جامعة ذي قار
كلية التربية للعلوم الصرفة
قسم علوم الحاسبات
المرحلة الاولى
مادة البرمجة المهيكلة

Lecture 1

Instructor: Zainab Rustum Mohsin

What Is Programming and Program Development Life Cycle ?

- Programming is **a process of problem solving**
- Step 1: Analyze the problem
  - Outline the problem and its requirements
  - Design steps (algorithm) to solve the problem

- Step 2: Implement the algorithm
  - Implement the algorithm in code
  - Verify that the algorithm works

- Step 3: Maintenance
  - Use and modify the program if the problem domain changes

Algorithm:

- Step-by-step problem-solving process

Lecturer : Zainab Rustum Mohsin

The Problem Analysis–Coding–Execution Cycle

- Understand the Overall problem
- Understand problem requirements
  - Does program require user interaction?
  - Does program manipulate data?
  - What is the output?
- If the problem is complex, divide it into subproblems
  - Analyze each subproblem as above

The Problem Analysis–Coding–Execution Cycle (cont'd.)

- Run code through compiler
- If compiler generates errors
  - Look at code and remove errors
  - Run code again through compiler
- If there are no syntax errors
  - Compiler generates equivalent machine code
- Linker links machine code with system resources

The Problem Analysis–Coding–Execution Cycle (cont'd.)

- Once compiled and linked, loader can place program into main memory for execution
- The final step is to execute the program
- Compiler guarantees that the program follows the rules of the language
  - Does not guarantee that the program will run correctly

The Language of a Computer

- **Machine language:** language of a computer
- **Binary digit (bit):**
  - The digit 0 or 1
- **Binary code:**
  - A sequence of 0s and 1s
- **Byte:**
  - A sequence of eight bits

The Evolution of Programming Languages

- Assembly language instructions are mnemonic
- Assembler: translates a program written in assembly language into machine language

TABLE 1-2 Examples of Instructions in Assembly Language and Machine Language

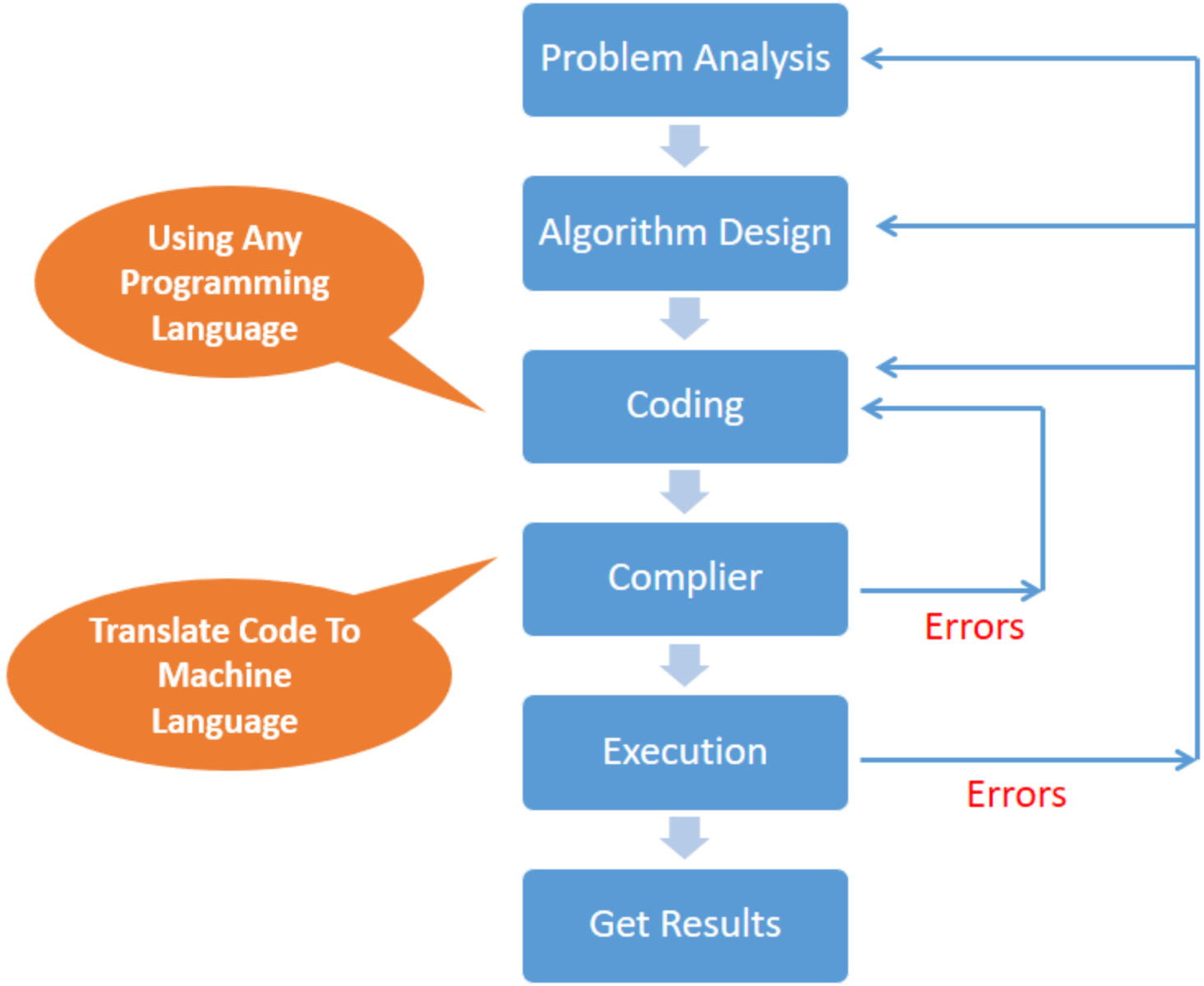
Assembly Language	Machine Language
LOAD	100100
STOR	100010
MULT	100110
ADD	100101
SUB	100011

The Evolution of Programming Languages

- High-level languages include Basic, FORTRAN, COBOL, Pascal, C, C++, C#, and Java

- **Compiler**: translates a program written in a high-level language to machine language

The Problem Analysis–Coding–Execution Cycle



Think

Then

Write Code

جامعة ذي قار
كلية التربية للعلوم الصرفة
قسم علوم الحاسبات
المرحلة الاولى
مادة البرمجة المهيكلة

Lecture 2

Instructor: Zainab Rustum Mohsin

ALGORITHMS AND FLOWCHARTS

A typical programming task can be divided into two phases:

1• Problem solving phase

produce an ordered sequence of steps that describe solution of problem this sequence of steps is called an ***algorithm***

2• Implementation phase

Implement the program in some programming language

ALGORITHM

The word “algorithm” relates to the name of the mathematician Al-khowarizmi, which means a procedure or a technique.Software Engineer commonly uses an algorithm for planning and solving the problems. An algorithm is a sequence of steps to solve a particular problem or algorithm is an ordered set of unambiguous steps that produce a result and terminates in a finite time.

Algorithm has the following characteristics

- **Input**: An algorithm may or may not require input
- **Output:** Each algorithm is expected to produce at least one result
- **Definiteness**: Each instruction must be clear and unambiguous.
- **Finiteness**: If the instructions of an algorithm are executed, the algorithm should terminate after finite number of steps

Lecturer : Zainab Rustum Mohsin

The algorithm and flowchart include following three types of control structures.

1. Sequence: In the sequence structure, statements are placed one after the other and the execution takes place starting from up to down.

2. Branching (Selection): In branch control, there is a condition and according to a condition, a decision of either TRUE or FALSE is achieved. In the case of TRUE, one of the two branches is explored; but in the case of FALSE condition, the other alternative is taken. Generally, the 'IF-THEN' is used to represent branch control.

3. Loop (Repetition): The Loop or Repetition allows a statement(s) to be executed repeatedly based on certain loop condition e.g. WHILE, FOR loops.

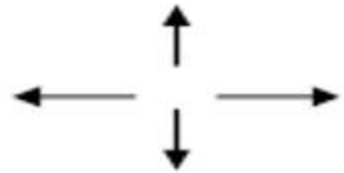
FLOWCHART

The first design of flowchart goes back to 1945 which was designed by John Von Neumann. Unlike an algorithm, Flowchart uses different symbols to design a solution to a problem. It is another commonly used programming tool. By looking at a Flowchartone can understand the operations and sequence of operations performed in a system. Flowchart is often considered as a blueprint of a design used for solving a specific problem.

Advantages of flowchart

- Flowchart is an excellent way of communicating the logic of a program.
- Easy and efficient to analyze problem using flowchart.
- Makes program development process easier.
- The flowchart makes program or system maintenance easier.
- It is easy to convert the flowchart into any programming language code.

. To draw a flowchart following standard symbols are use

Symbol Name	Symbol	function
Oval		Used to represent start and end of flowchart
Parallelogram		Used for input and output operation
Rectangle		Processing: Used for arithmetic operations and data-manipulations
Diamond		Decision making. Used to represent the operation in which there are two/three alternatives, true and false etc
Arrows		Flow line Used to indicate the flow of logic by connecting symbols
Circle		Page Connector

different operations:

‘+’ for Addition

‘-’ for Subtraction

‘*’ for Multiplication

‘/’ for Division and

‘←’ for assignment. For example A←X*3 means A will have a value of X*3.

EXAMPLE

Write an algorithm and draw a flowchart that will read your name and print it.

Pseudocode

1- START

2-Input the name

3- Print name

4- END

Problem2: Write an algorithm to read two numbers and find their sum.

Inputs to the algorithm:

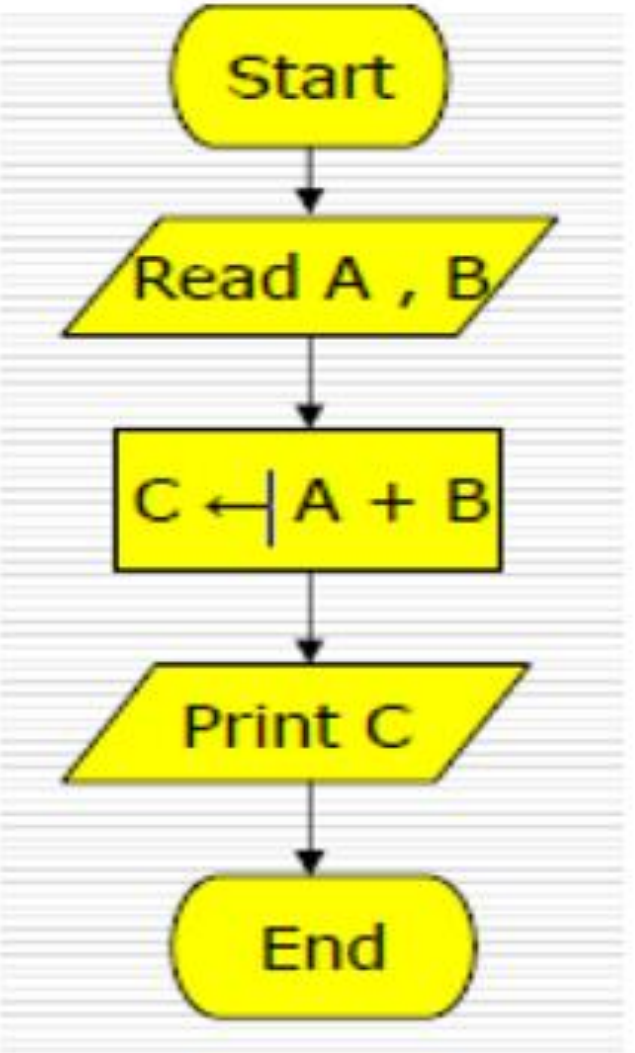
First num1.
Second num2.

Expected output:

Sum of the two numbers.

Algorithm:

1: Start
2: Read\input A
3: Read\input B
4: C ← A+B // calculation of sum
5: Print C
6: End



Algorithm & Flowchart to find Area and Perimeter of Square

L : Side Length of Square
AREA : Area of Square
PERIMETER : Perimeter of Square

Algorithm

1 Start

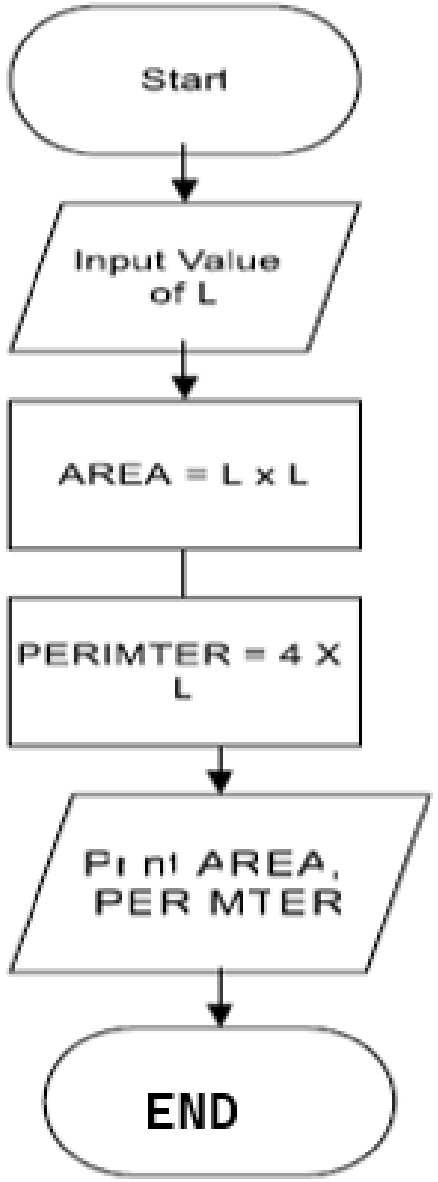
2 Input Side Length of Square say L

3 Area = L x L

4 PERIMETER = 4 x L

5 **PRINT** AREA, PERIMETER

6 **END**



Algorithm & Flowchart to find Area and Perimeter of Rectangle

L : Length of Rectangle
B : Breadth of Rectangle
AREA : Area of Rectangle
PERIMETER : Perimeter of Rectangle

