C++ Programming

Ninevah University
College of Electronics Engineering
Department of Electronic Engineering

2nd Year
2024 – 2025

Lecturer No.8
Prof Dr. Qais Thanon

All the lectures of this course will upload at the Google classroom

Two-Dimensional Arrays

- Arrays that we have consider up to now are one dimensional arrays, a single line of elements.

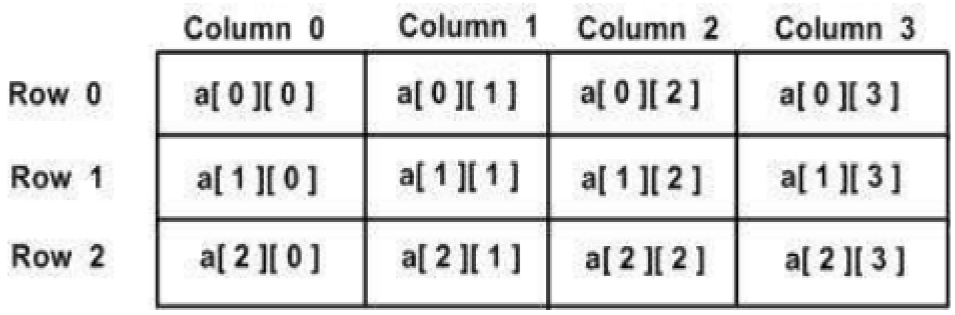
- Often data come naturally in the form of a table, e.g., spreadsheet, which need a two-dimensional array. The simplest form of the multidimensional array is the two-dimensional array.

- A two-dimensional array is, in essence, a list of one-dimensional arrays. To declare a two-dimensional integer array of size x, y, you would write something as follows:

```
type arrayName [ x ][ y ];
```

where x and y should be integers

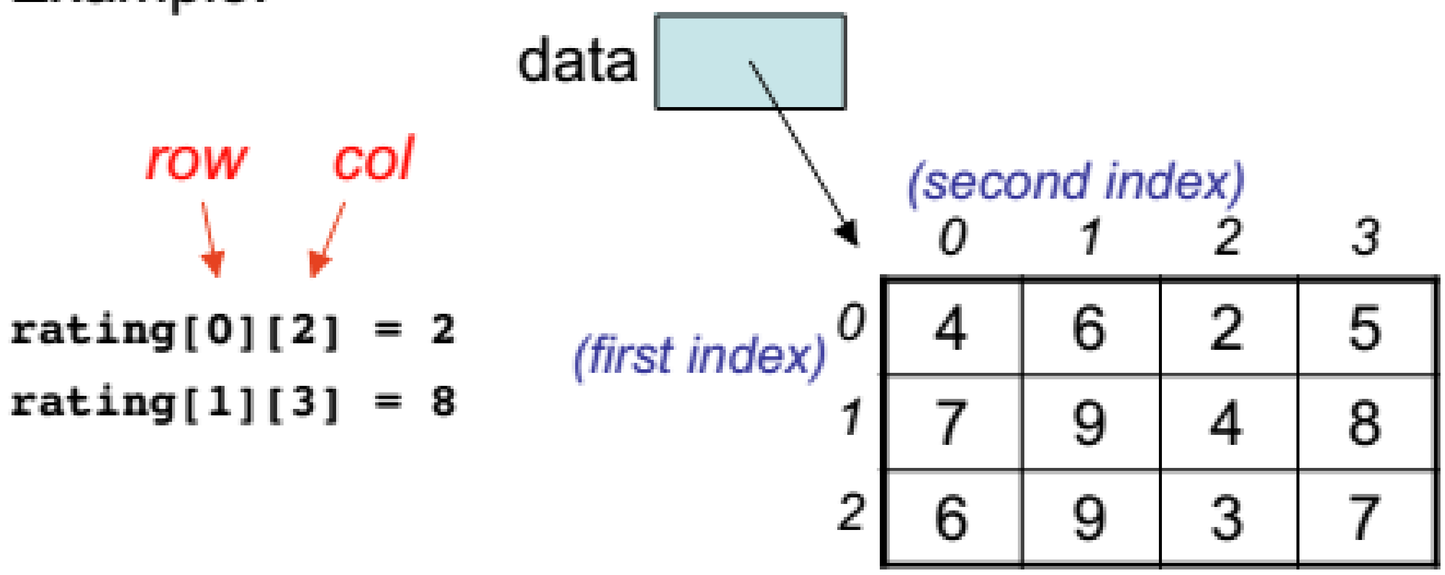
```
int a [ 3 ][ 4 ];
```



	Column 0	Column 1	Column 2	Column 3
Row 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]

Two-Dimensional Arrays

- Two-dimensional (2D) arrays are indexed by two subscripts, one for the row and one for the column.
- Example:



```
rating[0][2] = 2
rating[1][3] = 8
```

$$\begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & 4 & 6 & 2 & 5 \\ 1 & 7 & 9 & 4 & 8 \\ 2 & 6 & 9 & 3 & 7 \end{array}$$

Similarity with 1D Arrays

- Each element in the 2D array must by the same type,
- Subscripted variables can be use just like a variable:

```
rating[0][3] = 10;
```

- Array indices must be of type `int` and can be a variable, or expression.

```
rating[3][j] = j;
```

Initializing Two-Dimensional Arrays

Multi-dimensioned arrays may be initialized by specifying bracketed values for each row. Following is an array with 3 rows and each row have 4 columns.

```
int a[3][4] = { {0, 1, 2, 3} ,   /* initializers for row indexed by 0 */
                {4, 5, 6, 7} ,   /*initializers for row indexed by 1 */
                 {8, 9, 10, 11}  /*initializers for row indexed by 2 */
                      };
```

The nested braces are optional

```
int a[3][4] = {0,1,2,3,4,5,6,7,8,9,10,11};
```

How to enter data in a Two Dimensional Arrays

Indirect Initializing

Nested loop is used to enter data in 2-D arrays.

Suppose you want to fill an array with dimension 5x5 with number 10

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

void main(){

int matrix [5][5];
for (int m1=0 ; m1<5 ; m1++)
                 {
for (int m2=0 ; m2<5 ; m2++)
                    {
          matrix [m1][m2] = 10 ;

                    }

                }
getch();
}
```

10	10	10	10	10
10	10	10	10	10
10	10	10	10	10
10	10	10	10	10
10	10	10	10	10

Suppose the 5×5 array as shown below, write a C++ program count:

1. How many positive number in this array?
2. The average of odd numbers?

```
#include<iostream.h>
#include<conio.h>
 void main(){
 int i, j, N=0, SUM=0, d=0,m[5][5] =
{7,-3,1,17,3,5,9,5,-21,11,1,-5,12,10,
-2,-21,9,3,-6,8,8,-7,-12,-3,11};

for (i = 0 ; i < 5 ; i++){
for (j = 0 ; j < 5 ; j++){
if (m[i][j]>0) N++;
if( m[i][j]%2 != 0) {SUM+=m[i][j];
                      d++;}
                       }
                       }
cout << "\n\n the number positive values is"<<N;
cout << "\n\n the average of odd numbers is"<<SUM/d;

getch();
}
```

7	-3	1	17	3
5	9	5	-21	11
1	-5	12	10	-2
-21	9	3	-6	8
8	-7	-12	-3	11

Can a Two-Dimensional Array used for Character?

