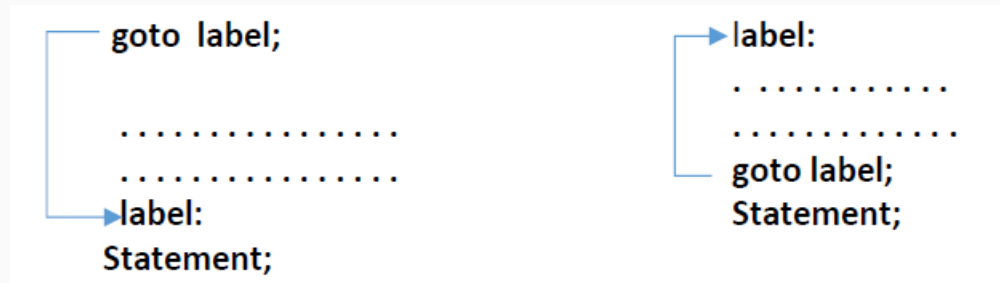
```
#include<stdio.h>
#include<conio.h>
int main()
{
    int i;
    for(i=1;i<=5;i++)
    {
        if(i==3)
        {
            continue;
        }
        printf("%d\t",i);
    }
    getch();
    return 0;
}
```

**Output:**  
1 2 4 5

## iii) goto statement

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

### Syntax:



**Forward Jump**

**Backward Jump**

where, label is an identifier that is used to label the target statement to which control will be transferred.

## goto statement example

```
#include<stdio.h>
#include<conio.h>
int main()
{
    int a,b;
    printf("Enter two numbers\n");
    scanf("%d%d",&a,&b);
    if(a>b)
    {
        goto label1;
    }
    else
    {
        goto label2;
    }
    label1:
    printf("Greatest number=%d",a);
    return;
    label2:
    printf("Greatest number=%d",b);
    getch();
    return 0;
}
```





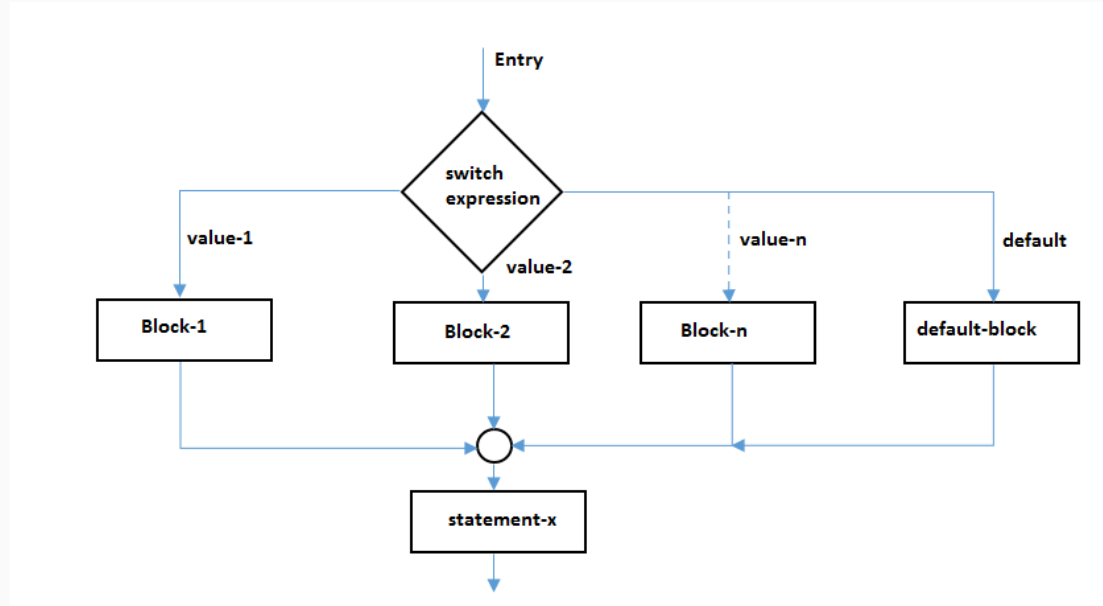
- The control statement that allows us to make a decision from number of choices is called switch case statement.
- The switch case statement successively test the value of a given variables (an expression) with a list of case value (integer or character constants)
- When match is found, the statement associated with that case is executed.
- If none of the case value matches the expression then default statement is executed.