Algorithm

1 Start

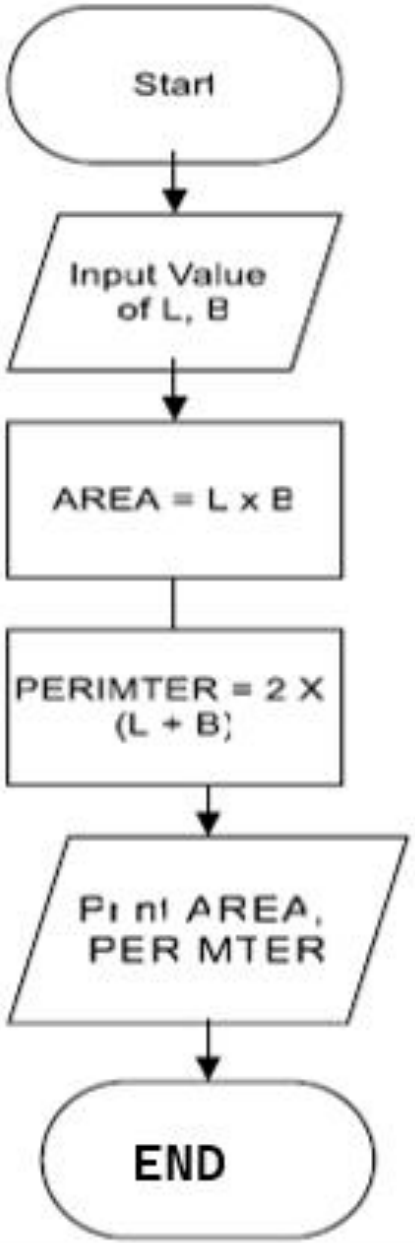
2 Input L, B

3 Area = L x B

4 PERIMETER = 2 x (L + B)

5 **PRINT** AREA, PERIMETER

6 **END**



Algorithm & Flowchart to Swap Two Numbers using Temporary Variable

Lecturer : Zainab Rustum Mohsin

Algorithm

1 Start

2 Input Two Numbers Say NUM1,NUM2

3 **PRINT** Before Swap Values NUM1, NUM2

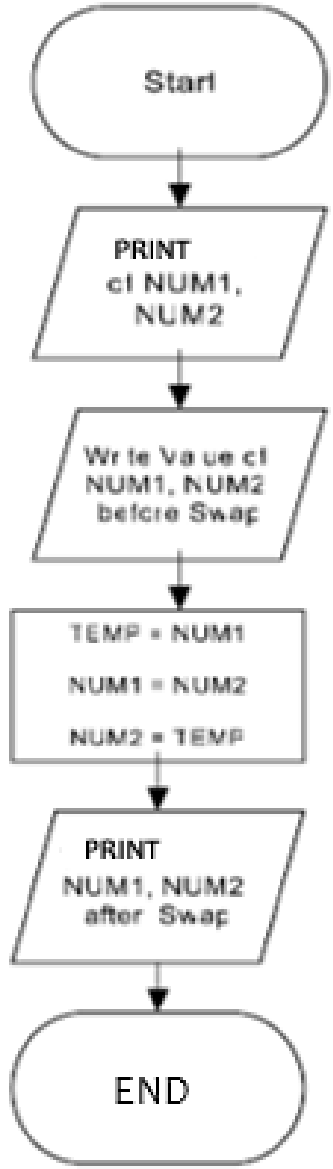
4 TEMP = NUM1

5 NUM1 = NUM2

6 NUM2 = NUM1

7 **PRINT** After Swap Values NUM1,NUM

8 **END**



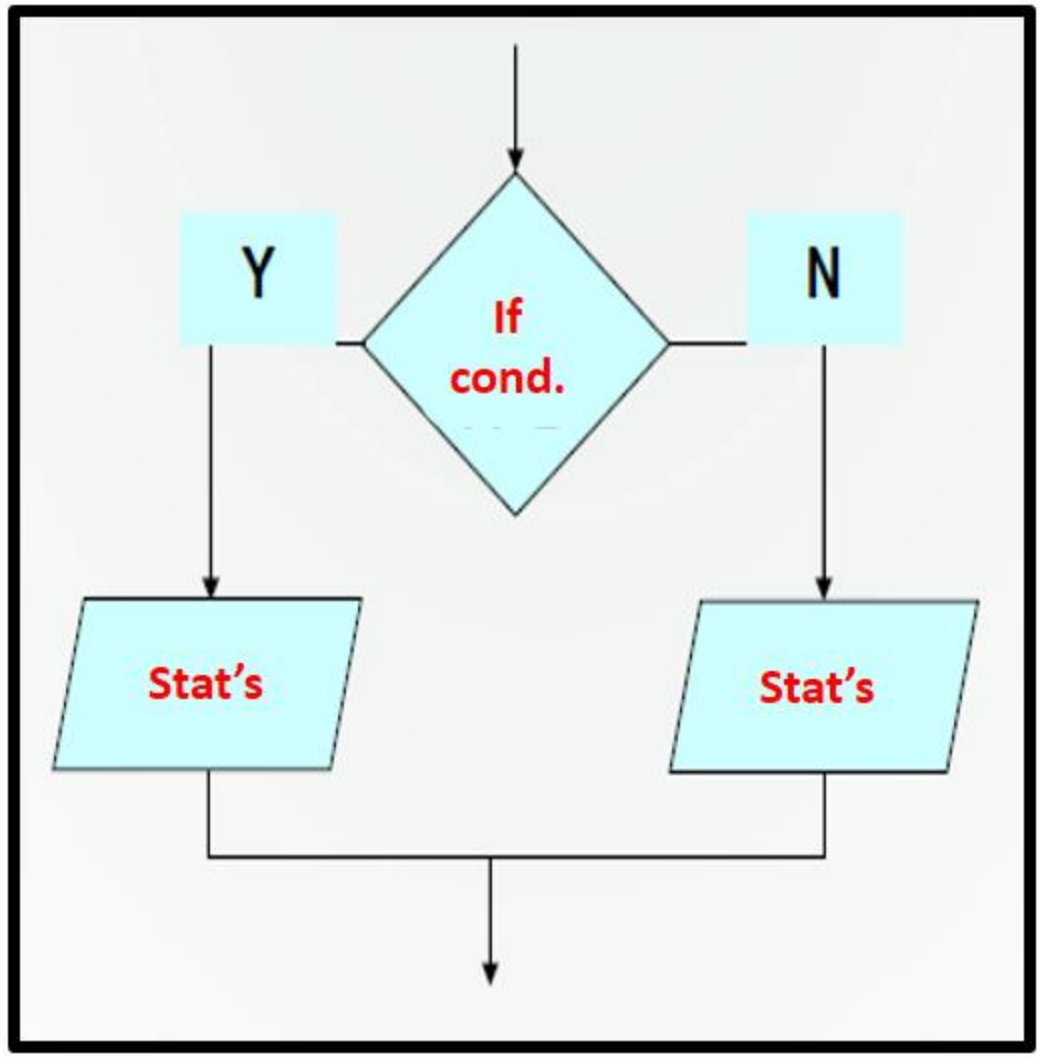
جامعة ذي قار
كلية التربية للعلوم الصرفة
قسم علوم الحاسبات
المرحلة الاولى
مادة البرمجة المهيكلة

Lecture 3

Instructor: Zainab Rustum Mohsin

DECISION STRUCTURES

The *branch* refers to a binary decision based on some condition. If the condition is true, one of the two branches is explored; if the condition is false, the other alternative is taken. This is usually represented by the 'if-then' construct in pseudo-codes and programs. In flowcharts, this is represented by the diamond-shaped decision box. This structure is also known as the *selection* structure.



IF–THEN–ELSE STRUCTURE

- The structure is as follows

If condition then

true alternative

else

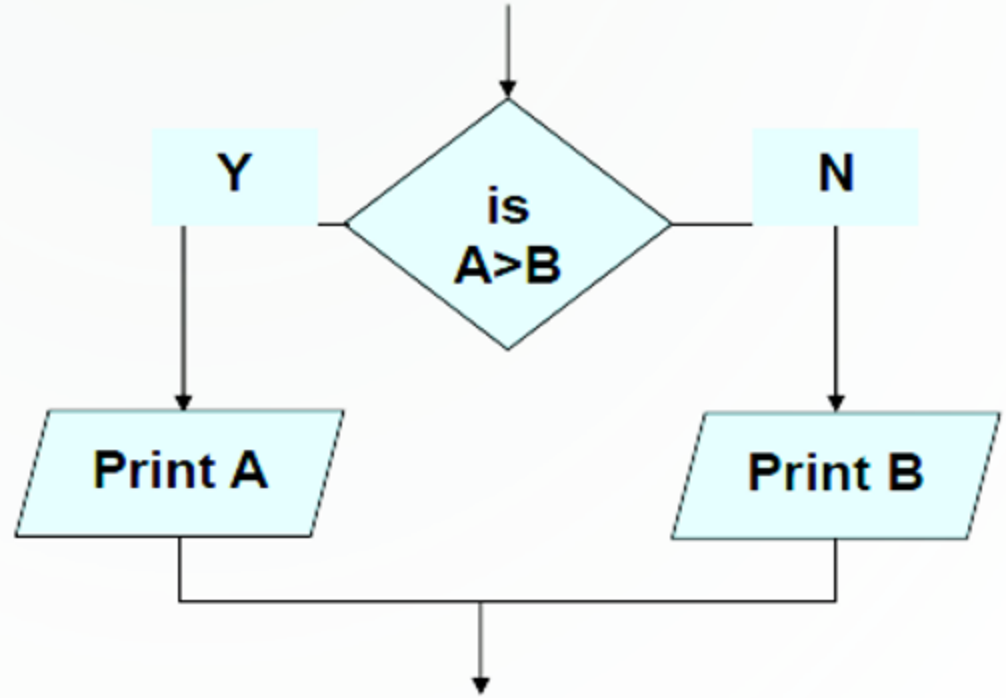
false alternative

endif

IF–THEN–ELSE STRUCTURE

- The algorithm for the flowchart is as follows:

```
If A>B then
   print A
else
   print B
endif
```



Mathematical Operators:

Operator	Meaning	Example
+	Addition	A + B
-	Subtraction	A – B
*	Multiplication	A * B
/	Division	A / B
^	Power	A^3 for A^3
%	Reminder	A % B

Relational Operators

Operator	Meaning	Example
<	Less than	A < B
<=	Less than or equal to	A <= B
= or ==	Equal to	A = B
# or !=	Not equal to	A # B or A !=B
>	Greater than	A > B
>=	Greater tha or equal to	A >= B

Logical Operators

Operator	Example	Meaning
AND	A < B AND B < C	Result is True if both A<B and B<C are true else false
OR	A< B OR B < C	Result is True if either A<B or B<C are true else false
NOT	NOT (A >B)	Result is True if A>B is false else true

Selection control Statements

Selection Control	Example	Meaning

IF (Condition) Then
…
ENDIF
	IF (X > 10) THEN
	Y=Y+5
	ENDIF
		If condition X>10 is True execute the statement between THEN and ENDIF

IF (Condition) Then
…
ELSE
…..
	IF (X > 10) THEN
	Y=Y+5
	ELSE
	Y=Y+8
	Z=Z+3
		If condition X>10 is True execute the statement between THEN and ELSE otherwise execute the statements between ELSE

Loop control Statements

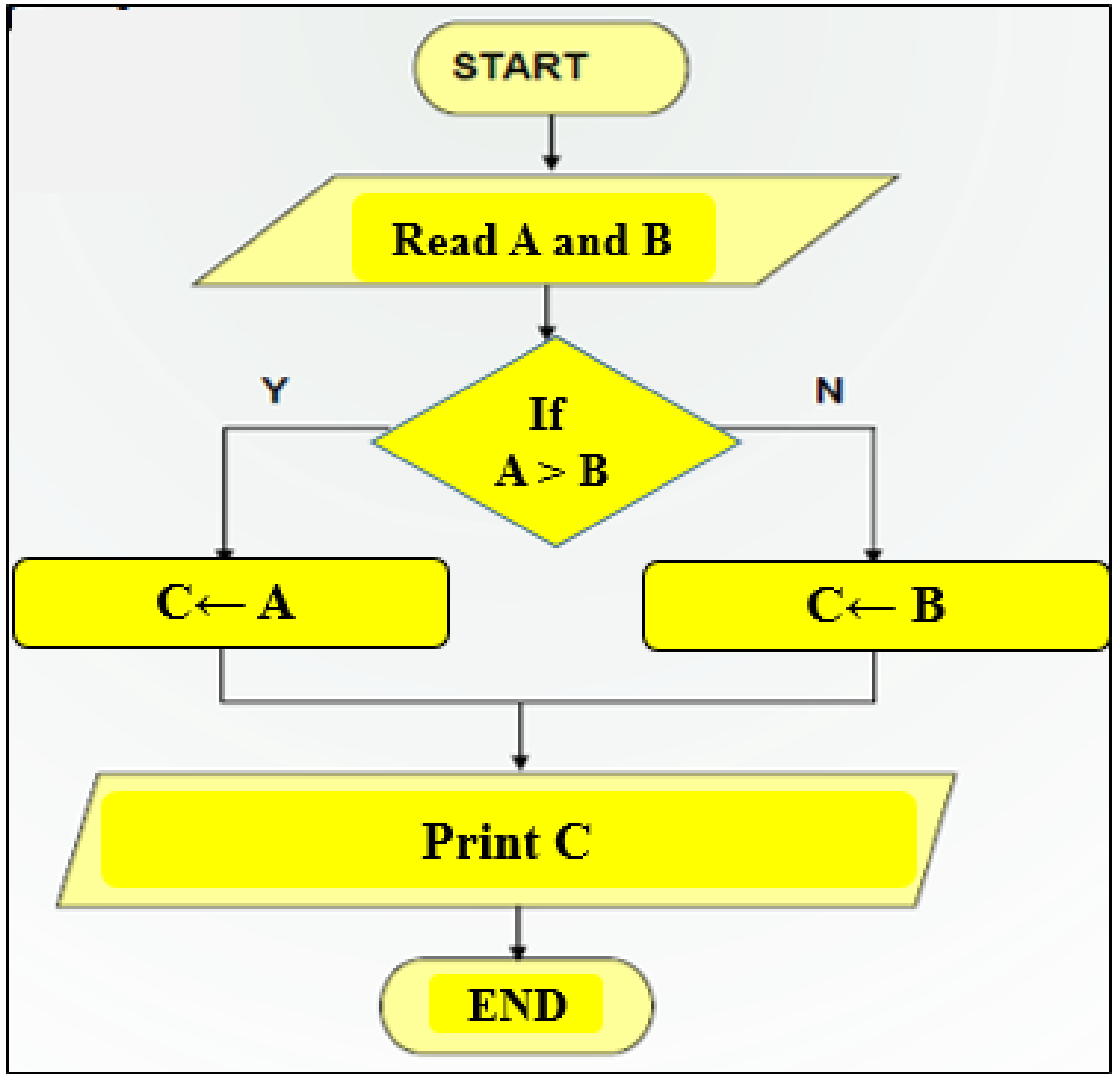
Selection Control	Example	Meaning

WHILE (Condition)
DO
..
..
ENDDO
	WHILE (X < 10)
	DO
	print x
	x=x+1
	ENDDO
		Execute the loop as long as the condition is TRUE

DO …. … UNTILL (Condition)	DO print x x=x+1 UNTILL (X >10)	Execute the loop as long as the condition is false

Example : **write algorithm to find the greater number between two numbers**

1: Start
2: Read A and B
3: If A > B then C← A
else C← B
endif
4: Print C
5: End

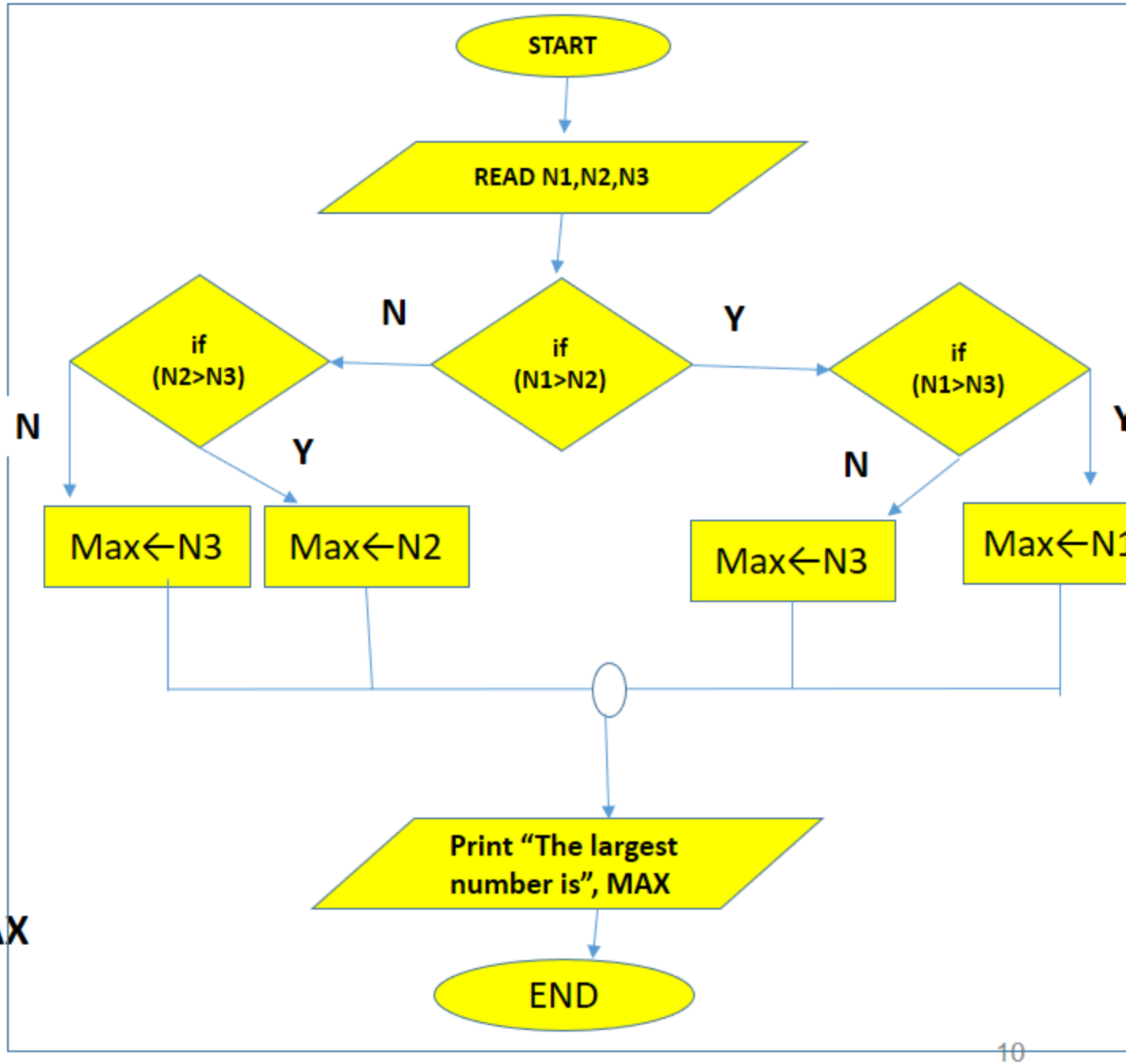


NESTED IF 'S

- One of the alternatives within an IF–THEN–ELSE statement
  - may involve further IF–THEN–ELSE statement

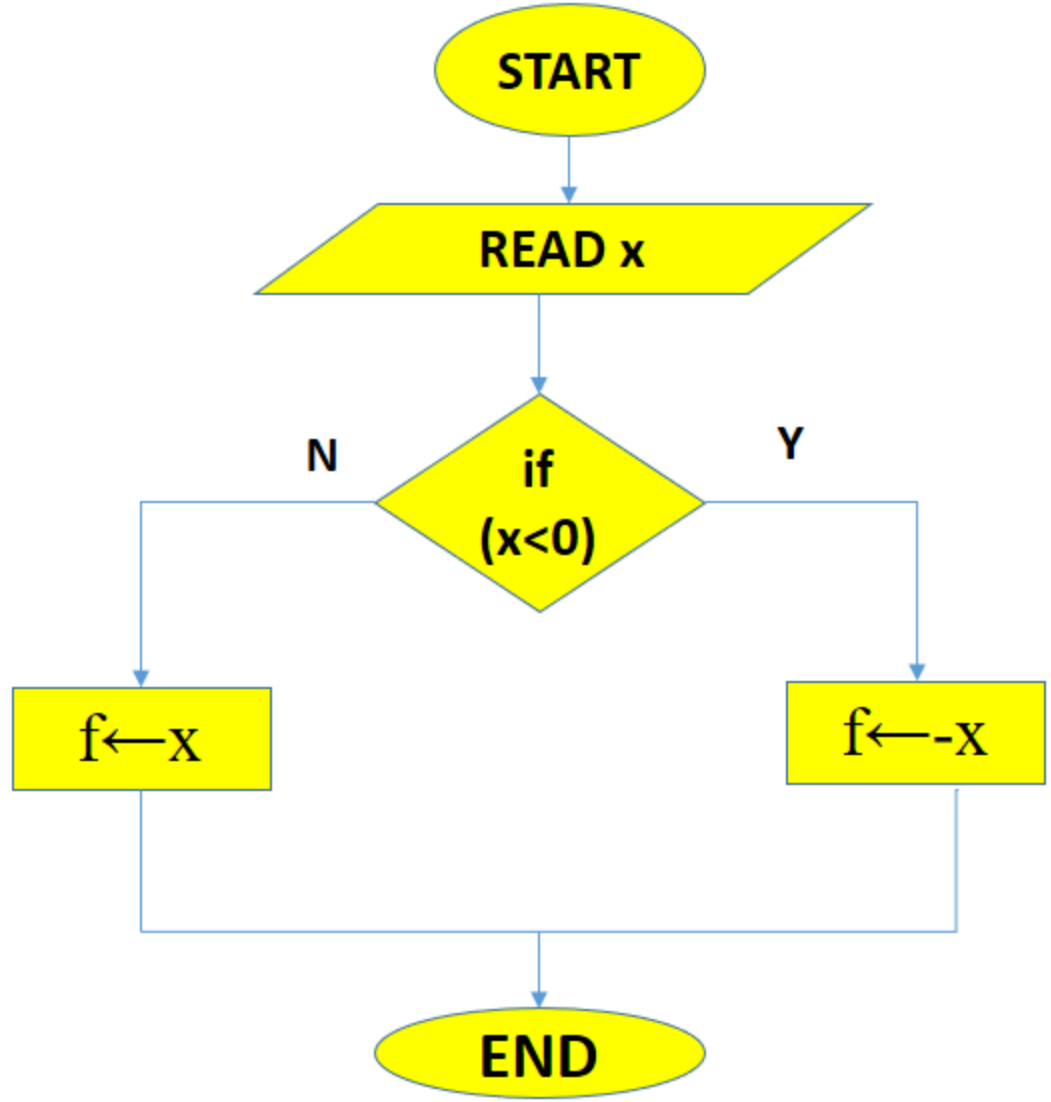
EX: Write an algorithm that reads **three** numbers and prints the value of the largest number.

```
1: START
2: Read  N1, N2, N3
3: if (N1>N2) then
      if (N1>N3) then
           MAX ←N1
         else
           MAX← N3
       endif
    else
      if (N2>N3) then
          MAX ← N2
        else
          MAX←N3
      endif
    endif
4: Print "The largest number is", MAX
5: END
```