```
#include<iostream.h>
#include<conio.h>
 void main(){
char cmatrix [3][3];
int q1, m2;
for (q1=0 ; q1<3 ; q1++){
for (m2=0 ; m2<3 ; m2++){
    cout<<"Enter name :";
    cin>>cmatrix [q1][m2];
                                }
}
 // For displaying elements of a matrix on a screen //
for (q1=0 ; q1<3 ; q1++){
for (m2=0 ; m2<3 ; m2++){
cout<<cmatrix [q1][m2] << "\t";
}
cout<<"\n";
}
 getch();
}
```

R	m	a
K	z	v
T	E	Q

C++ Programming

Ninevah University
College of Electronics Engineering
Department of Electronic Engineering

2nd Year
2024 – 2025

Lecturer No.9

Prof Dr. Qais Thanon

FUNCTIONS IN C++

All the lectures of this course will upload at the Google classroom

✓ What is the function in C++?

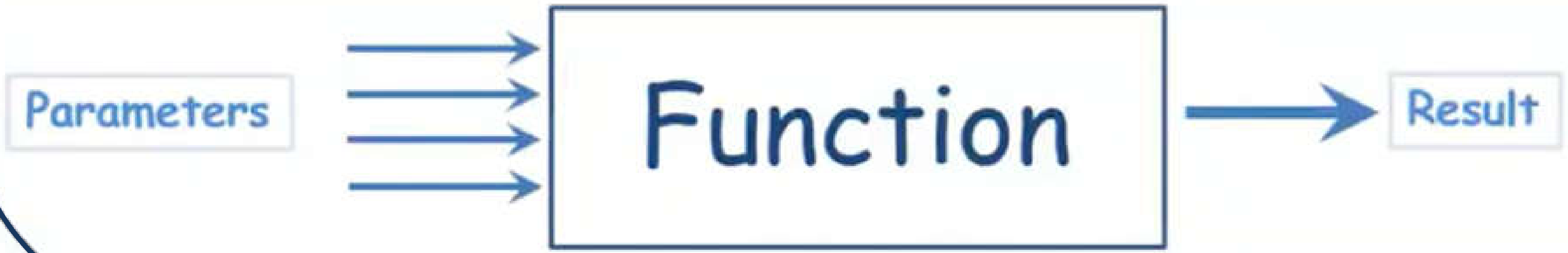
A function is block of code which is used to perform a particular task.

✓ Why we should use the function in C++?

- A program may need to repeat the same piece of code at various places.
- It may be required to perform certain task repeatedly.
- The program may become very large if functions are not used.

- Easier to Code
- Easier to Modify
- Easier to Maintain
- Reusability
- Less Programming Time
- Easier to Understand

The real reason for using function is to divide program into different parts.



There are two types of function:

1.Standard Library Functions: Predefined in C++

2.User-defined Function: Created by users

In this lecture, we will focus mostly on user-defined functions.

C++ allows the programmer to define their own function.

A user-defined function groups code to perform a specific task and that group of code is given a name (identifier).

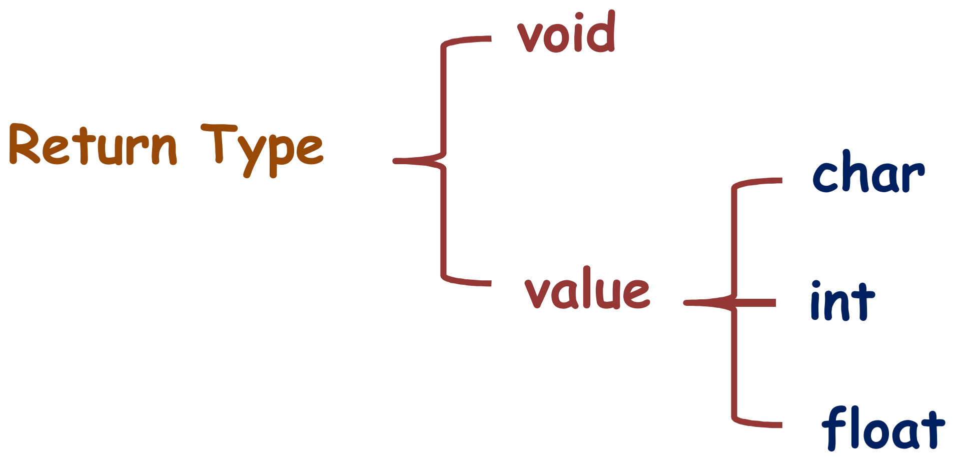
When the function is invoked from any part of the program, it all executes the codes defined in the body of the function.

There are 3 aspects in each C++ function. They are,

- ✓ Function declaration or prototype – This informs compiler about the function name, function parameters and return value's data type.
- ✓ Function call – This calls the actual function
- ✓ Function definition – This contains all the statements to be executed.

The syntax to declare a function is:

```
returnType functionName (parameter1, parameter2,...)
        {
// function body
        }
```



A function is *called* by specifying its name followed by its arguments.

Non-value returning functions:

```
    function_name (data passed to function);
```

Value returning functions:

```
    results = function_name (data passed to function);
```

Before using any function it must be defined in the program. Function definition has three principal components: the first line, the parameter declarations and the body of the functions.

The first line of a function definition contains the data type of the information return by the function	function name*	set of arguments or **parameters**, separated by commas and enclosed in parentheses
int	`FACT`	`(a)`
float	`D1`	`(x1, x2)`
long	`COM_2`	`(m, n, k)`
double	`Calc`	`(y_1)`
data-type	*function-name*	(formal argument 1, formal argument 2…formal argument n)

Example: Write a C++ program to find the maximum between two numbers.

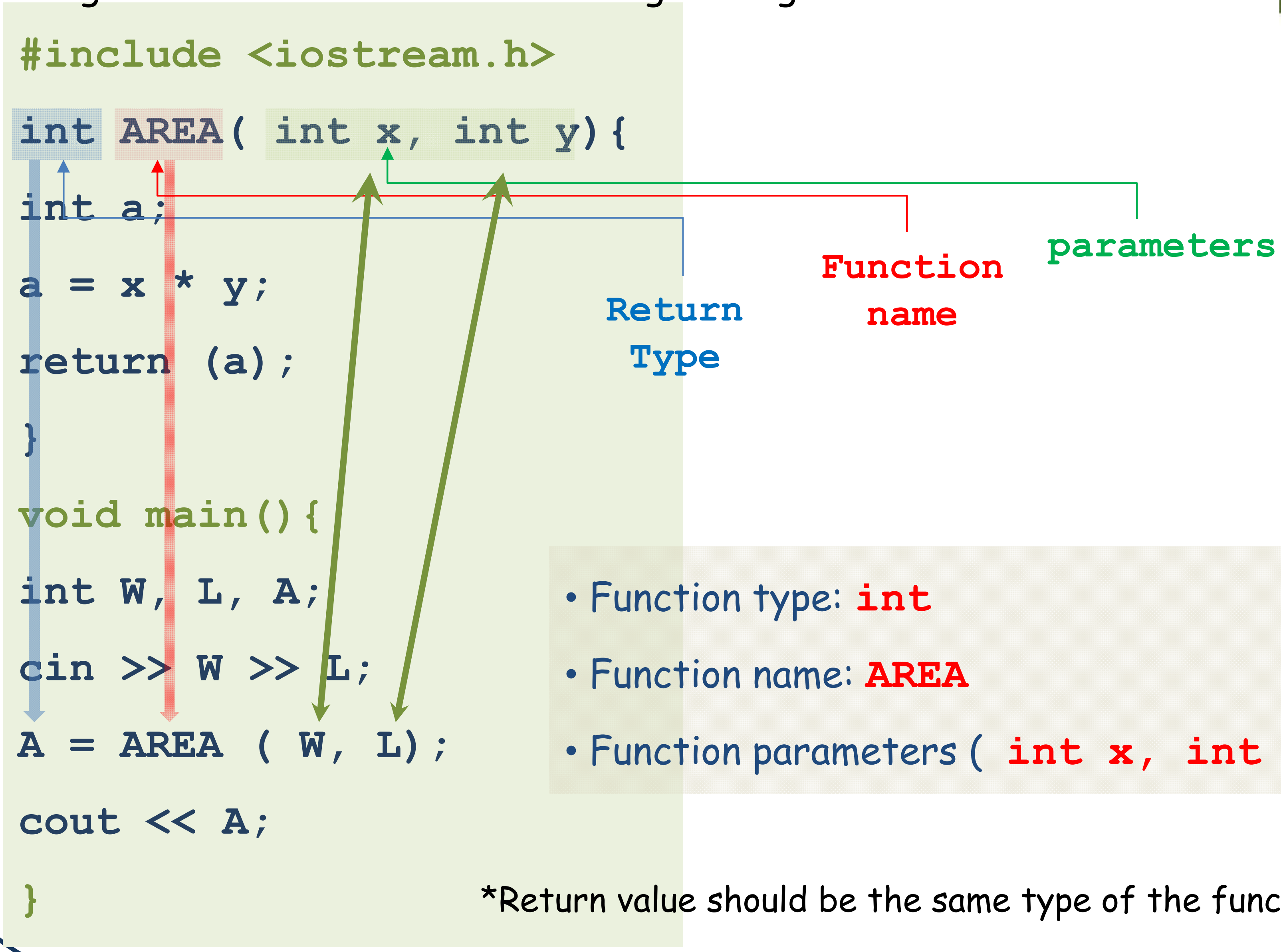
```
#include <iostream.h>

void main(){
 int a , b , c;
 cin >> a >> b;
 if(a > b) c = a;
 else c = b;
 cout << c;
}
```

Without function

```
#include <iostream.h>

int maxi (int x, int y){
 int m;
if(x > y) m = x;
 else m = y;
return m;
}

void main(){
 int a , b , c;
 cin >> a >> b;
 c = maxi (a, b);
 cout << c;
}
```

Value Return Type

Program to find the area of rectangle using function.