**General form:**

```
switch(expression)
{
case value-1:
block-1;
break;
case value-2:
block-2;
break;
.....
default:
default-block;
break;
}
statement-x;
```

# switch case statement

Compiled by ab



## switch statement example:

WAP to display the corresponding days of a week according to the numbers entered.

```
#include<stdio.h>
#include<conio.h>

int main()
{
    int day;
    printf("Enter the numeric day of week\n");
    scanf("%d",&day);
    switch (day)
    {
        case 1:
            printf("Day is Sunday") ;
            break;
        case 2:
            printf("Day is Monday");
            break;
        case 3:
            printf("Day is Tuesday");
            break;
        case 4:
            printf("Day is Wednesday");
            break;
```

## switch statement example:

WAP to display the corresponding days of a week according to the numbers entered.

```
case 5:
printf("Day is Thursday");
break;
case 6:
printf("Day is Friday");
break;
case 7:
printf("Day is Saturday");
break;
default:
printf("Invalid choice!");
}
getch();
return 0;
}
```

WAP to display the following menu and perform the following operations.

1. Find the simple interest
2. Convert degree Celsius to Fahrenheit
3. Convert character into ASCII code
4. Find the area of circle
5. Exit from the program and perform above operation until user want to exit.

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
int main()
{
    int choice;
    float p,t,r,i,C,f,area;
    char ch;
    while(1)
    {
        printf("\nMenu");
        printf("\n1.Find simple intrest");
        printf("\n2.convert celcius of fahreheit");
        printf("\n3.convert character to ASCII code");
        printf("\n4.Find area of circle");
        printf("\n5.Exit from program");
        printf("\nEnter your choice\n");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
                printf("Enter Principal,Time and Rate\n");
                scanf("%f%f%f",&p,&t,&r);
                i=(p*t*r)/100;
                printf("simple intrest is %f",i);
                break;
```

WAP to display the following menu and perform the following operations.

1. Find the simple interest
2. Convert degree Celsius to Fahrenheit
3. Convert character into ASCII code
4. Find the area of circle
5. Exit from the program and perform above operation until user want to exit.

```
case 2:
printf("Enter the temperature in celcius\n");
scanf("%f",&c);
f=1.8*c+32;
printf("Converted temp is %f",f);
break;
```

```
case 3:
printf("Enter a character\n");
fflush(stdin);
scanf("%c",&ch);
printf("The corresponding ASCII code is %d",ch);
break;
```

```
case 4:
printf("Enter the radius of circle");
scanf("%f",&r);
area=3.14*r*r;
printf("Area=%f",area);
break;
```

```
case 5:
exit(0);
default:
printf("Wrong choice!");
}
}
getch();
return 0;
}
```

## Some Questions to Practice



## Question 1

WAP to find factorial of a given number

```
#include<stdio.h>
#include<conio.h>
int main()
{
    int num,i,fact = 1;
    printf("Enter a number \n");
    scanf("%d", &num);
    for (i= 1; i <= num; i++)
    {
        fact = fact * i;
    }
    printf("Factorial of given number= %d \n", fact);
    getch();
    return 0;
}
```

## Question 2

WAP to enter numbers  
until user press zero(0)  
and find the sum of  
supplied numbers

```
#include<stdio.h>
#include<conio.h>
int main()
{
    int sum=0,num;
    do
    {
        printf("Enter number");
        scanf("%d",&num);
        sum=sum+num;
    }while(num!=0);
    printf("Total sum of Entered number=%d",sum);
    getch();
    return 0;
}
```

### Question 3

WAP to check the given number is palindrome or not.