EX: write algorithm to find the result of equation: $f(x) = \begin{cases} -x, & x < 0 \\ x, & x \geq 0 \end{cases}$

1: START
2: Read x
3: If x < 0 then f←-x
else f ← x
endif
4: Print f
5: END



LOOPING STRUCTURES

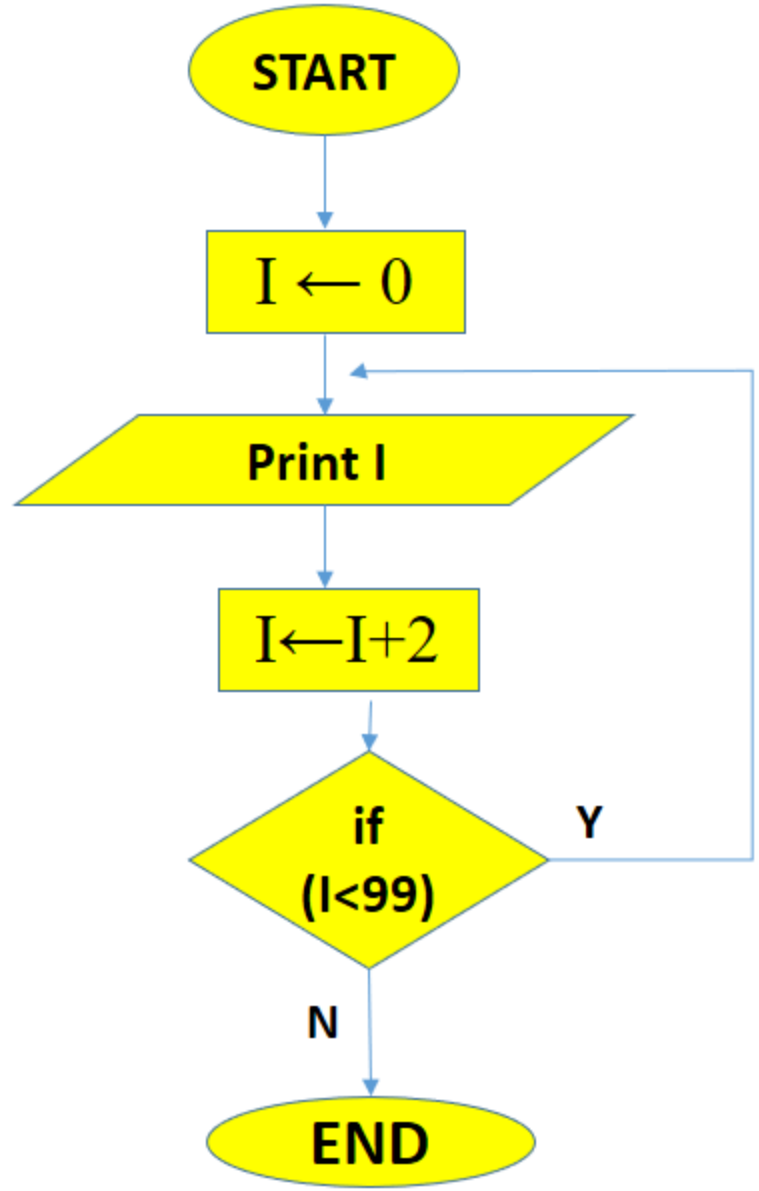
The loop allows a statement or a sequence of statements to be repeatedly executed based on some loop condition. It is represented by the ‘while’ and ‘for’ constructs in most programming languages, for unbounded loops and bounded loops respectively. (Unbounded loops refer to those whose number of iterations depends on the eventuality that the termination condition is satisfied; bounded loops refer to those whose number of iterations is known before-hand.) In the flowcharts, a **back arrow** hints the presence of a loop

A trip around the loop is known as iteration. You must ensure that the condition for the termination of the looping must be satisfied after some finite number of iterations, otherwise it ends up as an infinite loop, a common mistake made by inexperienced programmers. The loop is also known as the repetition structure.

GO TO statement also called unconditional transfer of control statement is used to transfer control of execution to another step/statement. . e.g. the statement GOTO n will transfer control to step/statement n.

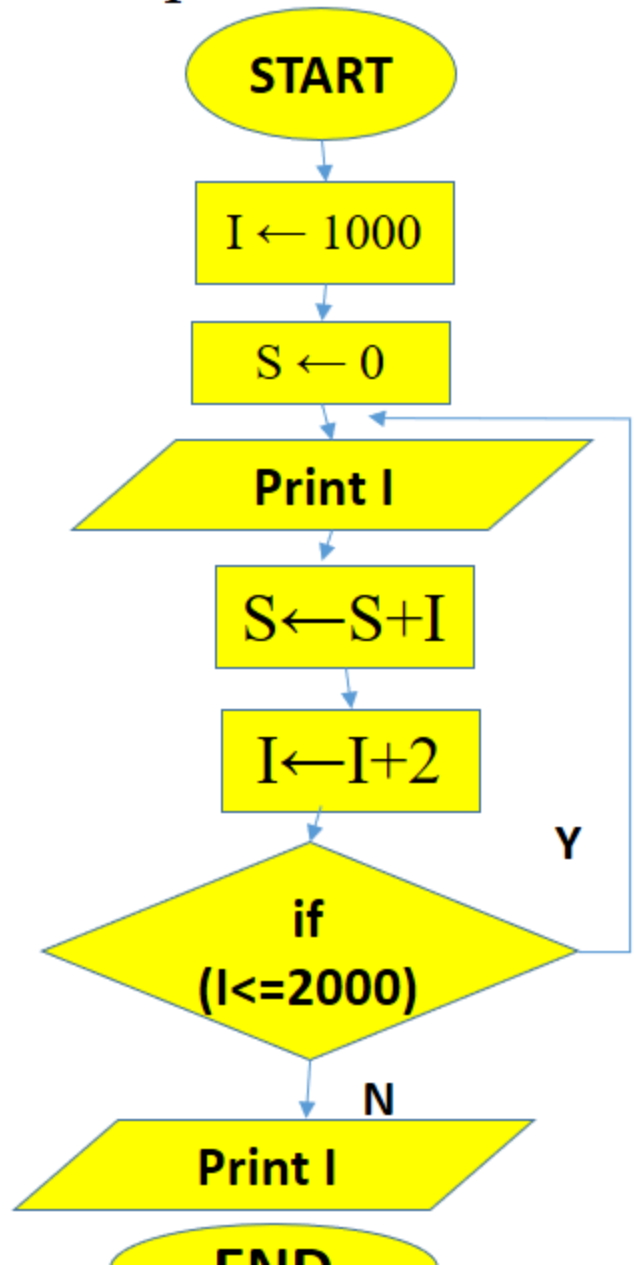
EX: An algorithm to calculate and print even numbers between 0 and 99

1: START
2: I ← 0
3: Print I
4: I ← I+2
5: If (I < 99) then go to line 3
6: END



EX: Design an algorithm which generates even numbers between 1000 and 2000 and then prints them in the standard output. It should also print total sum

1: START
2: I ← 1000
3: S ← 0
4: print I
5: S ← S + I
6: I ← I + 2
7: If (I <= 2000) then go to line 4
8: Print S
8. END



ASSIGNMENT

Design an algorithm which:

1: gets n ,as input and print odd numbers equal or less than n.

2: gets n, as its input ,then write the result of the series: $S = ½ + ¼ + \ldots + 1/n$

3: calculate the average from 5 exam scores

4: Find factorial of N?

5:determining prime number?

6: Draw a flowchart which generates first 50 items of the Fibonacci series: 0,1,1,2,3,5,8,13…….

Lecturer : Zainab Rustum Mohsin

جامعة ذي قار
كلية التربية للعلوم الصرفة
قسم علوم الحاسبات
المرحلة الاولى
مادة البرمجة المهيكلة

Lecture 4

Instructor: Zainab Rustum Mohsin

Basic Definitions

Programming language:

a set of rules, symbols, and special words used to write computer programs.

Computer program

Sequence of statements whose objective is to accomplish a task.

Syntax:

rules that specify which statements (instructions) are legal

Lecturer : Zainab Rustum Mohsin

What C++ ?

-C++ is a cross-platform language that can be used to create high-performance applications.

-C++ was developed by Bjarne Stroustrup, as an extension to the C language.

-C++ gives programmers a high level of control over system resources and memory.

- C++ was updated 4 major times in 2011, 2014, 2017, and 2020 to C++11, C++14, C++17, C++20.

-C++ is a popular programming language.

-C++ is used to create computer programs, and is one of the most used language in game development.

Lecturer : Zainab Rustum Mohsin

Computer System

Example 1

- **Write a program to find the Area of a rectangle**

The area of the Rectangle are given by the following formula:

Area = Rect Length * Rect Width.

Input :

Rectangle Length , Rectangle Width.

Processing :

Area = Rect Length * Rect Width.

Output :

Print Out The area.

Example 2

- **Write a program to find the perimeter and area of a square**

The perimeter and area of the square are given by the following formulas:

```
perimeter = Side Length * 4
area = Side Length * Side Length
```

Input:

Square Side Length

Processing:

perimeter = Side Length * 4

area = Side Length * Side Length

Output:

Print Out The Perimeter and Area.

Your First C++ Program

```
#include <iostream>
using namespace std;
int main()
{
  // This program

    cout << "My first C++ program." << endl;

    return 0;
}
```

Sample Run:

```
My first C++ program.
```

Interacting With User: Displaying Messages on Screen

- In C++ , we use Cout << " Text To be Displayed on The screen " ;
- To use Cout << , we must use
  - #include <iostream> ;
  - using namespace std;

- #include <iostream> notifies the preprocessor to include the contents of the input/output stream header file <iostream> in the program
- We can use the Escape Sequence to format the Displayed Text.

Escape sequence	Description
\n	Newline. Position the screen cursor to the beginning of the next line.
\t	Horizontal tab. Move the screen cursor to the next tab stop.
\r	Carriage return. Position the screen cursor to the beginning of the current line; do not advance to the next line.
\a	Alert. Sound the system bell.
\\	Backslash. Used to print a backslash character.
\'	Single quote. Use to print a single quote character.
\"	Double quote. Used to print a double quote character.

Lecturer : Zainab Rustum Mohsin

Interacting With User: Comments

- We can put comment on our programs using
- // to comment a single line

```
// this progrm is written
```

- /*

  Mulitple Lines

- */ to comment multiple lines

```
/* This program is written
On Monday 11/1/2012
*/
```

Interacting With User: Accept Input From User

- In C++ , we use Cin >> Variable; To accept an input from the user.
- To use Cin >>, we must use #include <iostream> ;
- `#include <iostream>` notifies the preprocessor to include in the program the contents of the input/output stream header file `<iostream>`.

- A variable is a location in the computer's memory where a value can be stored for use by a program.
- All variables must be **declared** with a **name** and a **data type** before they can be used in a program.

- Declaration

DataType Identifier;

```
Int     width ;
Float   salary ;
```

Lecturer : Zainab Rustum Mohsin

C++ Data Types

char	Character or small integer.	1byte	signed: -128 to 127 unsigned: 0 to 255
int	Integer.	4bytes	signed: -2147483648 to 2147483647 unsigned: 0 to 4294967295
short int (short)	Short Integer.	2bytes	signed: -32768 to 32767 unsigned: 0 to 65535
long int (long)	Long integer.	4bytes	signed: -2147483648 to 2147483647 unsigned: 0 to 4294967295
bool	Boolean value. It can take one of two values: true or false.	1byte	true or false
float	Floating point number.	4bytes	+/- 3.4e +/- 38 (~7 digits)
double	Double precision floating point number.	8bytes	+/- 1.7e +/- 308 (~15 digits)
long double	Long double precision floating point number.	8bytes	+/- 1.7e +/- 308 (~15 digits)

Example 1

- **Write a program to find the Area of a rectangle**

The area of the Rectangle are given by the following formula:

Area = Rect Length * Rect Width.

Input :

Rectangle Length , Rectangle Width.

Processing :

Area = Rect Length * Rect Width.

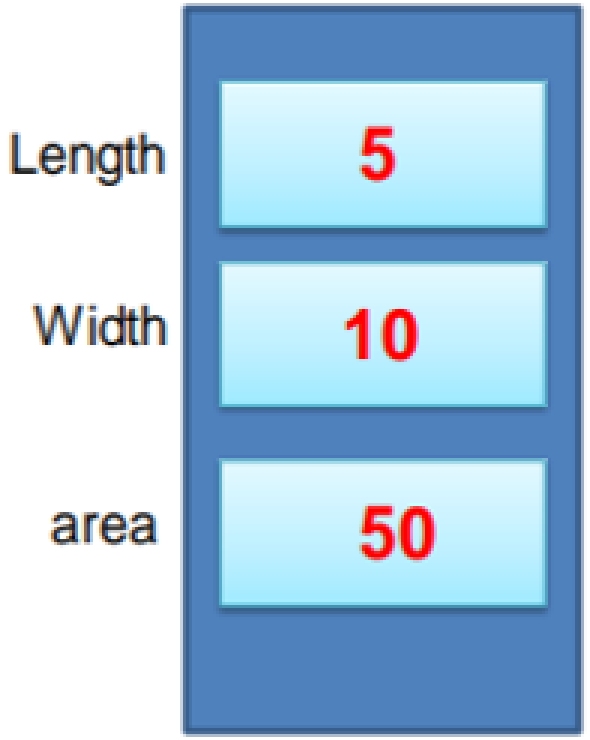
Output :

Print Out The area.

Lecturer : Zainab Rustum Mohsin

Working With Variable

```
Int  length ;
Int  width;
Int  area;

Cin>>Length;

Cin>>width;

Area =  Length * width ;
```



Declaring & Initializing Variables

- Initialization means to give a variable an initial value.
- Variables can be initialized when declared:

```
int first=13, second=10;
char ch=' ';
double x=12.6;
```

- All variables must be initialized before they are used in an arithmetic operatation
  - But not necessarily during declaration

Rules on Variable Names

- DO NOT use reserved words as variable names
  (e.g. if, else, int, float, case, for, …).
- The first character has to be a letter or underscore. It can not be a numeric digit.
- The second and the other characters of the name can be any letter, any number, or an underscore "_".

Examples

Some valid names:

my_name, m113_1, salary, bluemoon , _at

Some invalid names:

my name, my-name , 1stmonth , salary! , guns&roses ,

Frequently used data types

Data Types	Bytes Used
int	4 Bytes
short	2 Bytes
double	8 Bytes
unsigned	4 Bytes
Float	4 Bytes
Double	8 Bytes
Char	1 Byte

- The data type **unsigned** is used to represent positive integers.
- **float** and **double** data types for storing real numbers.

The **float** data type has a precision of seven digits

-This means after the decimal point you can have seven digits

-Example: *3.14159* *534.322344* *0.1234567*

- **The double** data type has a precision of fifteen digits

Example :

-3738.7878787878 3.141592653589790

0.123456789123456

Frequently used data types

- We can use *Scientific Notation* to represent real numbers that are very large or very small in value.
- The letters **e** or **E** is used to represent times **10** to the power.

Example:

- 1.23×10^{5} *is represented as* 1.23e5 *or* 1.23e+5 *or* 1.23E5
- 1×10^{-9} *is represented as* 1e-9

Example 1

- **Write a program to find the Area of a rectangle**

The area of the Rectangle are given by the following formula:

Area = Rect Length * Rect Width.

Input :

Rectangle Length , Rectangle Width.

Processing :

Area = Rect Length * Rect Width.

Output :

Print Out The area.

```
{
    // the input from the user
    int width;
    int length;
    int area;
    cout <<"please enter rectangle width \n";
    cin>>width;

    cout <<"please enter rectangle length \n";
    cin>>length;

    // processing
    area = width * length;

    // output
    cout << area;

return 0;
```

```
{
    // the input from the user
    int width = 0 , length ;

    cout <<"please enter rectangle width and length\n";
    cin>> width >> length;

    //cout <<"please enter rectangle length \n";
    //cin>>length;

    // processing
    //area = width * length;

    // output
    cout <<" Area of rectangle equal " << width * length << "\n" ;

retur
}
```

جامعة ذي قار
كلية التربية للعلوم الصرفة
قسم علوم الحاسبات
المرحلة الاولى
مادة البرمجة المهيكلة

Lecture 5

Instructor: Zainab Rustum Mohsin

Arithmetic Operations

C++ operation	C++ arithmetic operator	C++ expression
Addition	+	f + 7
Subtraction	-	p - c
Multiplication	*	b * m
Division	/	x / y
Modulus	%	r % s

| Arithmetic operators.