```
#include <iostream.h>
int AREA( int x, int y){
int a;
a = x * y;
return (a);
}
void main(){
int W, L, A;
cin >> W >> L;
A = AREA ( W, L);
cout << A;
}
```

- Function type: **int**
- Function name: **AREA**
- Function parameters (**int x, int y**)

*Return value should be the same type of the function

Example: Write a C++ program to find the value of the following series.

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots + \frac{x^n}{n!} + \ldots$$

```
#include<iostream.h>
#include<math.h>

long FACT(int n){
long f = 1;
for( int i=1; i<=n; ++i)
  f* = i;
 return (f);
}
void main(){
        int N, x;
        float Res =0.0;
        cout <<"Enter the number of treams :";
        cin >> N;
        cin >> x;
        for( int i = 0; i <= N; i++)
        Res+= pow(x,i)/FACT(i);
        cout<< Res;
        getch();}
```

Can function return more than one value?

In terms of the keyword return, no.

But it is possible to function contains two or more return statements but only one can return value to the main program.

Example: write a C++ program to find the maximum between two float variables using function.

```
#include<iostream.h>

float COMP (float N1, float N2){
 if(N1 > N2) return N1;
 else return N2;
}
```

```
void main(){
 float x , y , z;
 cin >> x >> y;
 z = COMP (x, y);
 cout << z;
}
```

⇔

```
void main(){
 float x , y;
 cin >> x >> y;
 cout << COMP (x, y);
}
```

Example: write a C++ program to find the maximum between three float variables using function.

```
#include<iostream.h>
```

```
float COMP (float N1, float N2){
 if(N1 > N2) return N1;
 else return N2;
}
```

```
void main(){
 float x , y , z, M;
 cin >> x >> y >> z;
 M = COMP (x, y);
 M = COMP (M, z);
 cout << M;
}
```

⟺

```
void main(){
 float x , y , z, M;
 cin >> x >> y >> z;
 M = COMP (x, COMP(y,z));
 cout << M;
}
```

```
cout << COMP (x, COMP(y,z));
```

Mathematical calculation can be made in return statements

Example: Write C++ program, using function, to find the area of: [1] Circle. [2] Triangle. [3] Rectangle.

```
#include<iosream.h>
#include<conio.h>
#include<math.h>
// FUNCTIONS SHOULD BE HERE
//
float CIRCLE(float R){
return (R * R * M_PI); }
```

Circle function

```
int TRIANGLE(int BASE, int
HIGHT){
return (BASE * HIGHT/2); }
```

Triangle function

```
int Rectangle(int WIDTH, int LENGTH){
return (WIDTH * LENGTH); }
```

Rectangle function

```
void main(){
int S, x, y ;
float Area, r;
clrscr();
cout<<"\n For circle enter 1 \n For triangle enter
2 \n For rectangle enter 3 ");
cin>> S;
switch(S){
case 1: cin >> r;
Area = CIRCLE(r); break;
case 2: cin>> x >> y;
Area = TRIANGLE(x, y); break;
case 3: cin>> x >> y;
Area = Rectangle(x, y);
}
cout << Area;
getch();}
```

YouTube IQ https://www.youtube.com/watch?v=-H-lUbMLy00

C++ Programming

Ninevah University
College of Electronics Engineering
Department of Electronic Engineering

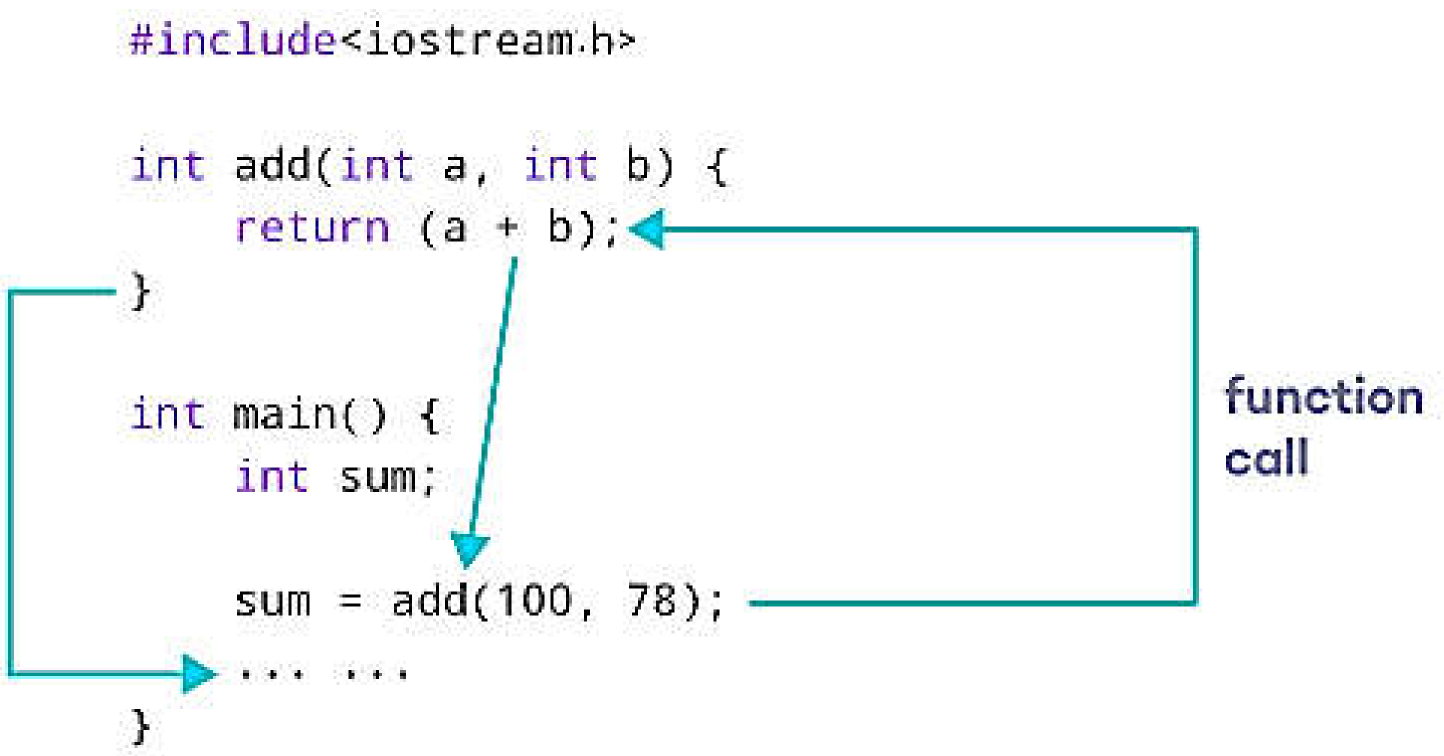
2nd Year
2024 – 2025

Lecturer No.10
Prof Dr. Qais Thanon

FUNCTIONS IN C++ (Part II)

All the lectures of this course will upload at the Google classroom

Functions in C++



```
#include<iostream.h>

int add(int a, int b) {
    return (a + b);
}

int main() {
    int sum;

    sum = add(100, 78);
    ... ...
}
```

Example: write a C++ program to find the maximum between three float variables using function.