```
#include<stdio.h>
#include<conio.h>
int main()
{
    int num,rem,rev=0,a;
    printf("Enter the number\n");
    scanf("%d",&num);
    a=num;
    while (num!=0)
    {
        rem=num%10;
        rev=rev*10+rem;
        num=num/10;
    }
    if (a==rev)
    {
        printf("Entered number is palindrome");
    }
    else
    {
        printf("Entered number is not palindrome");
    }

    getch();
    return 0;
}
```

## Question 4

WAP to Check the given  
number is Armstrong  
number or not

```
#include <stdio.h>
#include<conio.h>
#include <math.h>
int main()
{
    int num,orgnum,rem,sum=0,n=0;
    printf("Enter the number: ");
    scanf("%d", &orgnum);
    num=orgnum;

    while (num != 0)
    {
        num =num/10;
        ++n;
    }

    num=orgnum;
```

## Question 4

WAP to Check the given  
number is Armstrong  
number or not

```
while (num != 0)
{
    rem = num%10;
    sum = sum+pow(rem, n);
    num=num/10;
}
if(sum == orgnum)
{
    printf("Given number is armstrong number");
}
else
{
    printf("Given number is not an Armstrong
number");
}
getch();
return 0;
}
```

## Question 5

WAP to find whether the entered year is leap year or not.

```
#include<stdio.h>
#include<conio.h>
int main()
{
    int year;
    printf("Enter the year\n");
    scanf("%d",&year);
    if (((year % 4 == 0) && (year % 100 != 0)) || (year%400 == 0))
    {
        printf("%d is a leap year", year);
    }
    else
    {
        printf("%d is not a leap year", year);
    }
    getch();
    return 0;
}
```

## Question 6

Write a program and draw a flowchart to read a positive integer value and compute the following sequence .If the number is even, half it ,if it is odd ,multiply 3 and add 1 print the result. If the input value is less than 1, print a message containing word “ERROR”.

```
#include<stdio.h>
#include<conio.h>
int main()
{
    int num;
    printf("Enter the integer number\n");
    scanf("%d", &num);
    if(num<1)
    {
        printf("ERROR");
    }
    else
    {
        if (num%2==0)
        {
            num=num/2;
        }
        else
        {
            num=num*3+1;
        }
    }
    printf("Number after result=%d", num);
}
getch();
return 0;
}
```

## Question 6

Write a program and draw a flowchart to read a positive integer value and compute the following sequence .If the number is even, half it ,if it is odd ,multiply 3 and add 1 print the result. If the input value is less than 1, print a message containing word “ERROR”.

```
#include<stdio.h>
#include<conio.h>
int main()
{
    int num;
    printf("Enter the integer number\n");
    scanf("%d", &num);
    if(num<1)
    {
        printf("ERROR");
    }
    else
    {
        if (num%2==0)
        {
            num=num/2;
        }
        else
        {
            num=num*3+1;
        }
    }
    printf("Number after result=%d", num);
}
getch();
return 0;
}
```



## Common Pitfalls: Infinite loops, Misplacement of Conditionals

An infinite loop occurs when the loop condition never becomes false, causing the loop to execute indefinitely.

**Missing or Incorrect Update Statement:** Forgetting to update the loop control variable.

```
int i = 0;
while (i < 10) {
    // Missing i++ or similar update
    printf("%d\n", i);
}
```

**Incorrect Condition:** The condition may always evaluate to true.

```
int i = 0;
while (i >= 0) {
    printf("%d\n", i);
    i++;
}
```

## **Infinite Loops Prevention:**

- Ensure the loop condition will eventually become false.
- Double-check the loop control variable is being updated correctly.
- Use for loops when the number of iterations is known, as they bundle initialization, condition, and update in one place.

- Placing conditional statements (like if, else if, else) in the wrong order or incorrectly nesting them can lead to logical errors.

## Common Causes:

**Overlapping Conditions:** Conditions that are not mutually exclusive can cause unexpected behavior.

```
int x = 10;
if (x > 5) {
    printf("x is greater than 5\n");
} else if (x > 8) {
    printf("x is greater than 8\n");
}
```

The second condition ( $x > 8$ ) will never be evaluated because the first condition ( $x > 5$ ) is always true for  $x = 10$ .

**Fix:** Change the logic so both conditions are checked properly.

- These errors occur when a loop iterates one time too many or one time too few, often due to incorrect boundary conditions.

## Common Causes:

### **Incorrect Loop Bounds: Using `<=` instead of `<` or vice versa**

```
for (int i = 0; i <= 10; i++) {  
    printf("%d\n", i); // Prints 0 to 10 (11 iterations)  
}
```

If the intention was to print 0 to 9, the condition should be `i < 10`.

## Prevention:

- Carefully check loop conditions, especially when dealing with arrays.
- Use constants or sizeof to determine array sizes dynamically.
- Test with boundary values to ensure correct behavior.

THANK YOU

Any Queries ?