- Parentheses are used in C++ expressions in the same manner as in algebraic expressions.
- For example, to multiply **a** times the quantity **b + c**

we write **a * (b + c).**

- There is no arithmetic operator for exponentiation in C++,

so **x2** is represented as **x * x.**

Precedence of arithmetic operations

Operator(s)	Operation(s)	Order of evaluation (precedence)
( )	Parentheses	Evaluated first. If the parentheses are nested, the expression in the innermost pair is evaluated first. [*Caution:* If you have an expression such as (a + b) * (c - d) in which two sets of parentheses are not nested, but appear "on the same level," the C++ Standard does *not* specify the order in which these parenthesized subexpressions will be evaluated.]
*, /, %	Multiplication, Division, Modulus	Evaluated second. If there are several, they're evaluated left to right.
+ -	Addition Subtraction	Evaluated last. If there are several, they're evaluated left to right.

Precedence of arithmetic operators.

For example,

```
2 + 3 * 5 and (2 + 3) * 5
```

both have different meanings

Precedence of arithmetic operations

Lecturer : Zainab Rustum Mohsin

Example :

```
1.  y = 2 * 5 * 5 + 3 * 5 + 7;      (Leftmost multiplication)
        2 * 5 is 10
        ↓
2.  y = 10 * 5 + 3 * 5 + 7;         (Leftmost multiplication)
        10 * 5 is 50
        ↓
3.  y = 50 + 3 * 5 + 7;             (Multiplication before addition)
            3 * 5 is 15
            ↓
4.  y = 50 + 15 + 7;                (Leftmost addition)
        50 + 15 is 65
        ↓
5.  y = 65 + 7;                     (Last addition)
        65 + 7 is 72
        ↓
Step 6. y = 72                      (Last operation—place 72 in y)
```

Precedence of arithmetic operations

- ? = **1 + 2 * (3 + 4)**
  - Evaluated as **1 + (2 * (3+4))** and the result is **15**
- ? = **5 * 2 + 9 % 4**
  - Evaluated as **(5*2) + (9 % 4)** and the result is **11**
- ? = **5 * 2 % (7 – 4)**
  - Evaluated as **(5 * 2) % (7 – 4)** and the result is **1**

Data Type of an Arithmetic Expression

Data type of an expression depends on the type of its operands

- Data type conversion is done by the compiler

If operators are *, /, +, or – , then the type of the result will be:

- integer, if all operands are integer.
  - » Int A , B;
  - » A+ B → Integer.
- float, If at least one operand is float and there is no double
  - » Int A ; Float B;
  - » A + B → Float.
- double, if at least one operand is double
  - Int A ; Float B; Double C;
    - » A + B + C → double.

Data Type of an Arithmetic Expression

Example

int * int; → result int

int + float; → result float

Int + double / float; → result double

int – double; → result double

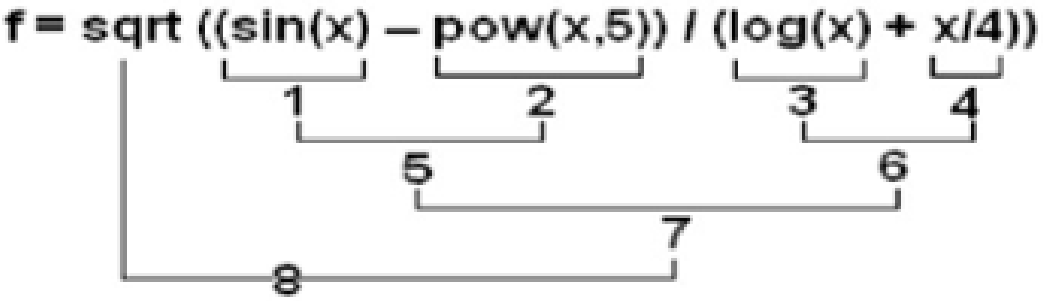
Wrie the following equation as a C++ expression and state the order of evaluation of the binary operators

$$f = \sqrt{\frac{\sin(x) - x^5}{\ln(x) + \frac{x}{4}}}$$

Solution:

f = sqrt ((sin(x) – pow(x,5)) / (log(x) + x/4))

Order of evaluation:



Exercise:

Write the following equation as a C++ expression and state the order of evaluation of the binary operators:

$$z = \sqrt{\frac{x^2 y - 3\sin(x)}{\tan x^3 + x^3 / y}}$$

Increment and Decrement Operators

- Increment operator: increment variable by 1
  - Pre-increment: ++variable
  - Post-increment: variable++
- Decrement operator: decrement variable by 1
  - Pre-decrement: --variable
  - Post-decrement: variable --
- Examples :

  ++K , K++ → k= K+1

  --K , K-- → K= K-1

Increment and Decrement Operators

- If the value produced by ++ or – – is not used in an expression, it does not matter whether it is a pre or a post increment (or decrement).
- When ++ (or – –) is used before the variable name, the computer first increments (or decrements) the value of the variable and then uses its new value to evaluate the expression.
- When ++ (or – –) is used after the variable name, the computer uses the current value of the variable to evaluate the expression, and then it increments (or decrements) the value of the variable.
- **There is a difference between the following**

```
x = 5;
Cout << ++x;
```

```
x = 5;
Cout << x++;
```

Lecturer : Zainab Rustum Mohsin

The difference between the Prefix and Postfix notations:

Prefix notation

```
int y;
int x = 7;
cout<< ++x <<endl;
y=x;
cout<<y;
```

Output:

```
8
8
```

Postfix notation

```
int y;
int x = 7;
cout<< x++ <<endl;
y=x;
cout<<y;
```

Output:

```
7
8
```

special assignment statements

- C++ has special assignment statements called compound assignments

+= , -= , *= , /= , %=

- Example:

```
x +=5 ; means   x = x + 5;
x *=y;  means   x = x * y;
x /=y;  means   x = x / y;
```

Lecturer : Zainab Rustum Mohsin

Q\ Write program to read three different inputs and outputs it ?

```cpp
#include <iostream>

using namespace std;

int main()
{
    int num;
    char ch;
    float f;
    cout << "Enter a number = " ;
    cin >> num ;
    cout << "Enter a character = " ;
    cin >> ch ;
    cout << "Enter a float number = " ;
    cin >> f ;
    cout << "You enter the number = " << num << endl;
    cout << "You enter a character = " << ch << endl;
    cout << "You enter a float number = " << f << endl ;

    return 0;
}
```

```
Enter a number = 58312
Enter a character = e
Enter a float number = 45.89
You enter the number = 58312
You enter a character = e
You enter a float number = 45.89

Process returned 0 (0x0)   execution time : 14.564 s
Press any key to continue.
```

Lecturer : Zainab Rustum Mohsin

Q\ Write program to read two numbers and compute the arithmetic operations ?

```cpp
#include <iostream>

using namespace std;

int main()
{
    int n1,n2, sum , sub , div, mut;
    cout << "Enter any two numbers = " ;
    cin >> n1 >> n2 ;
    sum = n1+n2;
    sub = n1-n2;
    div = n1/n2;
    mut = n1*n2;
    cout << "The summation of two numbers = " << sum <<endl ;
    cout << "The subtraction of two numbers = " << sub <<endl ;
    cout << "The division of two numbers = " << div <<endl ;
    cout << "The multiplication of two numbers = " << mut <<endl ;

    return 0;
}
```

```
Enter any two numbers = 9 3
The summation of two numbers = 12
The subtraction of two numbers = 6
The division of two numbers = 3
The multiplication of two numbers = 27
```

Is there any errors !!!!! explore

The following program to compute different division operators

```
#include <iostream>

using namespace std;

int main()
{
    int x,y,z,r,s;
    x = 17/2;
    y = 17/(-2);
    z = -17/2;
    r = -17/(-2);
    s = 17%2 ;
    cout << "x = " << x <<endl ;
    cout << "y = " << y <<endl ;
    cout << "z = " << z <<endl ;
    cout << "r = " << r <<endl ;
    cout << "s = " << s <<endl ;

    return 0;
}
```

```
x = 8
y = -8
z = -8
r = 8
s = 1
```

Lecturer : Zainab Rustum Mohsin

Q/ Write program to read three marks and compute the average of them?

```
#include <iostream>

using namespace std;

int main()
{
    int x1,x2,x3;
    double av;
    cout << "Enter the three marks of student ,  " ;
    cin >> x1 >> x2 >> x3 ;
    av=(x1+x2+x3)/3 ;
    cout << "The average is = " << av <<endl ;

    return 0;
}
```

```
Enter the three marks of student ,  67 89 96
The average is = 84
```

Is there any errors !!!!!! explore

جامعة ذي قار
كلية التربية للعلوم الصرفة
قسم علوم الحاسبات
المرحلة الاولى
مادة البرمجة المهيكلة

Lecture 6

Instructor: Zainab Rustum Mohsin

Lecturer : Zainab Rustum Mohsin

Control Statements

- Normally, statements in a program are executed one after the other in the order in which they're written.
- This is called sequential execution.

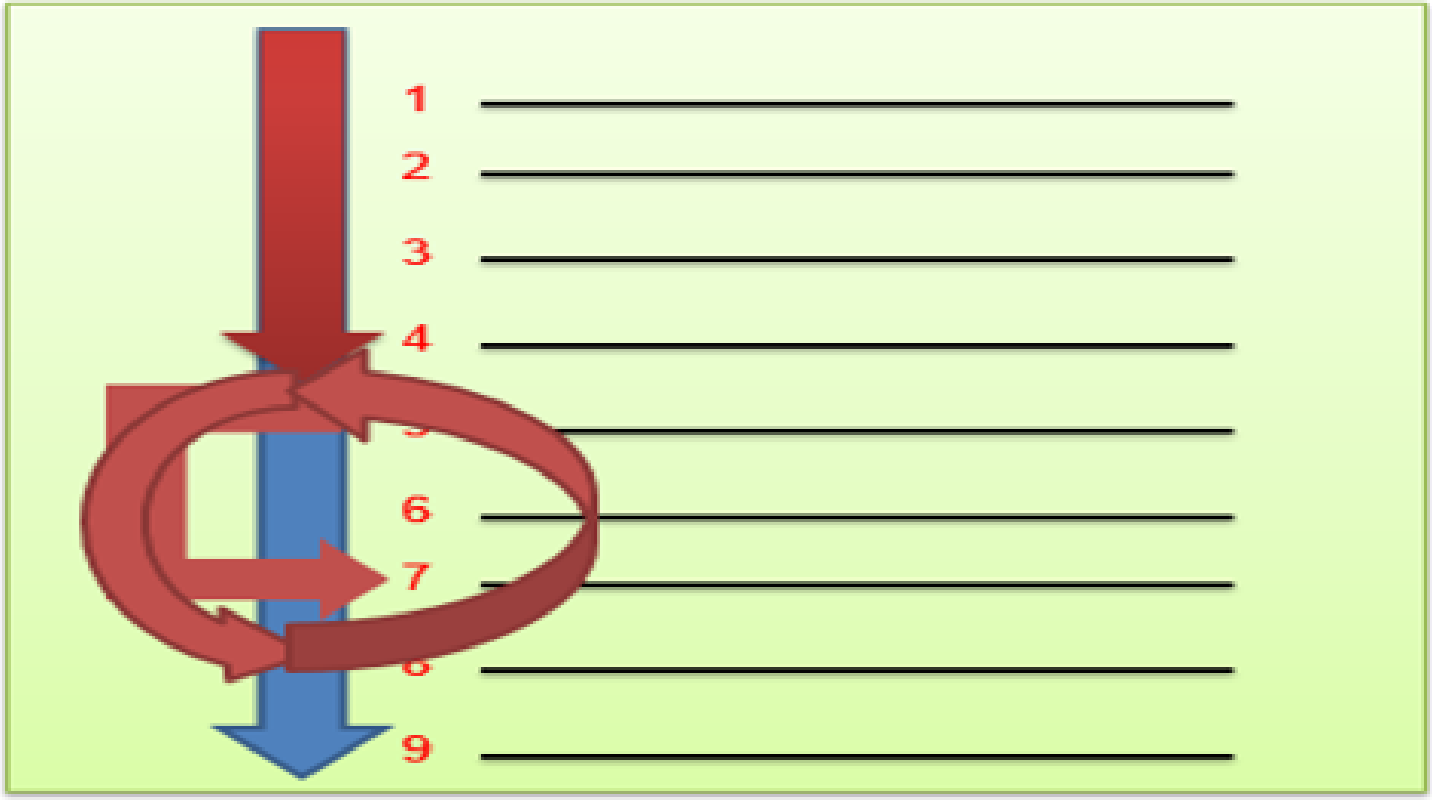
- There are control statements enable you to specify that the next statement to be executed may be other than the next one in sequence.
- This is called transfer of control.

- The control statements are categorized in almost two groups:

Selection control statements

Repetition control statements

Transfer of Control

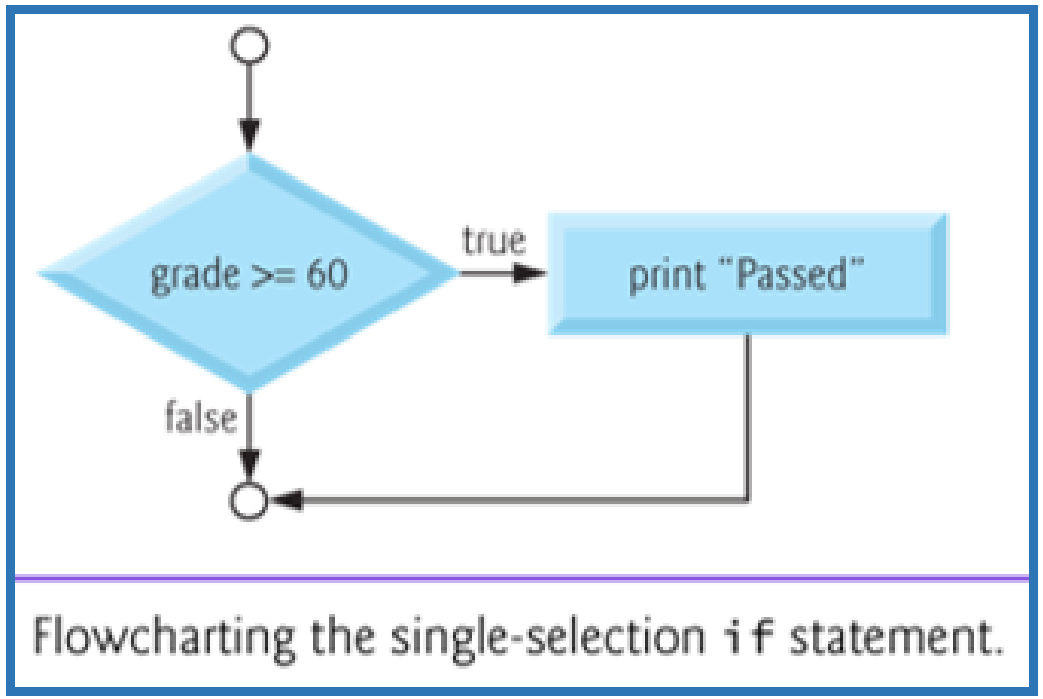


Selection Statements : If Statement

- Selection statements are used to choose among alternative courses of action.

For example, suppose the passing mark on an exam is 60. The pseudocode statement

If student's marks is greater than or equal to 60 Then
Print "Passed"



Flowcharting the single-selection if statement.

Lecturer : Zainab Rustum Mohsin

Lecturer : Zainab Rustum Mohsin

In C++ , The syntax for the If statement

```
if ( Expression)
action statement ;
```

```
if ( Expression)
        {
action statement 1 ;
action statement 2 ;
                    .
                    .
action statement n ;
                   }
```

```
if ( grade >= 60 )
cout <<"Passed\n";
```

- The **Expression** can be any valid expression including a relational expression and even arithmetic expression

- In case of using arithmetic expressions , a non-zero value is considered to be true, whereas a 0 is considered to be false

Lecturer : Zainab Rustum Mohsin

Relational Expression and Relational Operators

Relational expression is an expression which compares 2 operands and returns a TRUE or FALSE answer.

Example : a >= b , a == c , a >= 99 , 'A' > 'a'

Relational expressions are used to test the conditions in selection, and looping statements.