```
#include<iostream.h>
```

```
float COMP (float N1, float N2){
 if(N1 > N2) return N1;
 else return N2;
}
```

```
void main(){
 float x , y , z, M;
 cin >> x >> y >> z;
 M = COMP (x, y);
 M = COMP (M, z);
 cout << M;
}
```

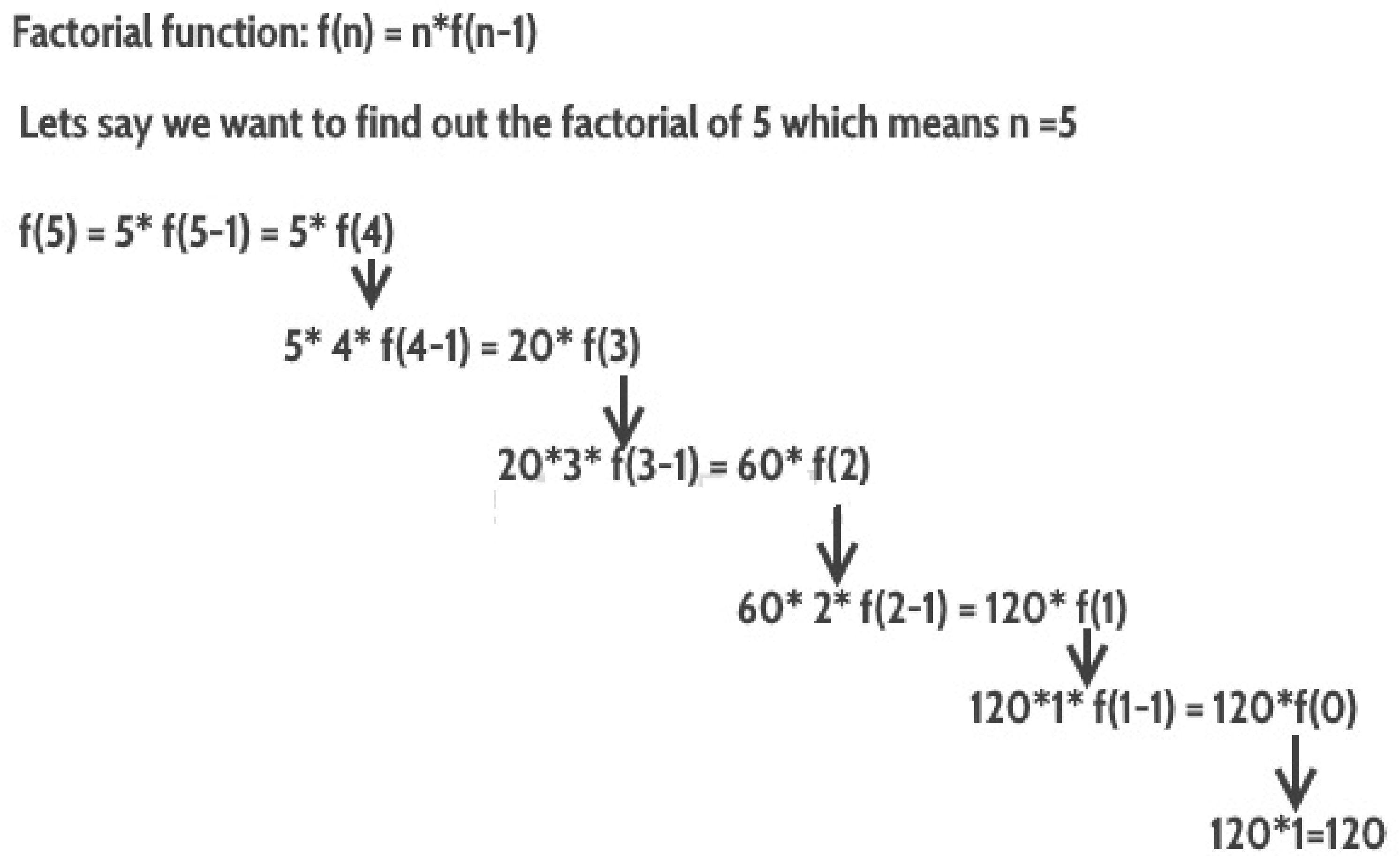
⟺

```
void main(){
 float x , y , z, M;
 cin >> x >> y >> z;
 M = COMP (x, COMP(y,z));
 cout << M;
}
```

```
cout << COMP (x, COMP(y,z));
```

Recursive function

The process in which a function calls itself is known as recursion. The popular example to understand the recursion is factorial function.



Factorial function: f(n) = n*f(n-1)

Lets say we want to find out the factorial of 5 which means n =5

f(5) = 5* f(5-1) = 5* f(4)

↓

5* 4* f(4-1) = 20* f(3)

↓

20*3* f(3-1) = 60* f(2)

↓

60* 2* f(2-1) = 120* f(1)

↓

120*1* f(1-1) = 120*f(0)

↓

120*1=120

Example: Write C++ program to find the factorial of any integer number using function.

```
#include <iostream.h>
```

```
double fa(int n){
double F=1;
for(int i=1; i<=n; i++)
F*= i;
return (F);
}
```

```
double fa(int n){
if(n<=1) return 1;
else
return n*fa(n-1);
}
```

```
void main (){
double FACT;
int k;
cin>> k;
FACT = fa( k );
cout<< FACT<<"\n";}
}
```

Pass Array to Function

In C++, we can pass arrays as an argument to a function.

The syntax for passing an array to a function is:

```
returnType functionName (dataType arrayName[arraySize])
 {
Code;
}
```

Let's see an example,

```
int total(int marks[5])
{
Code ;
}
  void main(){
  //  ACCESSEMENT OF A FUNCTION //
  var = functionName (arrayName);
  }
```

Example: Write a C++ program, using function, find mean value of N entered integers?

$$m = \frac{\text{sum of the terms}}{\text{number of terms}}$$

```
#inculde<iostream.h>
  double mean(double arr[100], int n){
  double sum = 0.0;
  for (int i = 0; i < n; i++)
  sum += arr[i];
  return sum/n;
  }
void main(){
double arr[100], sampleMean;
int size;
cout << "Enter the value of N \n";
cin >> size;
cout << "Enter %d real numbers \n";
for (int i = 0; i < size; i++)
cin>> arr[i];
// Call functions //
sampleMean = mean(arr, size);
cout<< "Mean ="<< sampleMean;
}
```

Example: Write a C++ program, using function, find count the odd number in 5X5 integer array?

```
#inculde<iostream.h>

  int ODD(int arr[5][5]){
  int sum = 0, i, j;
  for (i = 0; i < 5; i++)
  for (j = 0; j < 5; j++)
  if (arr[i][j]%2 !=0) sum ++;
  return sum;
  }
void main(){
int arr[5][5], i, j;
cout << "Enter the elements of the array\n";
for ( i = 0; i < 5; i++)
for (j = 0; j < 5; j++)
cin>> arr[i][j];
// Call functions //
cout<< "the odd numbers ="<< ODD(arr);
}
```

C++ Programming

Ninevah University
College of Electronics Engineering
Department of Electronic Engineering

2nd Year
2024 – 2025

Lecturer No.11
Prof Dr. Qais Thanon

Functions in C++ language   Part III

All the lectures of this course will upload at the Google classroom

12/11/2024

Non value-returning functions void function

There is another type of function which is called void function. This functions with no type. A void() cannot return a value that can be used.

The syntax shown below for functions:

```
void name ( argument1, argument2  ...)
{ statements;  }



void main(){
//  ACCESSEMENT OF A FUNCTION //
name ( actual argument1, actual argument2 ...)
}
```

Calling a function

Example: Write a C++ program to find the maximum between two numbers.

```
#include <iostream.h>
/*function definition*/
int maxi(int x,int y) {
int z;
if(x>=y)z=x;
else z=y;
return (z); }

void main() {
int a, b, M;
cin >> a >> b;
/*call function*/
M = maxi(a, b);
Cout << R;
}
```

```
#include <iostream.h>
/*function definition*/
void maxi(int x,int y) {
int z;
if(x>=y)z=x;
else z=y;
cout << z; }

void main() {
int a, b;
cin >> a >> b;
/*call function*/
maxi(a, b);
}
```

Example :Write a C/C++ program using function to swap two integer numbers?

```
#include<iostream.h>
#include<conio.h>
void SWAP(int x1, int x2){
int TEMP;
TEMP = x2;
x2 = x1;
x1 = TEMP;
cout << "first no.=" << x1;
cout << "Second no.=" << x2;
}
void main(){
int N1, N2;
clrscr();
cout<"Enter the numbers to be swapped :";
cin>> N1 >> N2;
SWAP(N1, N2);
getch();
}
```

Can the void function be recursive?

Example: Write a program in C to print first 50 natural numbers using recursion

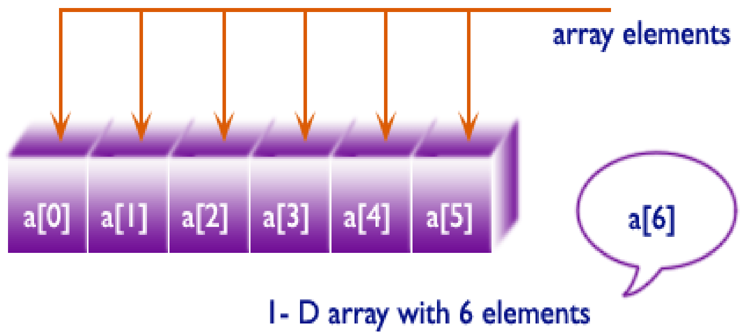
```
#include <iostream.h>

void nat_pnt(int z){
if(z<=50){
cout<< z;
 nat_pnt (z+1);}
}
void main(){

int n=1;

nat_pnt( n );
}
```

First 50 natural numbers

```
1  2  3  4  5  6  7  8  9  10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30 31
32 33 34 35 36 37 38
39 40 41 42 43 44 45
46 47 48 49 50
```

Example: Write a program in C to print the array elements using recursion.

```
#include <iostream.h>

void ArrayEle(int a[6], int n)
{
if(n<6){
cout<< a[n]<<"\t";
 ArrayEle(a,n+1);}
}


void main (){
int arr[6], i;
for (i=0; i<6; i++)
cin >> arr[i];
ArrayEle (arr,0);
}
```



Input the number of elements to be stored in the array :6
Input 6 elements in the array:
element- 0 => -4
element- 1=>   7
element- 2 => 9
element -3 => 0
element -4 => 11
element 5 => -1
Expected Output:
The elements in the array are : -4 7 9 0 11 -1

Example: Write a program in C to convert a decimal number to binary using recursion.

```
#include <iostream.h>

void DIG(int z){
cout<< z%2;
 if (z!=0)
  DIG (z/2);
}

void main (){
int n;
cin>> n;
DIG( n );
}
```

Input any decimal number : 66
Expected Output:
The Binary value of decimal no. 66 is :
 1000010

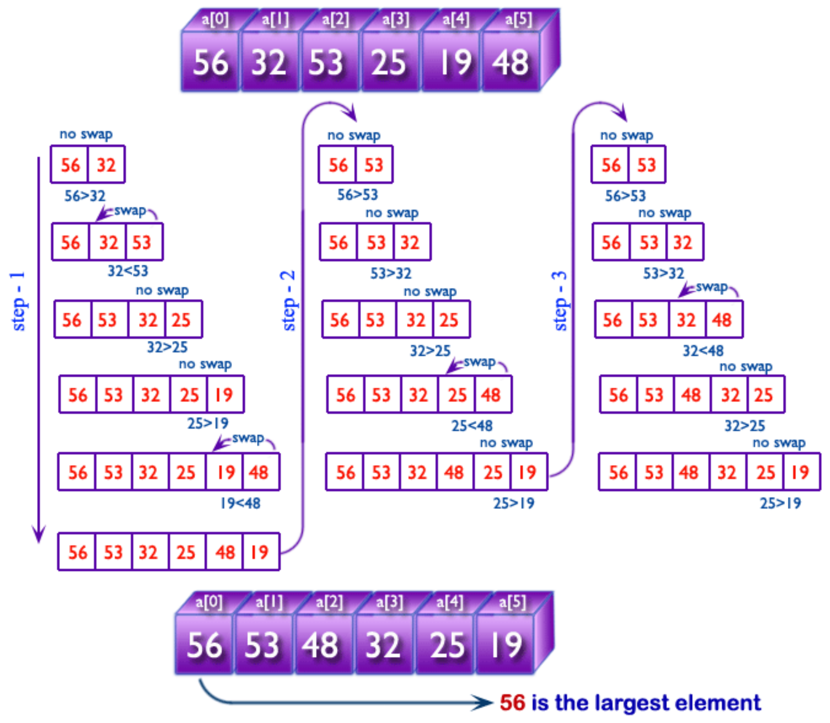
Example: Write a program in C to find the sum of digits of a number using recursion.

```
#include <iostream.h>

int cal(int N){
if(N == 0) return 0;
return ((N % 10) + cal(N / 10));
}

void main (){
int number, sum;
cin >> number;

sum = cal (number);
cout << sum;
}
```

```
The Sum of digits of 371=11
```

Example: Write a program in C to get the largest element of an array using recursion

```
#include <iostream.h>
int MaxE (int arr1[5], int n)
{
static int i=0,High =-9999;
if(i < n) {
      if(High<arr1[i])
        High=arr1[i];
        i++;
        MaxE (arr1, 5); }
       return High;
}
void main (){
int arr1[5], i, M;
for (i=0; i<5; i++)
cin >> arr1[i];
M=MaxE (arr1,5);
cout << M;
}
```



What is the static variable?

Static variables have a property of preserving their value even after they are out of their scope

```
#include<iostream.h>
int fun() {
   int count = 0;
   count++;
   return count;
}

void main(){
 cout << fun();
 cout << fun();
}
```

```
1  1
```

```
#include<iostream.h>
int fun() {
   static int count = 0;
   count++;
   return count;
}

void main(){
 cout << fun();
 cout << fun();
}
```

```
1  2
```

Example: Write a program in C to find the power of any integer number using recursion.

```
#include <iostream.h>

int POWR(int B, int P){
if(P == 0) return 0;
return (B * POWR(B , (P-1)));
}

void main (){
int Result, m, n;
cin >> m >> n;

Result = POWR (m, n);
cout << Result;
}
```

Result = m^n

Solve this program using a void function

C++ Programming

Ninevah University
College of Electronics Engineering
Department of Electronic Engineering
MEDICAL INSTRUMENTATION

2nd Year
2024 – 2025

Lecturer
Prof Dr. Qais Thanon

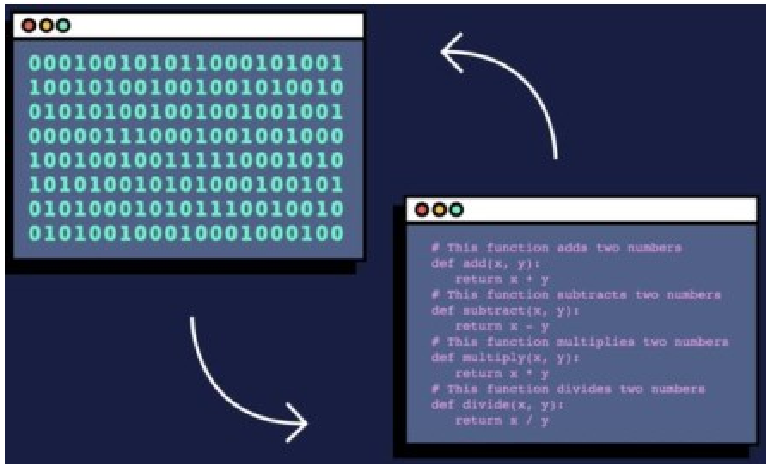
Fundamentals of C++

All the lectures of this course will upload at the Google classroom

Introduction

What is programming language?

A programming language is a vocabulary and set of grammatical rules for instructing a computer or computing device to perform specific tasks.