Operator	Means
==	Equal To
!=	Not Equal To
<	Less Than
<=	Less Than or Equal To
>	Greater Than
>=	Greater Than or Equal To

Example 1:

```
if ( avrg >= 3.5 )
        cout << "good";
```

Example 2:

```
if ( x > 0.0 )
        sum += x;
```

Example 3:

```
cin >> num;
if ( num == 0 )
        zcount = zcount + 1;
```

Example : write a program that accept an integer from the user and in case this integer is even print out the following message

"This number is even " .

```
#include <iostream>

using namespace std;

int main()
{
    int a;
    cout << "Enter integer number = " ;
    cin >> a;
    if (a%2== 0)
        cout << "this number is even \n";
    return 0;
}
```

```
C:\Users\2020\Desktop\c++p
Enter integer number = 6
this number is even

Process returned 0 (0x0)   execution time : 3.816 s
Press any key to continue.
```

Example : write a program to read anumber and check if it is positive , if it is so print it , add it to sum, and decrement it by 6.

```cpp
#include <iostream>

using namespace std;

int main()
{
    int num , sum=5 , sub;
    cout << "Enter integer number = " ;
    cin >> num;
    if (num > 0)
        {cout << "this number is positive \n";
          sum +=num ; //sum= sum+num
          sub = num-6;
          cout<< "the summation = "<< sum << endl;
          cout<< "the subtraction = " << sub << endl;
        }
    return 0;
}
```

```
C:\Users\2020\Desktop\c++p
Enter integer number = 9
this number is positive
the summation = 14
the subtraction = 3

Process returned 0 (0x0)   execution time : 3.579 s
Press any key to continue.
```

```cpp
#include <iostream>

using namespace std;

int main()
{
    int num , sum=5 , sub;
    cout << "Enter integer number = " ;
    cin >> num;
    if (num > 0)
        cout << "this number is positive \n";
          sum +=num ; //sum= sum+num
          sub = num-6;
          cout<< "the summation = "<< sum << endl;
          cout<< "the subtraction = " << sub << endl;
    return 0;
}
```

```
C:\Users\2020\Desktop\c++p
Enter integer number = -8
the summation = -3
the subtraction = -14

Process returned 0 (0x0)   execution time : 7.949 s
Press any key to continue.
```

Selection Statements : If .. Else Statement

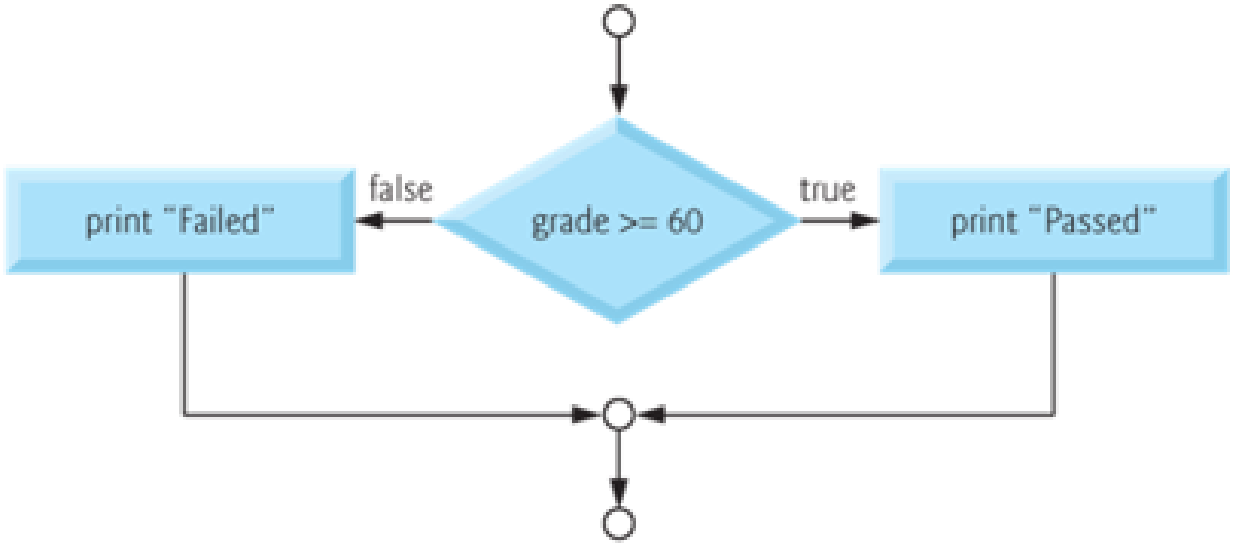
The IF…Else selection statement allows you to specify that there is a course of actions are to be performed when the condition is true and another course of actions will be executed when the condition is false.

- For example, the pseudocode statement
  - **If** student's mark is greater than or equal to 60

    Print "Passed"

    else

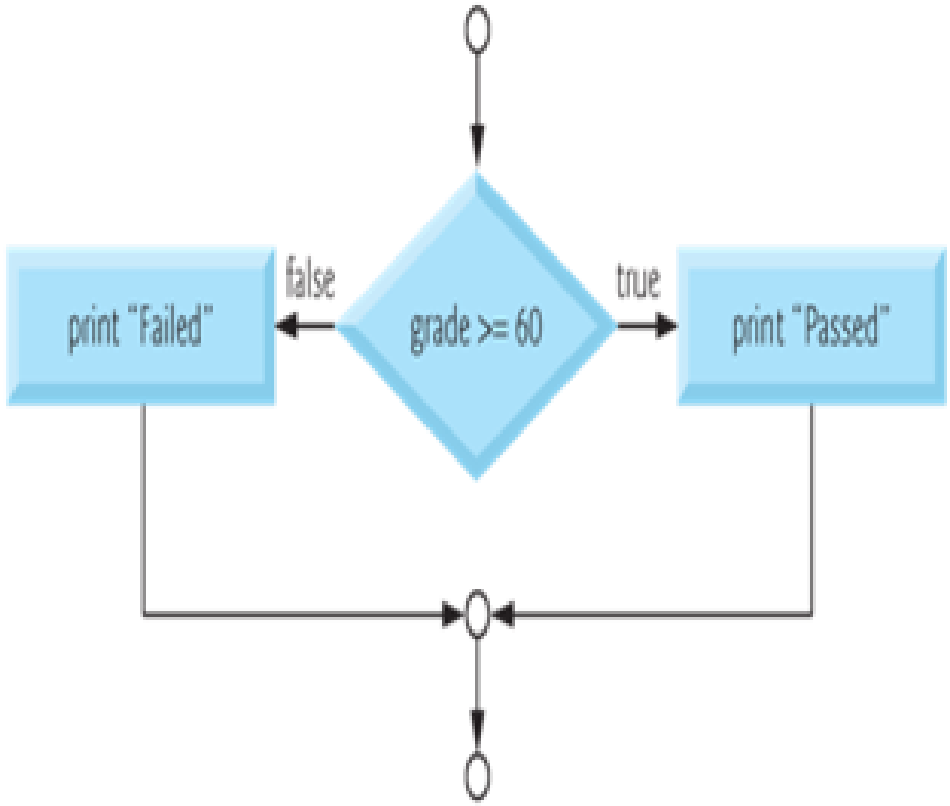
    Print "Failed"



In C++ , The syntax for the If...Else statement

```
if ( Expression)
   action statement ;
Else
   action statement ;
```

```
if ( Expression)
 {
   action statements 1 ;
   .
   action statement n ;
}
Else
{
action statements 1 ;
   .
action statement n ;
}
```

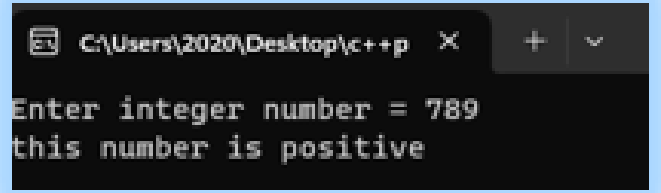


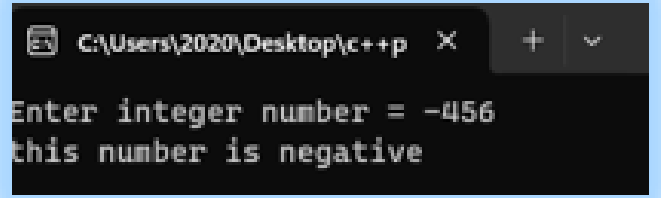

```
if ( grade >= 60 )
   cout <<"Passed\n";
Else
   cout <<"Failed\n"
```

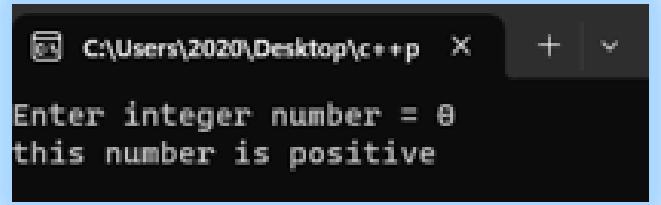
Lecturer : Zainab Rustum Mohsin

Example : write a program that accept an integer from the user and print out whether it is Positive or Negative number.

```
#include <iostream>

using namespace std;

int main()
{
    int num ;
    cout << "Enter integer number = " ;
    cin >> num;
    if (num >= 0)
        cout << "this number is positive \n";
    else
        cout<< "this number is negative \n ";

    return 0;
}
```





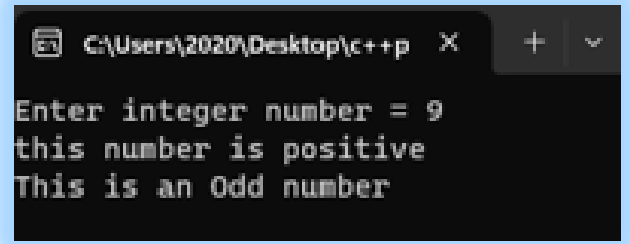


Nested If (IF – Else IF statement)

- Nested If : means to write an if statement within another if statement.

Example : write a program that accept an integer number from the user , in case the number is Positive , check and print out whether it is Even or Odd number.

```
#include <iostream>

using namespace std;

int main()
{
    int num ;
    cout << "Enter integer number = " ;
    cin >> num;
    if (num >= 0)
        { cout << "this number is positive \n";
          if (num % 2 == 0)
            cout <<" This is an Even number \n";
          else
           cout <<"This is an Odd number  \n";
        }
    else
        cout<< "this number is negative \n ";

    return 0;
```



Example write a program that ask the user to Enter 2 numbers and print out whether they are equal or there is one which is greater than the other.

```
#include <iostream>

using namespace std;

int main()
{
    int num1, num2;
    cout <<"Enter Number 1 , Number2  \n";
    cin >>num1>>num2;
    if ( num1 == num2 )
       cout << "Both Are Equal \n";
    else if (num1 > num2 )
            cout <<"Number 1 is greater than number 2 \n";
         else
            cout <<"Number 2 is greater than number 1 \n";

            return(0);
}
```

```
C:\Users\2020\Desktop\c++p
Enter Number 1 , Number2
45 89
Number 2 is greater than number 1
```

Example the following code will print:

A for exam grades greater than or equal to 90
B for grades greater than or equal to 80
C for grades greater than or equal to 70
D for grades greater than or equal to 60
F for all other grades.

```
C:\Users\2020\Desktop\c++p
Enter the grade =
78
C
```

```cpp
using namespace std;

int main()
{
    int grade;
    cout <<"Enter the grade =  \n";
    cin >>grade;
  if ( grade >= 90 )
    cout << "A\n";
  else if ( grade >= 80 )
    cout << "B\n";
  else if ( grade >= 70 )
    cout << "C\n";
  else if ( grade >= 60 )
    cout << "D\n";
  else
    cout << "F\n" ;
            return(0);
}
```

Lecturer : Zainab Rustum Mohsin

Combining more than one condition

To combine more than one condition we use the logical operators.

Operator	Means	Description
&&	And	The Expression Value Is true If and Only IF both Conditions are true
||	OR	The Expression Value Is true If one Condition Is True

Example : check whether num1 is between 0 and 100

```
IF ( (num1 >= 0) && (num1 <=100) )
    Cout <<"The Number Is between 1 and 100" ;
 Else
     Cout <<" The Number Is not in range";
```

Example print out the student grade according to the following formulas:

A for exam marks greater than or equal 90 and less than or equal 100 ,
B for exam marks greater than or equal 80 and less than 90 ,
C for exam marks than or equal to 70 and less than 80 ,
D for exam marks than or equal to 60, and less than 70 ,
F for all other marks.

```
C:\Users\2020\Desktop\c++p
Enter the grade =
67
D
```

```cpp
using namespace std;

int main()
{

   int marks;
   cout <<"Enter the grade =  \n";
   cin >>marks;
  if ( marks >= 90&& marks <= 100)
        cout << "A\n" ;
  else if (marks >= 80 && marks <90 )
         cout << "B\n";
       else if (marks >= 70 && marks <80 )
              cout << "C\n";
            else if (marks >= 60 && marks <70 )
    cout << "D\n";
  else
    cout << "F\n" ;

            return(0);
```

Lecturer : Zainab Rustum Mohsin

ASSIGMENTS

Write a C++ program to read any two numbers and print the largest value of it:

Write a C++ program to read a student degree, and check if it's degree greater than or equal to 50, then print pass, otherwise print fail:

Write C++ program to compute the value of Z according to the following equations:

$$Z = \begin{cases} x + 5 & : x < 0 \\ \cos(x) + 4 & : x = 0 \\ \sqrt{x} & : x > 0 \end{cases}$$

Write C++ program to find a largest value among three numbers:

Write a C++ program to read a number, and print the day of the week:

جامعة ذي قار
كلية التربية للعلوم الصرفة
قسم علوم الحاسبات
المرحلة الاولى
مادة البرمجة المهيكلة

Lecture 7

Instructor: Zainab Rustum Mohsin

Lecturer : Zainab Rustum Mohsin

Selection statements: Switch Statement.

switch control statement that allows us to make a decision from the number of choices

```
Switch (Expression )
{
case constant 1 :
  Action statements;
  break ;
case constant 2 :
  Action statements;
  break ;
default :
 Action statements;
}
```

Expression It could be an integer constant like 1, 2 or 3, or an expression that evaluates to an integer .

Constant : is a specific value.

2

Lecturer : Zainab Rustum Mohsin

```
int i;
Cin>>I;
switch ( i )
{
case 10 :
  Cout << "I am in case 1 \n“ ;
  break ;
case 20 :
  Cout << "I am in case 2 \n“ ;
  break ;
case 30 :
  Cout << "I am in case 3 \n“ ;
  break ;
default :
    Cout << "I am in default \n“ ;
}
```

```
char ch = 'x' ;
switch ( ch )
{
  case 'v' :
     Cout<< "I am in case v \n“  ;
     break ;
case 'a' :
   Cout<<  "I am in case a \n“  ;
   break ;
case 'x' :
   Cout<< "I am in case x \n” ;
   break ;
default :
     Cout<< "I am in default \n“  ;
}
```


3

Lecturer : Zainab Rustum Mohsin

Example :- Find Grade of Student using Switch Case and Function in C++

```
#include <iostream>

using namespace std;

int main()
{
  int score;
   char grade;
   cout << "Enter score(0-100): ";
   cin >> score;
   if(score<0 || score>100)
        cout << "Invalid Score";
   else {     // find grade for given score
```

```
switch( score / 10 )
        {
          case 10:
          case 9:
            cout<<"A";break;
case 8:
            cout<<"B";break;
          case 7:
            cout<<"C";break;
          case 6:
            cout<<"D";break;
          case 5:
           cout<<"E";break;
     default:
       cout<<"F";
   }}
   return 0;
}
```

Exercise:

Insert the missing parts to complete the following `switch` statement.

```
int day = 2;
switch (     ) {
      1:
    cout << "Saturday";
    break;
      2:
    cout << "Sunday";
      ;
}
```

Switch Versus IF – Else IF

Lecturer : Zainab Rustum Mohsin

There are some things that you simply cannot do with a switch::

1- A float expression cannot be tested using a switch .

2- Cases can never have variable expressions (for example it is wrong to say case a +3 :)

3- Multiple cases cannot use same expressions.

4- switch works faster than an equivalent IF - Else ladder.

6

جامعة ذي قار
كلية التربية للعلوم الصرفة
قسم علوم الحاسبات
المرحلة الاولى
مادة البرمجة المهيكلة

Lecture 8

Instructor: Zainab Rustum Mohsin

1. For Statement:

General Form of For statement:

```
for ( initialization ; continuation condition ; update )
    statement1 ;
```

```
for ( initialization ; continuation condition ; update )
{
    statement1 ;
    statement2 ;
        :
}
```

Example 1:

```
for ( i = 0;  i < 10;  i++ )
        cout << i;
```

Output:

0 1 2 3 4 5 6 7 8 9

Example 2:

```
for ( i = 0;  i < 10;  i += 2 )
        cout << i;
```

Output: *even numbers only*

0 2 4 6 8

Example 3:

```
for ( i = 1;  i < 10;  i += 2 )
        cout << i;
```

Output: *odd numbers only*

1 3 5 7 9

Example 1

Write C++ program to add the numbers between 1 and 100:

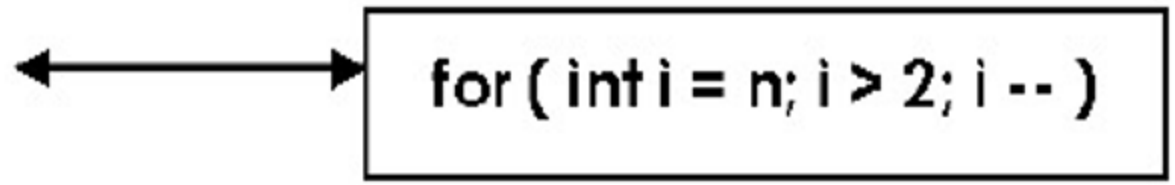
```
#include<iostream.h>
void main( )
{
        int sum = 0;
        for ( int i = 1; i <= 100; i ++ )
                sum = sum + i;
        cout << "sum is: " << sum;
}
```

Example 2

Write C++ program to find the factorial of n *(using for statement)*:

$$n! = n * n-1 * n-2 * n-3 * \dots * 2 * 1$$

```
#include<iostream.h>
void main( )
{
        int n, f = 1;
         cout << "enter positive number: ";
         cin >> n;
         for ( int i = 2; i <= n; i ++ )
             f = f * i;
         cout << "factorial is: " << f;
}
```



Example 3

Write C++ program to the result of the following:

$$\sum_{i=1}^{20} a_i^2$$

```
#include<iostream.h>
void main( )
{
        int sum = 0;
        for ( int i = 1; i <= 20; i ++ )
            sum = sum + ( i * i );
        cout << "The sum is: " << sum;
}
```

Example 4

Write C++ program to read 10 integer numbers, and find the sum of positive number only:

```
#include<iostream.h>
void main( )
{
        int num, sum = 0;
        for ( int i = 1; i <= 10; i ++ )
        {
            cout << "enter your number: ";
            cin >> num;
            if ( num > 0 )     sum = sum + num;
        }
         cout << "The sum is: " << sum;
}
```

Example 5

Write C++ program to print the following series: 1, 2, 4, 8, 16, 32, 64

```
#include<iostream.h>
void main( )
{
        int x;
        for ( x = 1; x < 65; x *= 2 )
            cout << x <<"   ";
}
```

2. More about For Statement:

☑ We can use more than one control with for statement, as follow:

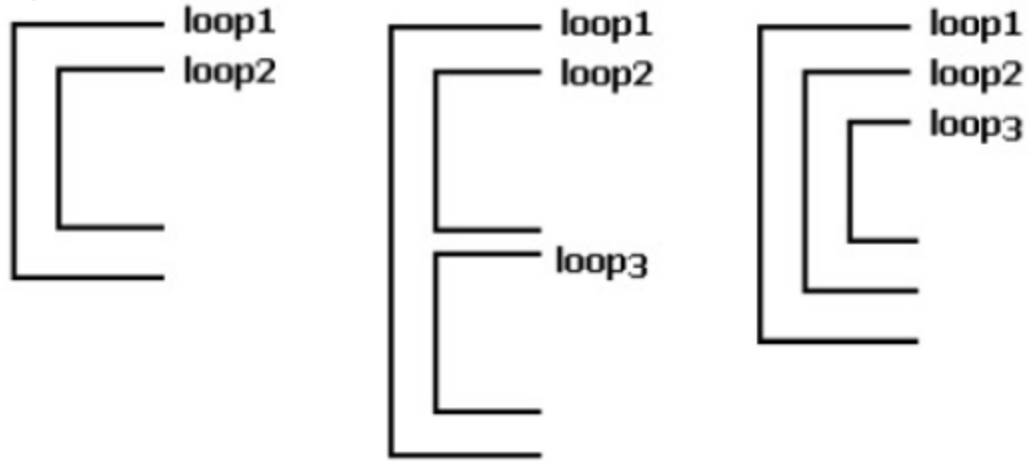
```
for ( int m = 1, int n = 8 ; m < n ; m ++ , n -- )
```

☑ We can create infinite loop, as follow:

```
for ( ; ; )
```

3. Nested Loops:

We can put loops one inside another to solve a certain programming problems. Loops may be nested as follows:



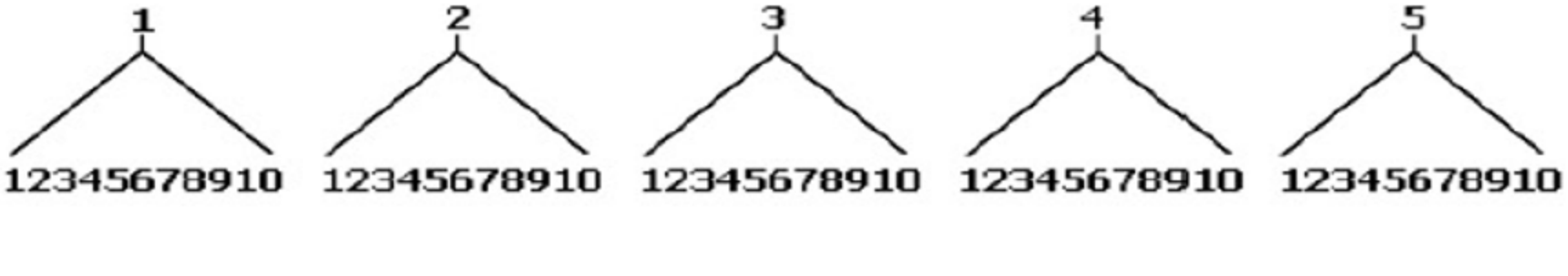
Example 8

Write C++ program to evaluate the following series:

$$\sum_{i=1}^{5} \sum_{j=1}^{10} i + 2j$$

$\sum_{i=1}^{5}$

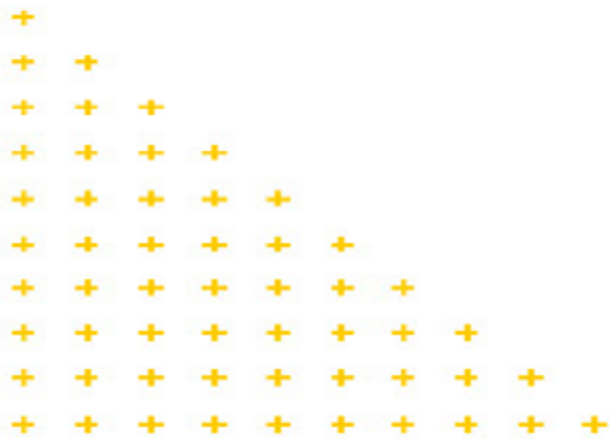
$\sum_{j=1}^{10}$




```
#include<iostream.h>
void main( )
{
        int i, j, sum = 0;
        for ( i = 1; i <= 5; i ++ )
            for ( j = 1; j <= 10; j ++ )
                sum = sum + ( i + 2 * j );
        cout << "sum is:" << sum;
}
```

Example 9

Write C++ program to print the following figure:



```
#include<iostream.h>
void main( )
{
        int i, j;
        for ( i = 1; i <= 10; i ++ )
        {
            for ( j = 1; j <= i; j ++ )
                cout << " + ";
            cout << "\n";
         }
}
```

Exercise:

What is the output of the following C++ program ?

```
#include<iostream.h>
void main( )
{
        int i, j, k;
        for ( i = 1; i <= 2; i ++ )
        {
            for ( j = 1; j <= 3; j ++ )
                {
                    for ( k = 1; k <= 4; k ++ )
                            cout << " + ";
                    cout << "\n";
                }
             cout << "\n";
         }
}
```

1. Breaking Control Statements:

For effective handling of the loop statements, C== allows the use of the following types of control break statements:

(a) Break Control Statement:

The break statement is used to terminate the control from the loop statements of the case-switch structure. The break statement is normally used in the switch-case loop and in each case condition; the break statement must be used. If not, the control will be transferred to the subsequent case condition also. The general format of the break statement is : (**Break**;)

Example 1:

```
for ( i = 1;  i < 100;  i ++ )
{
     cout << i;
     if ( i == 10 )   break;
}
```

Output:

1 2 3 4 5 6 7 8 9 10

Example 2:

```
for ( i = 1;  i < 10;  ++ i )
   for ( j = 1;  j < 20;  ++ j )
      {
            cout << i * j << endl;
            if ( j == 10 )   break;
       }
```

Example 3:

```
Switch (day) {
Case '1':cout<<"Monday\n";
            Break;
Case '2': …..
}
```

(b) Continue Control Statements:

The continue is used to repeat the same operations once again even it it checks the error. Its general syntax is: **(continue;)**

It is used for the inverse operation of the break statement. The following program segment will process only the positive integers. Whenever a zero or negative value is encountered, the computer will display the message "zero or negative value found" as an error and it continues the same loop as long as the given condition is satisfied.

```
Cin>>value;
While (value <=100)  {
If (value <=0)
Cout<<"zero or negative value found\n";
Continue;
}  }
```

(c) Goto Statement:

The goto statement is used to alter the program execution sequence by transferring the control to some other part of the program. Its general syntax is: **(goto label;)**

There are two ways of using this statement:

1. **Unconditional Goto:** Itis used just to transfer the control from one part of the programto the other part without checking any condition. It is difficult in use.

Example 2

Write C++ program to check if zero or negative value found:

```
#include<iostream.h>
void main( )
{
        Start: cout<<"***\n";
        Goto start;
}
```

2. **Conditional Goto:** Itis used to transfer the control of the execution from one part of the program to the other in certian conditional cases.

Example 2

Write C++ program to check if zero or negative value found:

```
#include<iostream.h>
void main( )
{
        Int value,i=0;
        While i<=10) {
        Cout<<"enter a number \n";
        Cin>>value;
        Cout<<"zero or negative value found \n";
        Goto  error;
}
Error:
      Cout<<"input data error \n";
}
```

Using For Statement	Using While Statement	Using Do/While Statement

Q1: Find the summation of the numbers between 1 and 100.

```
for( i=1 ; i<=100 ; i++ )
    s = s + i;
```

```
i = 1;
while ( i <= 100)
{
    s = s + i;
    i++;
}
```

```
i = 1;
do
{
    s = s + i;
    i++;
}
while ( i <= 100 );
```

Q2: Find the factorial of n.

```
cin >> n;
for( i=2 ; i<=n ; i++ )
   f = f * i;
```

```
cin >> n;
i = 2;
while ( i <= n)
{
    f = f * i;
    i++;
```

```
cin >> n;
i = 2;
do
{
    f = f * i;
    i++;
```

Q3: To find the result of the following: $\sum_{i=1}^{20} a_i^2$.

```
for( i=1 ; i<=20 ; i++ )
    s = s + (i *i);
```

```
i = 1;
while ( i <= 20)
{
    s = s + (i *i);
    i++;
}
```

```
i = 1;
do
{
    s = s + (i *i);
    i++;
}
while ( i <= 20);
```

Q4: Read 10 numbers, and find the sum of the positive numbers only.

```
for( i=1 ; i<=10 ; i++ )
{
    cin >> x;
    if ( x>0 )  s = s + x;
}
```

```
i = 1;
while ( i <= 10)
{
    cin >> x;
    if ( x>0 )  s = s + x;
    i++;
}
```

```
i = 1;
do
{
    cin >> x;
    if ( x>0 )  s = s + x;
    i++;
}
while ( i <= 10);
```

Q5: Represent the following series: 1, 2, 4, 8, 16, 32, 64.

```
for( i=1 ; i<65 ; i*=2 )
   cout << i;
```

```
i = 1;
while ( i<65)
{
    cout << i;
    i*=2;
}
```

```
i = 1;
do
{
    cout << i;
    i*=2;
}
while ( i<65);
```

Q6: Find the sum of the following s = 1 + 3 + 5 + 7 + … + 99.

```
for( i=1 ; i<=99 ; i+=2 )
    s = s + i;
```

```
i = 1;
while ( i<=99)
{
    s = s + i;
    i+=2;
}
```

```
i = 1;
do
{
    s = s + i;
    i+=2;
}
while ( i<=99);
```

Q7: Find the sum and average of the 8 degrees of the student.

```
for( i=1 ; i<=8 ; i++ )
{
    cin >> d;
    s = s + d;
}
av = s / 8;
```

```
i = 1;
while ( i<=8)
{
    cin >> d;
    s = s + d;
    i++;
}
av = s / 8;
```

```
i = 1;
do
{
    cin >> d;
    s = s + d;
    i++;
}
while ( i<=8);
av = s / 8;
```

Q8: Find the cub of n numbers, while the entered number is a positive.

Can't be solve this problem using For statement

```
cin >> x;
while ( x > 0 )
{
    c = x * x * x;
    cin >> x;
}
```

```
do
{
    cin >> x;
    c = x * x * x;
}
while ( x > 0 );
```

Using While Statement:

Q1: Write C++ program to find the summation of the odd numbers, between 0 and 100.

Q5: What are the output of the following segment of C++ code:

```
int i;
i = 12;
do
{
   cout << i << endl;
   i --;
}
while ( i > 0 );
```

Q17: What is the output of the following C++ segment of code:

```
for ( I = 0; I < 8; I ++ )
{
  if ( I % 2 == 0 )    cout << I + 1 << endl;
  else  if  ( I % 3 == 0 )  continue;
  else  if  ( I % 5 == 0 )  break;
  cout << "end program \n";
}
cout << "end ...";
```

Q19: Write C++ program to print the following searies:

1. $Sum=1+2^2+4^2+...+n^2$
2. $Sum=1-3^x+5^x-...+n^x$
3. $Sum=1+1/1!+2/2!+3/3!+...+n/n!$ where $n!=1*2*3*...*n$

جامعة ذي قار
كلية التربية للعلوم الصرفة
قسم علوم الحاسبات
المرحلة الاولى
مادة البرمجة المهيكلة

Lecture 9

Instructor: Zainab Rustum Mohsin

1. Arrays:

An array is a consecutive group of homogeneous memory locations. Each element (location) can be referred to using the array name along with an integer that denotes the relative position of that element within the array. The data items grouped in an array can be simple types like int or float, or can be user-defined types like structures and objects.

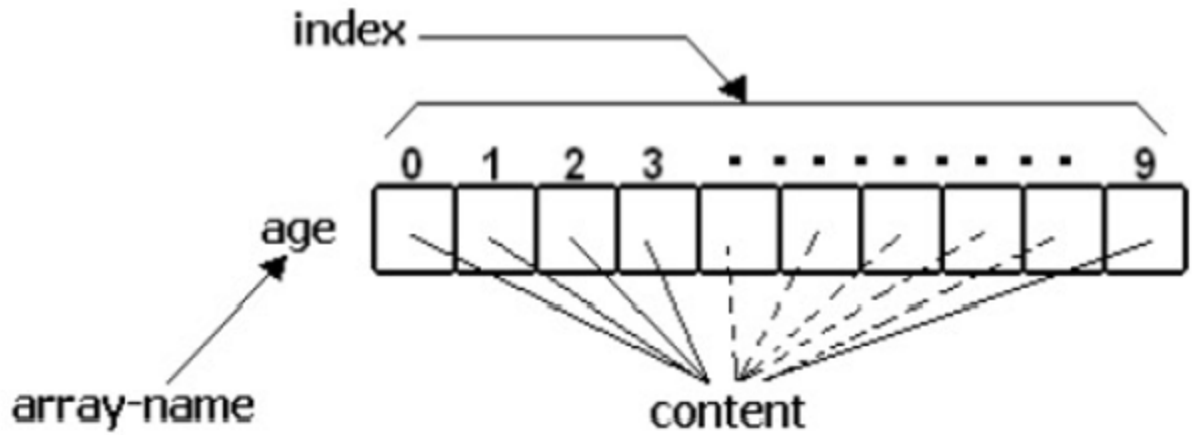
2. Array of One Dimension:

It is a single variable specifies each array element. The declaration of one dimensional arrays is:

General Form of 1D-Array:
data-type Array-name [*size*];

Examples:

```
int     age [10];
int     num [30];
float   degree[5];
char    a [15];
```



The item in an array are called elements (in contrast to the items in a structure which are called members). The elements in an array are of the same type only the values vary.

3. Initializing Array Elements:

- The first element of array age:

```
age [0] = 18;
```

- The last element of array age:

```
age [9] = 19;
```

- All elements of array age:

```
age [9] = { 18, 17, 18 ,18 ,19, 20 ,17, 18 ,19 };
```

- int x [ ] = { 12, 3, 5, 0, 11, 7, 30, 100, 22 };

- int y [10] = { 8, 10, 13, 15, 0, 1, 17, 22};

4. Accessing Array Elements:

We access each array element by written name of array, followed by brackets delimiting a variable (or constant) in the brackets which are called the array index.

- Accessing the first element of array num to variable x:

```
x = num [0];
```

- Accessing the last element of array num to variable y:

```
y = num [9];
```

- `cout << num [0] + num [9];`

- `num [0] = num [1] + num[2];`

- `num [7] = num [7] + 3;` ↔ `num [7] += 3;`

5. Read / Write / Process Array Elements:

- cout << num [4];

- if (num [5] > 5)
  cout << "greater";

- for (int i=0; i<10; i++)
  cin >> num[i];

- for (int i=0; i<10; i++)
  cout << num[i];

- for (int i=9; i>=0; i++)
  cout << num[i];

- sum=0;
  for (int i=0; i<10; i++)
  sum = sum + num[i];

Example 1

 Write C++ program to display 2nd and 5th elements of array distance:

```
#include<iostream.h>

void main( )
{
        double  distance[ ] = { 23.14, 70.52, 104.08, 468.78, 6.28};
         cout << "2nd element is: " << distance[1] << endl;
         cout << "5th element is: " << distance[4];
}
```

Example 2

Write C++ program to read 5 numbers and print it in reverse order:

```
#include<iostream.h>

void main( )
{
        int a [5];
        cout << "Enter 5 numbers \n";
        for ( int i =0; i <5; i++ )
          {
              cout << i << ": ";
              cin >> a [ i ];
              cout << "\n";
          }
        cout << "The reverse order is: \n";
        for ( i =4; i >=0; i-- )
             cout << i << ": " << a [ i ] << endl;
}
```

Example 3

Write C++ program, to find the summation of array elements:

```
#include<iostream.h>

void main ( )
{
        int const L = 10;
        int a [L];
        int sum = 0;
        cout << "enter 10 numbers \n";
        for ( int i =0; i <L; i++ )
            {
                cout << "enter value " << i << ": ";
                cin >> a [ i ];

                  sum += a [ i ];
              }
          cout << "sum is: " << sum << endl;
}
```

Example 4

Write C++ program, to find the minimum value in array of 8 numbers:

```
#include<iostream.h>

void main ( )
{       int n = 8;          int a [  ] = { 18, 25, 36, 44, 12, 60, 75, 89 };
        int min = a [ 0 ];
        for ( int i  = 0; i  < n; i++ )
        if ( a [ i ] < min )      min = a [ i ];
        cout << "The minimum number in array is: " << min;     }
```

Example 7

Write C++ program, to split the odd numbers and even numbers of one array into two arrays:

a = [1, 2, 3, 4, 5, 6, 7, 8, ... , 20]
aodd = [1, 3, 5, 7, ... , 19]
aeven = [2, 4, 6, 8, ... , 20]

```
#include<iostream.h>

void main ( )
{
        int a [ 20 ]= { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20 };
        int aodd[20], aeven [20];
        int i ,o=0, e=0;
        for ( i=0 ; i<20; i++ )
            if (a[i] % 2 !=0)
              {
                  aodd[o]=a[i];
                  o=o+1;
              }
             else
                  {
                      aeven[e]=a[i];
                      e=e+1;
                  }

        for ( i=0 ; i<o; i++ )
            cout<<aodd[i]<<" ";
        cout<<endl;
        for ( i=0 ; i<e; i++ )
            cout<<aeven[i]<<" ";
}
```

جامعة ذي قار
كلية التربية للعلوم الصرفة
قسم علوم الحاسبات
المرحلة الاولى
مادة البرمجة المهيكلة

Lecture 10

Instructor: Zainab Rustum Mohsin

1. Array of Two Dimension:

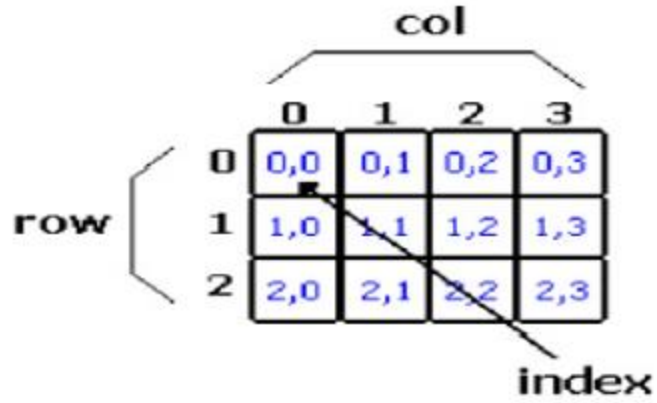
Arrays can have higher dimension. There can be arrays of two dimension which is array of arrays. It is accessed with two index. Also there can be arrays of dimension higher than two.