```
000100101011000101001
100101001001001010010
010101001001001001001
000001110001001001000
100100100111110001010
101010010101000100101
010100010101110010010
010100100010001000100
```

```
# This function adds two numbers
def add(x, y):
    return x + y
# This function subtracts two numbers
def subtract(x, y):
    return x - y
# This function multiplies two numbers
def multiply(x, y):
    return x * y
# This function divides two numbers
def divide(x, y):
    return x / y
```

The term programming language usually refers to high-level languages, such as BASIC, **C**, **C++**, COBOL, Java, FORTRAN, Ada, and Pascal.

C++
Java
Top Programming Languages
PHP
Python

What is the C++ programming language used for?

It is used when a low-level programming language is necessary. While C++ is commonly used for develop Desktop based applications, Games and Gaming Engines, 2D and 3D animations, Developing Web Browsers, Database Software, Media Access Software, Compilers, Operating Systems, Printing and Scanning Applications, Engineering and Medical Applications, Embedded and Real-time Applications.

Why C++ language get the most interest?

C++ is an object-oriented programming language which gives a clear structure to programs and allows code to be reused, lowering development costs.

LANGUAGE CHARACTER SET AND TOKENS

types of tokens:

1. Reserved words (keywords)
2. Identifiers
3. Constants
4. String literals
5. Punctuators
6. Operators

1. Reserved words :

Identify language entities, they have special meanings to the compiler. C reserved words must be typed fully in lowercase. Some examples of reserved words from the program are const, double, int, and return.

2. Identifiers

Programmer-defined words. Needed for program variables, functions, and other program constructs. Must be unique within the same scope

1. A to Z , a to z , 0 to 9 , and the underscore “_”
2. The first character must be a letter
3. Only the first 32 characters as significant.
4. There can be no embedded blanks.
5. Reserved words cannot be used as identifiers.
6. Identifiers are case sensitive.

3. Constants :

fixed values

```
MAX_V = 12.175
```

4. String Literals

characters surrounded by double quotation marks.

"NINEVAH UNIVERSITY"

5. Punctuators

[ ] ( ) { } , ; : ……… * #

6. Operators

result in some kind of computation or action

```
Final_R= int_value * n ;
```

THE STRUCTURE OF a C++ PROGRAM

C++ program consists of following components:

1. Program comments

   use /* and */ to surround comments, or // to begin comment lines.

2. Preprocessor directives

   Lines that begin with a pound sign, #,

3. Type declarations

   int data_in;

4. Named constants

   const double CITY_TAX_RATE = 0.0175;

5. Statements

A statement is a specification of an action to be taken by the computer as the program executes.

Compound Statements is a list of statements enclosed in braces, { }

6. Function declarations (prototypes)
7. Function definitions
8. Function calls

Program structure in C++

The Basic Structure of C++ Program

```
#include<XXXX.h>
void main(){
statement 1;
statement 2;
statement 3;
……
statement n;
}
```

The program begins with the including the libraries using

```
#include < >
```

Between the two tags the name of the directive file (library) the required in the code is written

It can be use more than one directive file (library)

The code start with main function `main()`

The code begin with { and end with }

- ✓ All the language statements and functions are written in lowercase
- ✓ Each line should end with semicolon ;
- ✓ Comments can be written with backslash \\\\

Variables in C++

A variable is a name given to a memory location. It is the basic unit of storage in a program.
The value stored in a variable can be changed during program execution.
A variable is only a name given to a memory location, all the operations done on the variable effects that memory location.

How to declare variables?

A typical variable declaration is of the form:

```
// Declaring a single variable
type variable_name;

// Declaring multiple variables:
type variable1_name, variable2_name, variable3_name;
```

In C++, all the variables must be declared before use. A variable name can consist of alphabets (both upper and lower case), numbers and the underscore '_' character. However, the name must not start with a number.

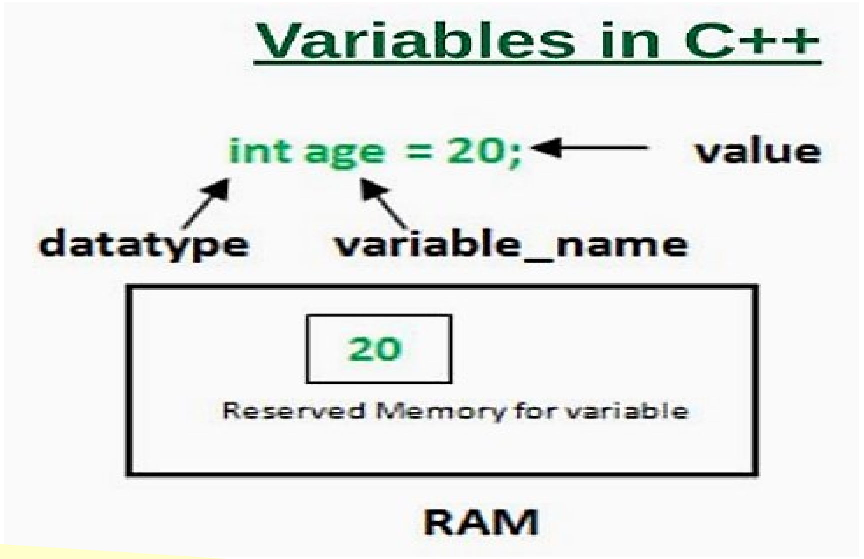
Fundamental Variable Types

There are following basic types of variable in C++

1 `char`	Typically a single octet (one byte). This is an integer type.
`int`	The most natural size of integer for the machine.
3 `float`	A single-precision floating point value.
4 `double`	A double-precision floating point value.

Example:

```
int i, j, k_1;
char c, ch;
float f, salary;
double d;
```



DON'T use C++ keywords as variable names.

Type	Size	Values
unsigned short int	2 bytes	0 to 65,535
short int	2 bytes	–32,768 to 32,767
unsigned long int	4 bytes	0 to 4,294,967,295
long int	4 bytes	–2,147,483,648 to 2,147,483,647
int (16 bit)	2 bytes	–32,768 to 32,767

Determining the Size of a Data Type

• The sizeof operator may be used to determine the size of a data type on any system.

Example **(sizeof (data type)**

```
#include <iostream.h>
void main() {
cout << "Size of char : " <<      sizeof(char);
cout << "Size of int : " <<       sizeof(int);
cout <<"Size of short int : "<<   sizeof(short int);
cout << "Size of long int : " << sizeof(long int);
cout << "Size of float : " <<     sizeof(float);
cout << "Size of double : " <<    sizeof(double);
}
```

C++ Programming

Ninevah University
College of Electronics Engineering
Department of Electronic Engineering
MEDICAL INSTRUMENTATION

2nd Year
2024 – 2025

10:30 – 12:30

Lecturer
Prof Dr. Qais Thanon
Lecture #2

All the lectures of this course will upload at the Google classroom

The cout Object:

Use the cout<< object to display information on the computer's screen.

- Its job is to output information using the standard output device.
- The << operator is used to send the string like "NINEVAH UNIVERSITY" to cout.
- cout does not produce a newline at the end of a statement

```
# include <iostream.h >
void main ()  {
 cout << " *** University of NINEVAH ***";
}
```

```
*** University of NINEVAH ***
```

The cin Object

- The cin>> object reads information types at the keyboard.
- Notice the >> and << operators appear to point in the direction information is flowing.

Arithmetic Operators

•There are many operators for manipulating numeric values and performing arithmetic operations

Operator	Meaning	Example
+	Addition	`total = cost + tax;`
–	Subtraction	`cost = total - tax;`
*	Multiplication	`tax = cost * rate;`
/	Division	`salePrice = original / 2;`
%	Modulus	`remainder = value % 3;`

```
# include <iostream.h >

 void main () {
 int x;
 x = 4 + 3;
 cout << x / 3.0 << " " << x * 2;
 }
```

```
x = 7
7/3 = 2
7 * 2 =14
```

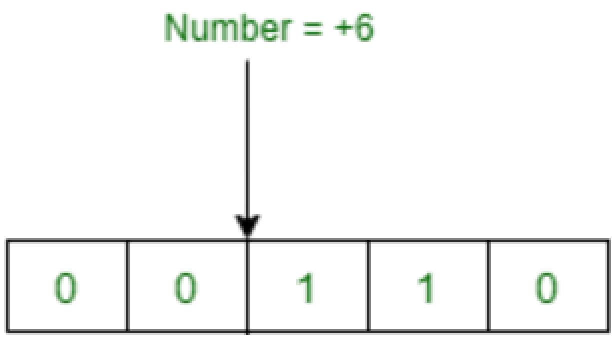
```
x = 7
7/3.0 = 2.33
7 * 2 =14
```

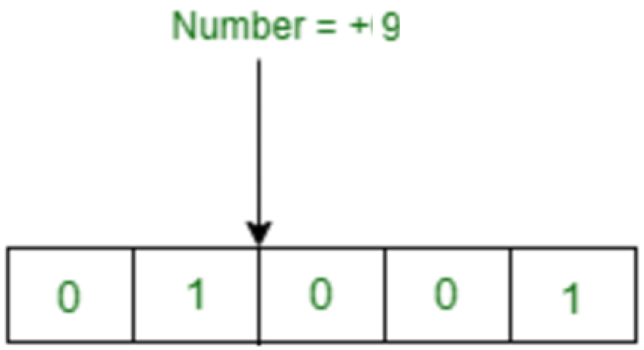
Calculations can be performed in a output statement

int / int = int
int / float = float

Bitwise operators

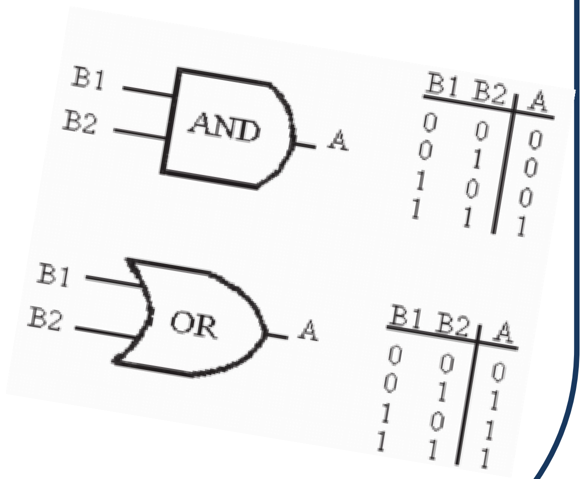
C++ provides bitwise operators, which provide bit-level control.



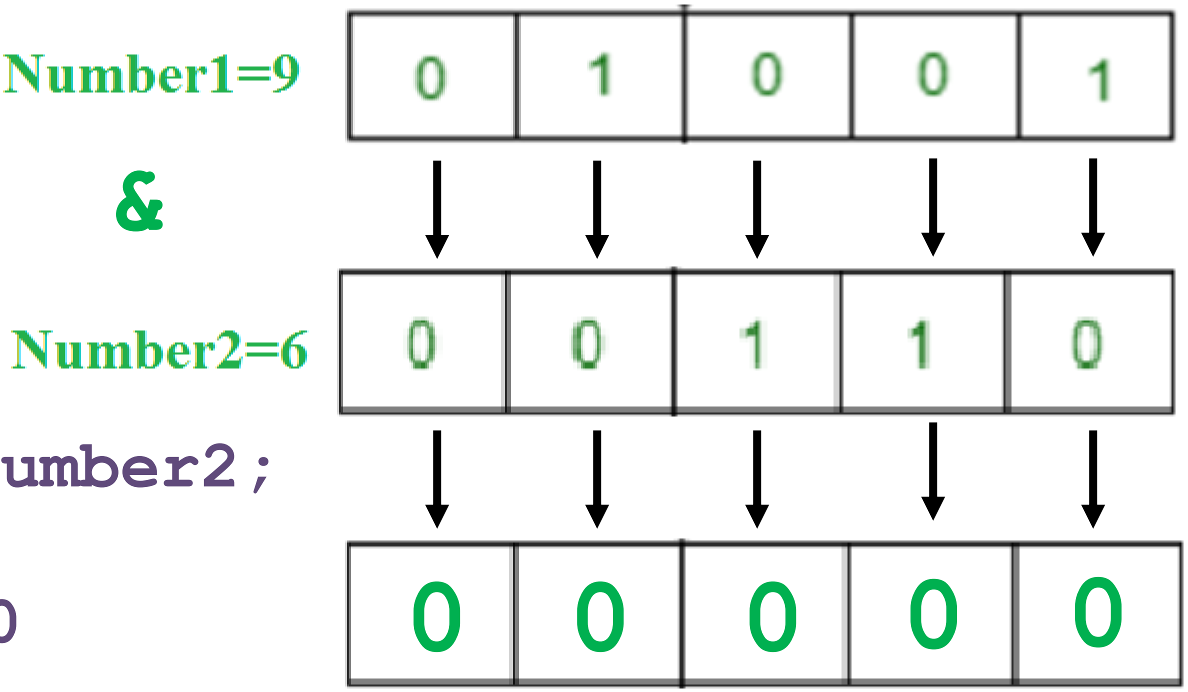


The following list describes these operations:

& and	| or
^ xor	~ not
>> left shift	<< right shift



AND (&)



Number1 & Number2;

9 & 6 = 0

```
# include <iostream.h >
void main () {
 int N1, N2;
 cin >> N1 >> N2;
 cout << N1 & N2;
}
```

Let N1 = 7 , N2 = 3

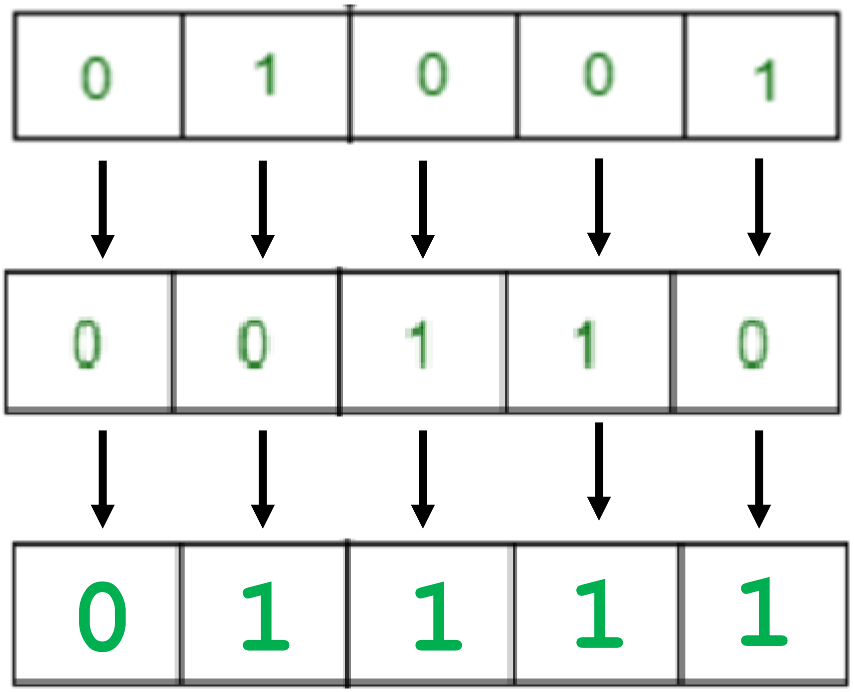
```
   0 1 1 1
&  0 0 1 1
  =======
   0 0 1 1
```

OR (|)

Number1=9

|

Number2=6



Number1 | Number2;

9 | 6 = 15