General Form of 2D-Array:

data-type **Array-name** [*Row-size*] [*Col-size*];

Examples:

```
int     a [10] [10];
int     num [3] [4];
```



2. Initializing 2D-Array Elements:

- The first element of array age:

```
a [2] [3] = { {1, 2, 3} , {4, 5, 6} };
```

1	2	3
4	5	6

3. Read / Write / Process Array Elements

Example 1

Write C++ program, to read 15 numbers, 5 numbers per row, the print them:

```
#include<iostream.h>

void main ( )
{
        int a [ 3 ] [ 5 ];
        int i , j;
        for ( i = 0 ; i < 3; i++ )
           for ( j = 0 ; j < 5; j++ )
              cin >> a [ i ] [ j ];

        for ( i = 0 ; i < 3; i++ )
           {
              for ( j = 0 ; j < 5; j++ )
                 cout << a [ i ] [ j ];
              cout << endl;
           }
}
```

Example 2

Write C++ program, to read 4*4 2D-array, then find the summation of the array elements, finally print these elements:

```
#include<iostream.h>

void main ( )
{
        int a [ 4 ] [ 4 ];
        int i , j, sum = 0;
        for ( i = 0 ; i < 4; i++ )
           for ( j = 0 ; j < 4; j++ )
              cin >> a [ i ] [ j ];

        for ( i = 0 ; i < 4; i++ )
           for ( j = 0 ; j < 4; j++ )
              sum += a [ i ] [ j ];
       cout << "summation is: " << sum << endl;

        for ( i = 0 ; i < 4; i++ )
           {
              for ( j = 0 ; j < 4; j++ )
                 cout << a [ i ] [ j ];
              cout << endl;
           }
}
```

Example 3

Write C++ program, to read 3*4 2D-array, then find the summation of each row:

```
#include<iostream.h>

void main ( )
{
        int a [ 3 ] [ 4 ];
        int i , j, sum = 0;
        for ( i = 0 ; i < 3; i++ )
           for ( j = 0 ; j < 4; j++ )
              cin >> a [ i ] [ j ];

        for ( i = 0 ; i < 3; i++ )
         {
              sum = 0;
              for ( j = 0 ; j < 4; j++ )
                 sum += a [ i ] [ j ];

              cout << "summation of row " << i  << " is: " << sum << endl;
         }
}
```

Example 4

Write C++ program, to read 3*4 2D-array, then replace each value equal 5 with 0:

```
#include<iostream.h>
void main ( )
{
        int a [ 3 ] [ 4 ];
        int i , j;
        for ( i = 0 ; i < 3; i++ )
           for ( j = 0 ; j < 4; j++ )
              cin >> a [ i ] [ j ];

        for ( i = 0 ; i < 3; i++ )
           for ( j = 0 ; j < 4; j++ )
                 if ( a [ i ] [ j ] == 5 )  a [ i ] [ j ] = 0;

        for ( i = 0 ; i < 3; i++ )
           {
              for ( j = 0 ; j < 4; j++ )
                 cout << a [ i ] [ j ];
              cout << endl;
           }
}
```

Example 5

Write C++ program, to addition two 3*4 arrays:

```
#include<iostream.h>
void main ( )
{
        int a [ 3 ] [ 4 ], b [ 3 ] [ 4 ], c [ 3 ] [ 4 ];
        int i , j;
        cout << "enter element of array A: \n";
        for ( i = 0 ; i < 3; i++ )
            for ( j = 0 ; j < 4; j++ )
                cin >> a [ i ] [ j ];
         cout << "enter element of array B: \n";
        for ( i = 0 ; i < 3; i++ )
            for ( j = 0 ; j < 4; j++ )
                cin >> b [ i ] [ j ];
        for ( i = 0 ; i < 3; i++ )
            for ( j = 0 ; j < 4; j++ )
                    c [ i ] [ j ] = a [ i ] [ j ] + b [ i ] [ j ];
        for ( i = 0 ; i < 3; i++ )
            {
                for ( j = 0 ; j < 4; j++ )
                    cout << c [ i ] [ j ];
                cout << endl;
             }
}
```

Example 6

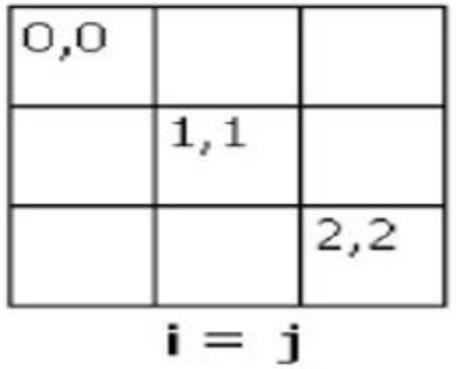
Write C++ program, to convert 2D-array into 1D-array:

```
#include<iostream.h>
void main ( )
{
        int a [ 3 ] [ 4 ];
        int b [ 12 ];
        int i , j, k = 0;

       for ( i = 0 ; i < 3; i++ )
           for ( j = 0 ; j < 4; j++ )
               cin >> a [ i ] [ j ];

        for ( i = 0 ; i < 3; i++ )
           for ( j = 0 ; j < 4; j++ )
               {
                    b [ k ] =  a [ i ] [ j ];
```

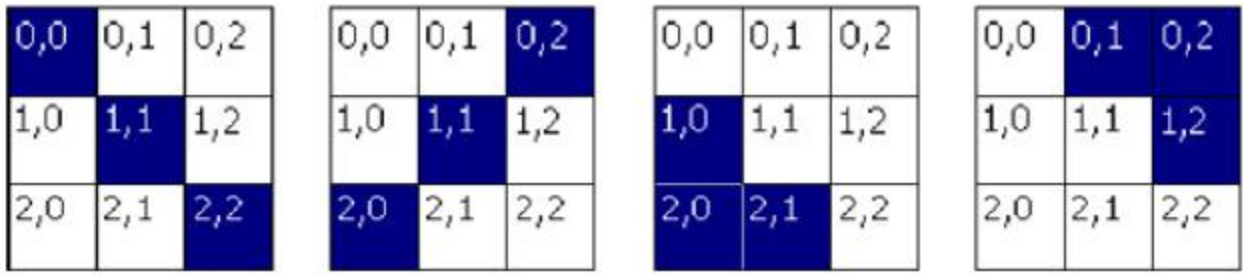
Example 7

Write C++ program, to replace each element in the main diameter (diagonal) with zero:

```
#include<iostream.h>
void main ( )
{
        int a [ 3 ] [ 3 ];
        int i , j;

       for ( i = 0 ; i < 3; i++ )
          for ( j = 0 ; j < 3; j++ )
             cin >> a [ i ] [ j ];

       for ( i = 0 ; i < 3; i++ )
          for ( j = 0 ; j < 3; j++ )
                if ( i == j )    a [ i ] [ j ] = 0;

       for ( i = 0 ; i < 3; i++ )
          {
             for ( j = 0 ; j < 3; j++ )
                cout << a [ i ] [ j ];
             cout << endl;
          }
}
```



i = j



Example 8

Write C++ program, print the square root of an array:

```
#include<iostream.h>

void main ( )
{
        int a [ 3 ] [ 3 ] , b [ 3 ] [ 3 ];
        int i , j;
        for ( i = 0 ; i < 3; i++ )  {
            for ( j = 0 ; j < 3; j++ )    {
                b[ i ][ j ]= sqrt(a[ i ][ j ]);
                cout << b [ i ] [ j ];

} } }
```

Example 9

Write C++ program, to read 3*3 2D-array, then find the summation of the main diagonal and its secondary diagonal of the array elements, finally print these elements:

```
#include<iostream.h>
void main ( )
{
        int a [ 3 ] [ 3 ];
        int i , j, x , y;
        for ( i = 0 ; i < 3; i++ )  {
           for ( j = 0 ; j < 3; j++ )   {
              cin >> a [ i ] [ j ];

              if ( i == j )
              x=x+a[ i ][ j ];
              if ( i + j =4)
              y=y+a[ i ][ j ];
                                    }   }
        cout << "summation of diagonal is: " << x << endl;
        cout << "summation of inverse diagonal is: " << y << endl;
}
```

Assignment

Write C++ program, to read 3*4 2D-array, then find the summation of each col.

Write C++ program, to replace each element in the second diameter (diagonal) with zero.

Write C++ program, to find the summation of odd numbers in 2D-array.

Write C++ program, to find (search) X value in 2D-array, and return the index of it's location.

جامعة ذي قار
كلية التربية للعلوم الصرفة
قسم علوم الحاسبات
المرحلة الاولى
مادة البرمجة المهيكلة

Lecture 11

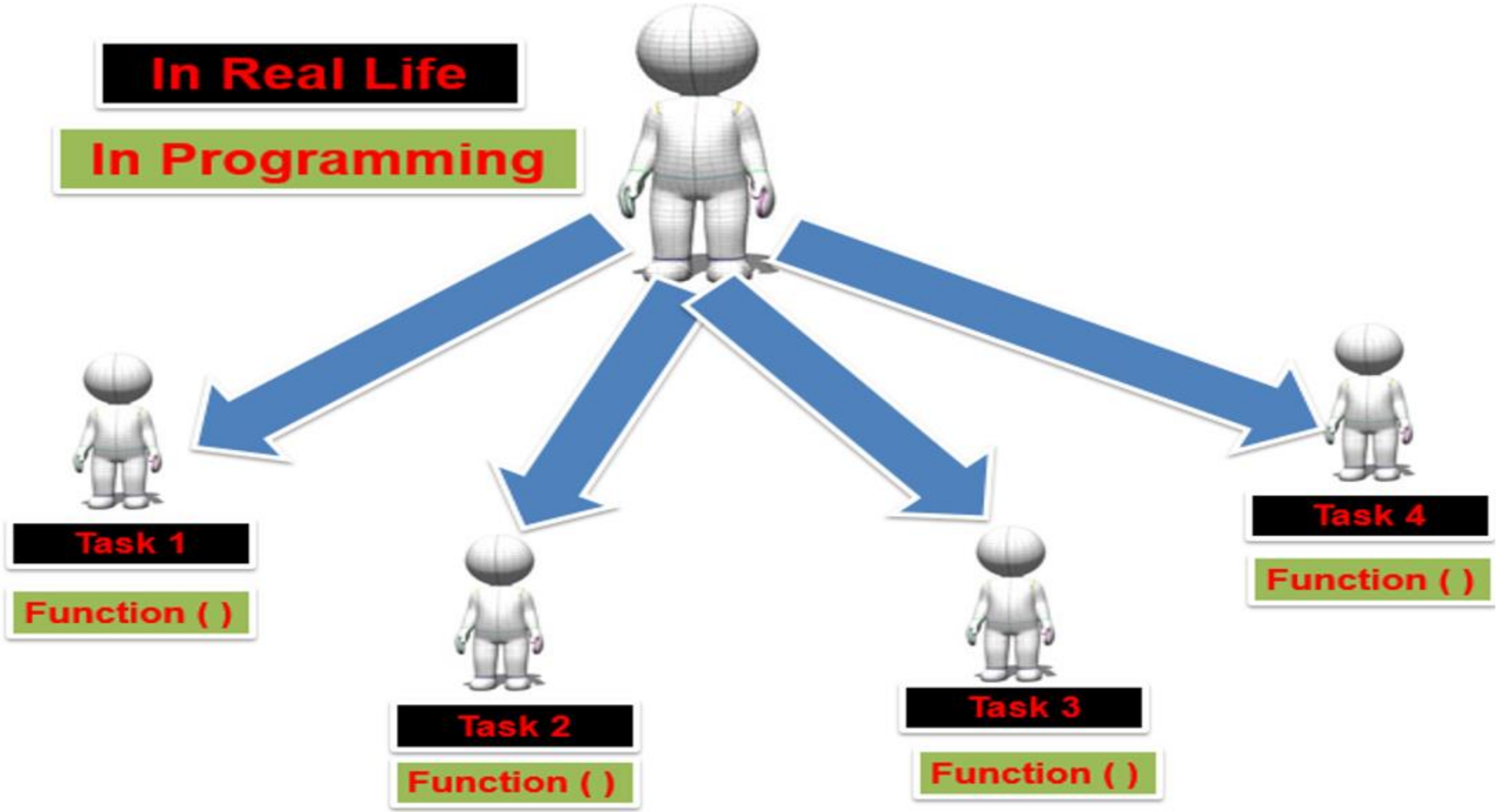
Instructor: Zainab Rustum Mohsin

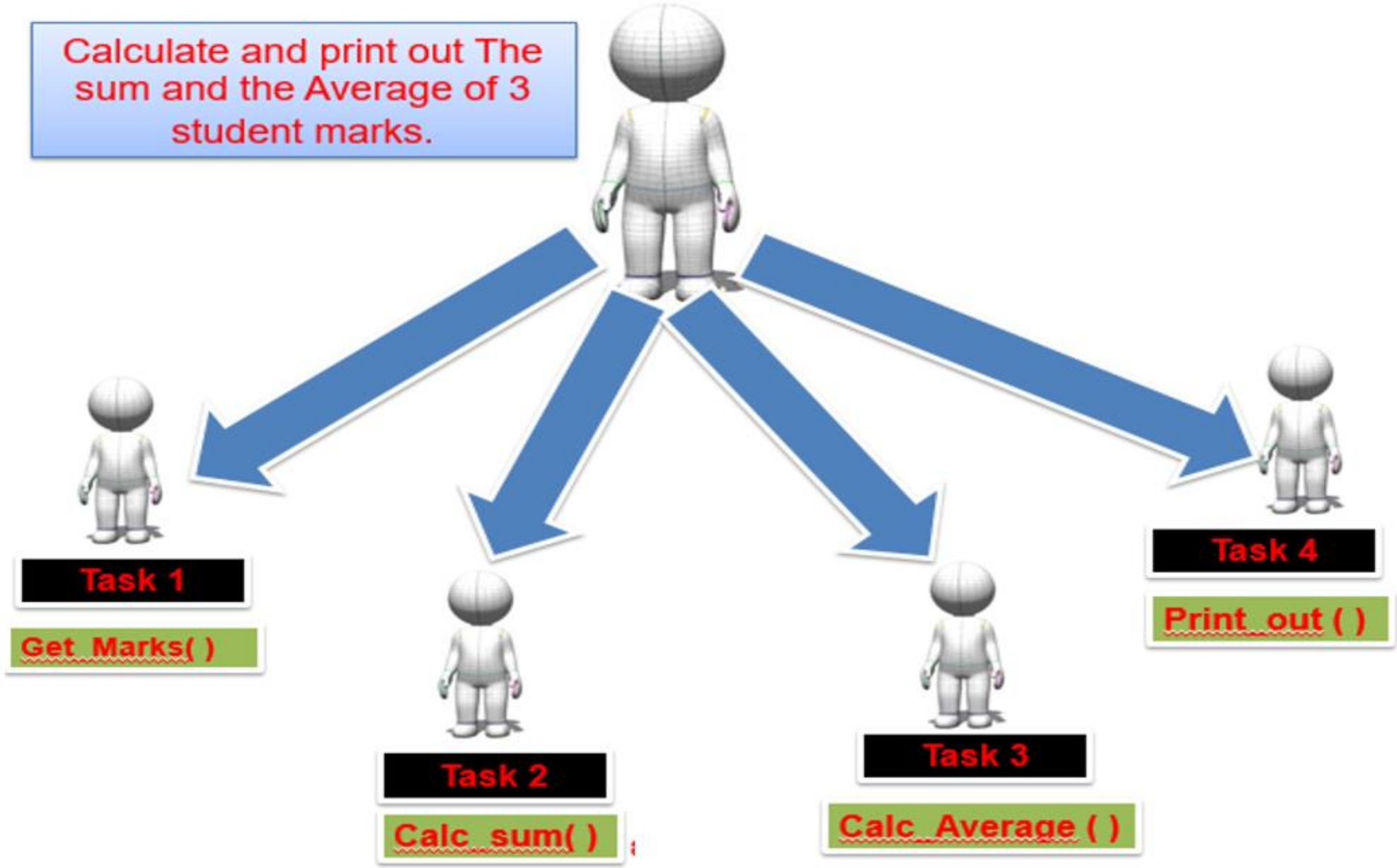
1. Functions:

A function is a set of statements designed to accomplish a particular task. Experience has shown that the best way to develop and maintain a large program is to construct it from smaller pieces or (modules). Modules in C++ are called functions.

Functions are very useful to read, write, debug and modify complex programs. They can also be easily incorporated in the main program. In C++, the main() itself is a function that means the main function is invoking the other functions to perform various tasks. The main advantages of using a function are:

- Easy to write a correct small function.
- Easy to read, write, and debug a function.
- Easier to maintain or modify such a function.
- Small functions tend to be self documenting and highly readable.
- It can be called any number of times in any place with different parameters.


In Real Life
In Programming
Task 1
Function ( )
Task 2
Function ( )
Task 3
Function ( )
Task 4
Function ( )


Calculate and print out The sum and the Average of 3 student marks.
Task 1
Get_Marks( )
Task 2
Calc_sum( )
Task 3
Calc_Average ( )
Task 4
Print_out ( )

2. Defining a Function

A function definition has a name, parentheses pair containing zero or more parameters and a body. For each parameter, there should be a corresponding declaration that occurs before the body. Any parameter not declared is taken to be an integer by default. The general format of the function definition is :

General Form of Function:

```
return-type function-name ( parameters-list )
    {
        (body of function)
        statement1 ;
        statement2 ;
            :
        statement-n ;
        (return something)
    }
```

Functions

Function Definition

```
return-value-type function-name( parameter-list )
{
Lines of code to be
executed
...
...
...
(Body)
}
```

return-value-type: Is the data type for the returned value from the function to the calling program. It is **Mandatory, if no returned value expected , we use keyword Void.**

Parameter-List : Is a list of data values supplied from the calling program as input to the function. It is **Optional**

The type of the function may be int, float, char, etc. It may be declared as type (void), which informs the compiler not to the calling program. For example:

Void function_name (---)

Int function_name (---)

Any variable declared in the body of a function is said to be local to that function. Other variables which are not declared either as arguments or in the function body are considered "global" to the function and must be defined externally. For example

Void square (int a, int b) → a,b are the formal arguments.

Float output (void) → function without formal arguments

Example 1:

```
void printmessage ( )
{
    cout << "University of Technology";
}

void main ( )
{
    printmessage( );
}
```

Example 2:

```
int max (int a, int b)
{
    int c;
    if (a > b)  c = a;
      else   c = b;
    return (c);
}

void main ( )
{
    cout << max (5, 6);
}
```

3. Return Statement:

The keyword return is used to terminate function and return a value to its caller. The return statement may also be used to exit a function without returning a value. The return statement may or may not include an expression. Its general syntax is:

Return;

Return (expression);

The return statements terminate the execution of the function and pass the control back to the calling environment.

the returned value can be used in one of the following scenarios:

- **Save the value for further calculation**

```
Int result;
Result = sum ( x , y );
```

- **Use the value in some calculation**

```
Result = sum(x , y) /2;
```

- **Print the value**

```
Cout << sum(x , y);
```

Example 1

Write C++ program to calculate the squared value of a number passed from main function. Use this function in a program to calculate the squares of numbers from 1 to 10:

```
#include<iostream.h>

int square ( int y )
{
        int z;
        z = y * y;
        return ( z );
}

void main( )
{
        int x;
        for ( x=1; x <= 10; x++ )
            cout << square ( x ) << endl;
}
```

Example 2

Write C++ program using function to calculate the average of two numbers entered by the user in the main program:

```
#include<iostream.h>
float aver (int x1, int x2)
{
        float z;
        z = ( x1 + x2) / 2.0;
        return ( z);
}

void main( )
{
        float x;
        int num1,num2;
        cout << "Enter 2 positive number \n";
        cin >> num1 >> num2;
        x = aver (num1, num2);
        cout << x;
}
```

Example 3

Write C++ program, using function, to find the summation of the following series: $\sum_{i=1}^{n} i^2 = 1^2 + 2^2 + 3^2 + \cdots + n^2$

```
#include<iostream.h>
int summation ( int x)
{
        int i = 1, sum = 0;
        while ( i <= x )
        {
                sum += i * i ;
                i++;
        }
        return (sum);
}
void main ( )
{
        int n ,s;
        cout << "enter positive number";
        cin >> n;
        s = summation ( n );
        cout << "sum is: " << s << endl;
}
```

Example 4

Write a function to find the largest integer among three integers entered by the user in the main function.

```
#include <iostream.h>
int max(int y1, int y2, int y3)
{
        int big;
        big=y1;
        if (y2>big) big=y2;
        if (y3>big) big=y3;
        return (big);
}
void main( )
{
        int largest,x1,x2,x3;
        cout<<"Enter 3 integer numbers:";
        cin>>x1>>x2>>x3;
        largest=max(x1,x2,x3);
        cout<<largest;
}
```

4. Passing Parameters:

There are two main methods for passing parameters to a program:

1) **passing by value**, and 2) **passing by reference.**

A- Passing by value:

When parameters are passed by value, a copy of the parameters value is taken from the calling function and passed to the called function. The original variables inside the calling function, regardless of changes made by the function to it are parameters will not change. All the pervious examples used this method.

```c
/**
 ** This example illustrates calling a function by value
 **/

#include <stdio.h>

void func (int a, int b)
{
    a += b;
    printf("In func, a = %d     b = %d\n", a, b);
}

int main(void)
{
    int x = 5, y = 7;
    func(x, y);
    printf("In main, x = %d     y = %d\n", x, y);
    return 0;
}
```

The output of the program is:

```
In func, a = 12 b = 7
In main, x = 5  y = 7
```

B- Passing by Reference:

When parameters are passed by reference their addresses are copied to the corresponding arguments in the called function, instead of copying their values. Thus pointers are usually used in function arguments list to receive passed references.

This method is more efficient and provides higher execution speed than the call by value method, but call by value is more direct and easy to use.

Example 6:

The following program illustrates passing parameter by reference.

```
#include <iostream.h>

void swap(int *a,int *b)
{
        int t;
        t=*a;
        *a=*b;
        *b=t;
}

void main( )
{
        int x=10;
        int y=15;
        cout<<"x before swapping is:"<<x<<"\n";
        cout<<"y before swapping is:"<<y<<"\n";
        swap(&x,&y);
        cout<<"x after swapping is:"<<x<<"\n";
        cout<<"y after swapping is:"<<y<<"\n";        }
```

جامعة ذي قار
كلية التربية للعلوم الصرفة
قسم علوم الحاسبات
المرحلة الاولى
مادة البرمجة المهيكلة

Lecture 12

Instructor: Zainab Rustum Mohsin

1. Types of Functions:

The user defined functions may be classified in the following three ways based on the formal arguments passed and the usage of the return statement, and based on that, there are three of user defined functions:

1. A function is invoked without passing any formal argument from the calling portion of a program and also the function does not return back any value to the called function.
2. A function is invoked with formal arguments from the calling portion of a program but the function does not return back any value to the calling portion.
3. A function is invoked with formal arguments from the calling portion of a program which return back a value to the calling environment.

Here are two programs that find the square of the number using and not using a return statement.

Example 1:

```
# include <iostream.h>
Void main()
{
Void square (int);
Int max;
Cout<<"Enter a value for n ?\n";
Cin>>max;
For (int i=0;i<=max-1;++i)
Square (i)
}
Void square(int n)
{
Float value;
Value=n*n;
Cout<<"i="<<n<<"square="<<value<<endl;
}
```

```
# include <iostream.h>
Void main()
{
float square (float);
float I,max,value;
max=1.5;
i=-1.5;
while (i<=max) {
value=square(i);
Cout<<"i="<<i<<"square="<<value<<endl;
I=i+0.5;
}
}
Float square(float n)
{
Float value;
Value=n*n;
Return (value);
}
```

Write program with function to check the average if pass or fail

```
#include<iostream>
using namespace std;
//=================
void result(int x)
{
    if(x>=50)
          cout<<"pass"<<endl;
    else
          cout<<"fail"<<endl;
}
//===============
 void main()
  {
     int grade;
     cin>>grade;
  result(grade);
}
```

Write function to check the number if even or odd

```
void oddeven(int x)
    {
    if(x%2==0)
          cout<<"Even"<<endl;
    else
          cout<<"odd"<<endl;
}
```

Write function to print the square to a number

```
 void sqr(int x)
 {
 cout<<x*x<<endl;
}
```

Write program with functions to enter and print array elements

```
#include<iostream>
using namespace std;

int i,s;
int x[100];

void parry(int a[],int n)
                    //الطباعة
{
cout<<"\n THE ARRAY \n";
for (i=1 ; i<=n; i++)
cout<<a[i];
cout<<endl;
}
void readar(int a[],int n)
                   // الادخال
{
cout<<"\nINSERT ARRAY:\n";
for(i=1 ;i<=n;i++)
   {
    cout<<"x["<<i<< "]:";
    cin>>a[i];
    cout<<endl;
  }
}
```

```
void main()
{
 cin>>s;
 readar(x,s);     الاستدعاء
 parry(x,s);      الاستدعاء
}
```

Write functions to find the max element from array

```
void max(int a[],int n)
{
int m;
m=a[0];
for(i=1; i<=n; i++)
if (m<a[i])
 m=a[i];
 cout<<m;
  cout<<endl;
}
```

```
max(x,s);      الاستدعاء
```

Example 2:

write C++ program, using function to find the sumation of the given series: Sum=x-(x^3)/3!+(x^5)/5!- ...(x^n)/n!

```
#include <iostream.h>
Void main(void)
{
Long int fact (int);
Float power(float,int);
Float sum,temp,x,pow;
Int sign,I,n;
Longint factorial;

Cout<<"Enter a value for n?"<<endl;
Cin>>n;
Cout<<"Enter a value for x ?"<<endl;
Cin>>x;
I=3; sum=x;  sign=1;
While (i<=n) {
Factval=fact(i);
Pow=power(x,i);
Sign=(-1)*sign;
Temp=sign*pow/factval;
Sum=sum+temp;
I=i+2;
}
Cout<<"sum of series ="<<sum;
}
```

```
Long int fact (int max)
{
Long intvalue;
Value=1;
For(int i=1;i<=max;++i)
Value=value*I;
Return (value);
}

Float power (float x, int n)
{
Float value2;
Value2=1;
For(int j=1;j<=n;++j)
Value2=value2*x;
Return(value2);
}
```

2. Actual and Formal Arguments:

The arguments may be classified under two groups, actual and formal arguments:

(a) **Actual arguments:** An actual argument is a variable or an expression contained in a function call that replaces the formal parameter which is a part of the function declaration. Sometimes, a function may be called by a portion of a program with some parameters and these parameters are known as the actual arguments.

(b) **Formal arguments:** are the parameters present in a function definition which may also be called as dummy arguments or the parametric variables. When the function is invoked, the formal parameters are replaced by the actual parameters. Formal arguments may be declared by the same name or by different names in calling a portion of the program or in a called function but the data types should b e the same in both blocks

For example:

```
# include <iostream.h>
Void main()
{
Int x,y;
Void output (int x, int y);              // function declaration
___
___
Output (x,y);                    // x and y are actual arguments
}
Void output (int a, int b)               // forma or dummy arguments
{
                    // body of function
 }
```

3. Local and Global variables:

The variables in general bay be classified as local or global variables.

(a) Local variables: Identifiers, variables and functions in a block are said to belong to a particular block or function and these identifiers are known

as the local parameters or variables. Local variables are defined inside a function block or a compound statement. For example,

```
Void func (int l, int j)
{
Int k,m;                    // local variables
.....                       // bodyofthe function
}
```

Local variables are referred only the particular part of a block or a function. Same variable name may be given to different parts of a function or a block and each variable will be treated as a different entity.

(b) Global variables: these are variables defined outside the main function block. These variables are referred by the same data type and by the same name through out the program in both the calling portion of the program and in the function block.

Example 3:

A program to find the sum of the given two numbers using the global variables.

```
#include <iostream.h>
Int x;
Int y=5;

void main( )
{
        X=10;
        Void sum(void);
        Sum();
 }
Void sum(void)
{
        Int sum;
        Sum=x+y;
        Cout<<"x="<<x<<endl;
        Cout<<"y="<<y<<endl;
        Cout<<"sum="<<sum<<endl;
}
```

Example 6

Write C++ program, using function, to find (search) X value in array, and return the index of it's location:

```
#include<iostream.h>

int search( int a[ ], int y)
{
    int  i= 0;
    while ( a [ i ] != y )
      i++;
    return ( i );
}

void main ( )
{
        int  X, f;
        int  a [ 10 ] = { 18, 25, 36, 44, 12, 60, 75, 89, 10, 50 };
         cout << "enter value to find it: ";
         cin >> X;
        f= search (a, X);
       cout << "the value " << X << " is found in location "<< f;
}
```

Assignment

Write a C++ program, using function, to see if a number is an integer (odd or even) or not an integer.

The Fibonacci Series is: 0, 1, 1, 2, 3, 5, 8, 13, 21, ... It begins with the terms 0 and 1and has the property that each succeeding term is the sum of the two preceding terms. Write a C++ program, using function, to calculate the nth Fibonacci number.

Write a C++ program, using function, to calculate the factorial of an integer entered by the user at the main program.

Write a C++ program, using function, to find x^y.

Lecturer : Zainab Rustum Mohsin

Write a C++ program, using function, to test the year if it's a leap or not.

Note: use `y % 4 == 0 && y % 100 != 0 :: y % 400 ==0`

Write a C++ program, using function, that reads two integers (feet and inches) representing distance, then converts this distance to meter.

Note: 1 foot = 12 inch
1 inch = 2.54 Cm

i.e.:

Input: feet: 8 or inches: 9

Write C++ program, using function, to find the summation of student's marks, and it's average, assume the student have 8 marks.

جامعة ذي قار
كلية التربية للعلوم الصرفة
قسم علوم الحاسبات
المرحلة الاولى
مادة البرمجة المهيكلة

Lecture 13

Instructor: Zainab Rustum Mohsin

What is File Handling in C++?

File handling in C++ is a mechanism to store the output of a program in a file and help perform various operations on it. Files help store these data permanently on a storage device.

The term “Data” is commonly referred to as known facts or information. we need to store it somewhere. You all would argue that there are so many text editors like ‘Notepad’ and ‘MS Office’, which help us store data in the form of text, But text editors like ‘Notepad’ and ‘MS Office’ are pre-built and cannot be accessed at the programming level to store data.

File Handling is a hot topic when it comes to storing such programming data.

C++ support a number of functions that perform basic file operations :

1- Create a file

2- Open a file

3- Read from a file

4- Write to a file

5- Close a file

So far, we have been using the **iostream** standard library, which provides **cin** and **cout** methods for reading from standard input and writing to standard output respectively.

to read and write from a file. This requires another standard C++ library called **fstream**,

Sr.No	Data Type & Description
1	**ofstream** This data type represents the output file stream and is used to create files and to write information to files.
2	**ifstream** This data type represents the input file stream and is used to read information from files.
3	**fstream** This data type represents the file stream generally, and has the capabilities of both ofstream and ifstream which means it can create files, write information to files, and read information from files.

To access the following classes, you must include the fstream as a header file like how we declare iostream in the header.

Example:

```
#include<iostream>

#include<fstream>
```

File Operations in C++

C++ provides us with four different operations for file handling. They are:

1. **open()** – This is used to create a file.
2. **read()** – This is used to read the data from the file.
3. **write()** – This is used to write new data to file.
4. **close()** – This is used to close the file.

Closing a file in C++

Closing a file is a good practice, and it is must to close the file. Whenever the C++ program comes to an end, it clears the allocated memory, and it closes the file. We can perform the task with the help of close() function.

Syntax:

```
FileName.close();
```

Opening files in C++

To read or enter data to a file, we need to open it first. This can be performed with the help of 'ifstream' for reading and 'fstream' or 'ofstream' for writing or appending to the file. All these three objects have open() function pre-built in them.

Syntax:

```
open( FileName , Mode );
```

Here:

FileName – It denotes the name of file which has to be opened.

Mode – There different mode to open a file

Lecturer : Zainab Rustum Mohsin

Mode	Description
iso::in	File opened in reading mode
iso::out	File opened in write mode
iso::app	File opened in append mode
iso::ate	File opened in append mode but read and write performed at the end of the file.
iso::binary	File opened in binary mode

In C++, we can use two modes simultaneously with the help of | (OR) operator.

Program for Opening File:

```
1  #include<iostream>
2  #include<fstream>
3
4  int main(){
5      fstream FileName;
6      FileName.open("FileName", ios::out);
7      if (!FileName){
8          cout<<"Error while creating the file"
9      }
10     else{
11         cout<<"File created successfully";
12         FileName.close();
13     }
14     return 0;
15 }
```

Output

File created successfully

Explanation

1. Here we have an iostream library, which is responsible for input/output stream.
2. We also have a fstream library, which is responsible for handling files.
3. Creating an object of the fstream class and named it as 'FileName'.
4. On the above-created object, we have to apply the open() function to create a ne
   file, and the mode is set to 'out' which will allow us to write into the file.
5. We use the 'if' statement to check for the file creation.
6. Prints the message to console if the file doesn't exist.
7. Prints the message to console if the file exists/created.
8. We use the close() function on the object to close the file.

Writing to File

We will use fstream or ofstream object to write
data into the file and to do so; we will use stream insertion operator (<<) along with the text enclosed within the double-quotes.

With the help of open() function, we will create a new file named 'FileName' and then we will set the mode to 'ios::out' as we have to write the data to file.

Syntax:

```
FileName<<"Insert the text here";
```

Program for Writing to File:

```
#include<iostream>
#include<fstream>

int main() {
    fstream FileName;
    FileName.open("FileName.txt", ios::out);
    if (!FileName) {
        cout<<" Error while creating the file ";
    }
    else {
        cout<<"File created and data got written to file";
        FileName<<"This is a blog posted on Great Learning";
        FileName.close();
    }
    return 0;
}
```

Output

File created and data got written to file

Explanation

1. Here we have an iostream library, which is responsible for input/output stream.
2. We also have a fstream library, which is responsible for handling files.
3. Creating an object of the fstream class and named it as 'FileName'.
4. On the above-created object, we have to apply the open() function to create a n file, and the mode is set to 'out' which will allow us to write into the file.
5. We use the 'if' statement to check for the file creation.
6. Prints the message to console if the file doesn't exist.
7. Prints the message to console if the file exists/created.
8. Writing the data to the created file.
9. We use the close() function on the object to close the file.

Reading from file in C++

can perform the reading of data from a file with the CIN to get data from the user, but then we use CIN to take inputs from the user's standard console. Here we will use fstream or ifstream.

Syntax:

```
FileName>>Variable;
```

Content of FileName.txt:

Hello World, Thank You for Visiting Great Learning.

Program for Reading from File:

```
#include<iostream>
#include <fstream>

int main() {
    fstream FileName;
    FileName.open("FileName.txt", ios::in);
    if (!FileName) {
        cout<<"File doesn't exist.";
    }
    else {
        char x;
        while (1) {
            FileName>>x;
            if(FileName.eof())
                break;
            cout<<x;
        }
    }
    FileName.close();
    return 0;
}
```

Output

Hello World, Thank You for Visiting Great Learning.

Explanation

1. Here we have an iostream library, which is responsible for input/output stream.
2. We also have a fstream library which is responsible for handling files.
3. Creating an object of the fstream class and named it 'FileName'.
4. On the above-created object, we have to apply the open() function to create a new file, and the mode is set to 'in' which will allow us to read from the file.
5. We use the 'if' statement to check for the file creation.
6. Prints the message to console if the file doesn't exist.
7. Creating a character(char) variable with the named x.
8. Iterating of the file with the help of while loop.
9. Getting the file data to the variable x.
10. Here we are using if condition with eof() which stands for the end of the file to tell the compiler to read till the file's end.
11. We use the 'break' statement to stop the reading from file when it reaches the end.
12. The print statement to print the content that is available in the variable x.
13. We use the close() function on the object to close the file