```
# include <iostream.h >
void main () {
 int N1, N2;
 cin >> N1 >> N2;
 cout << N1 | N2;
}
```

Let N1 = 7 , N2 = 3

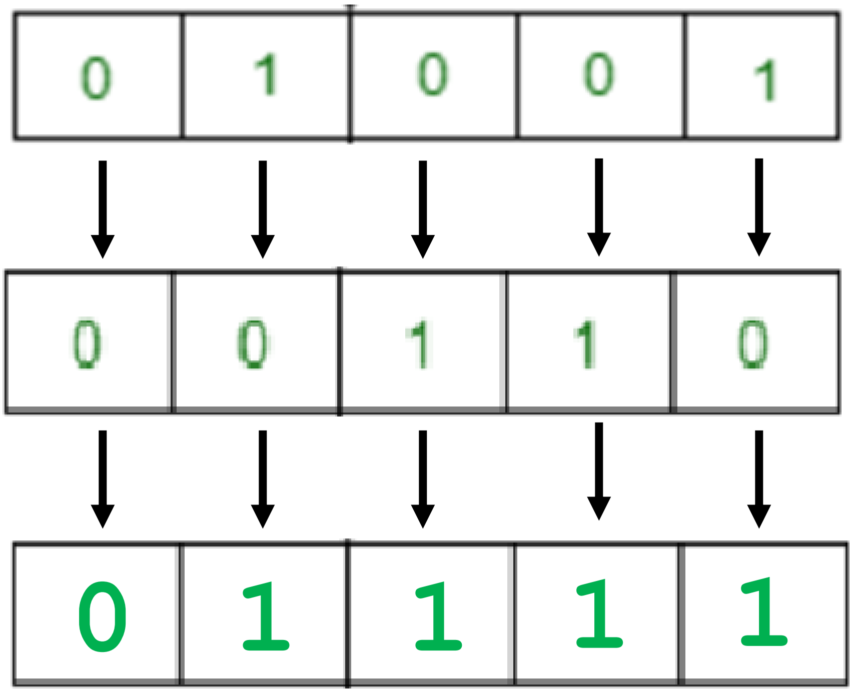
```
   0 1 1 1
|  0 0 1 1
   =======
   0 1 1 1
```

XOR(^)

Number1=9

^

Number2=6



Number1 ^ Number2;

9 ^ 6 = 15

```
# include <iostream.h >
void main () {
 int N1, N2;
 cin >> N1 >> N2;
 cout << N1 ^ N2;
}
```

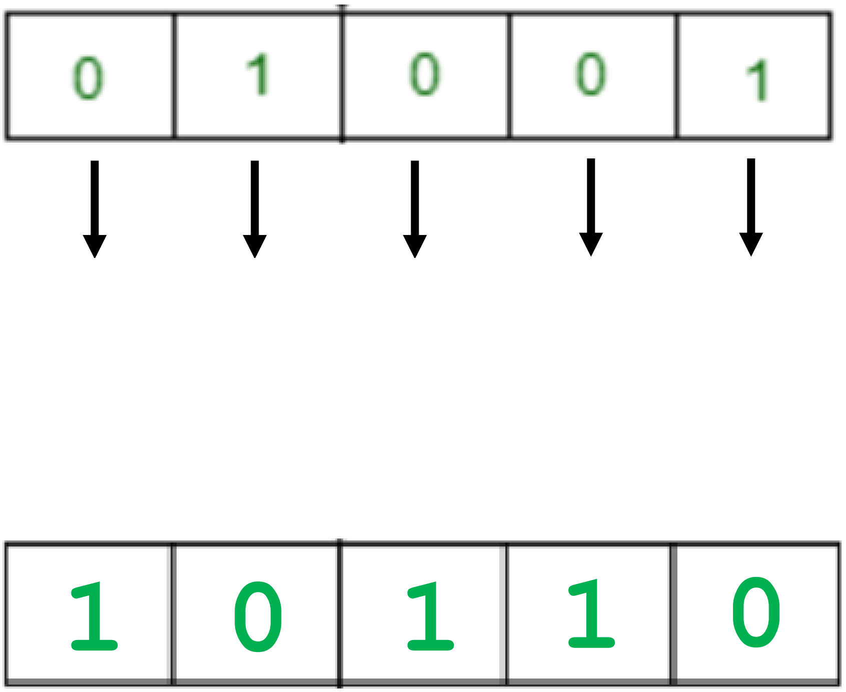
Let N1 = 7 , N2 = 3

```
    0 1 1 1
^   0 0 1 1
    =======
    0 1 0 0
```

NOT (~)

Number1=9

~



~Number1

~9 = 22

```
# include <iostream.h >
void main () {
 int N1, R;
 cin >> N1;
 R = ~ N1 ;
 cout << R;
}
```

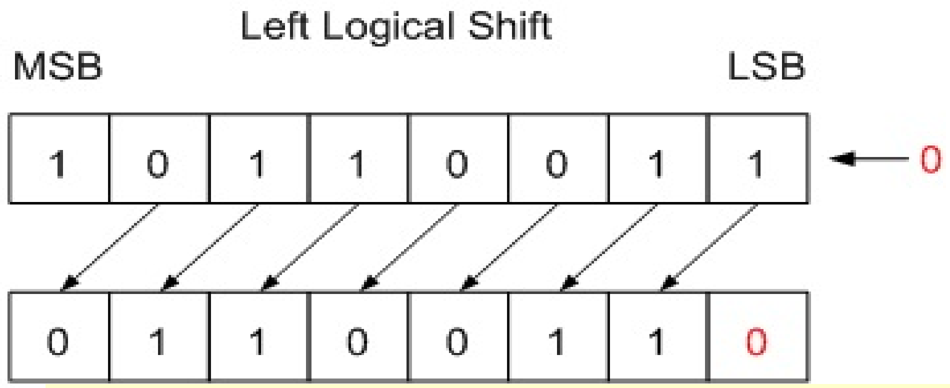
Let N1 = 7

```
~ 0 1 1 1
  =======
  1 0 0 0
```

Left Shift and Right Shift Operators in C++

Left Shift

A Left Logical Shift of one position moves each bit to the left by one. The vacant least significant bit (LSB) is filled with zero and the most significant bit (MSB) is discarded.



R = NUMBER << 1;

```
# include <iostream.h >
void main () {
 int N1, R;
 cin >> N1;
 R = N1 << 3;
 cout << R;
}
```

25 << 3

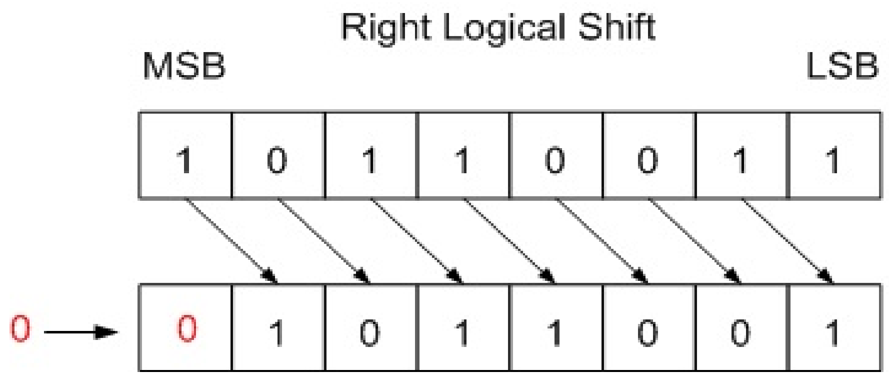
```
25 => 1 0 1 0 1
 <<3
      ============
 1 0 1 0 1 0 0 0
```

168

Left Shift and Right Shift Operators in C++

Right Shift

A Right Logical Shift of one position moves each bit to the right by one. The least significant bit is discarded and the vacant MSB is filled with zero.

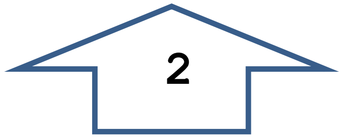


R = NUMBER >> 1;

```
# include <iostream.h >
void main () {
 int N1, R;
 cin >> N1;
 R = N1 >> 3;
 cout << R;
}
```

25 >> 3

```
25 => 1 0 1 0 1
 >>3
     ============
      0 0 0 1 0 1 0 1
```



Multiplication by left shift:

The result of a Left Shift operation is a multiplication by 2^n , where n is the number of shifted bit positions.

Example:

Let's take the decimal number 2 represented as 8 bit binary number *00000010*. By shifting in to the left with one position we get *00000100* which is 4 in decimal representation. If we shift it once more we get binary value *00001000* which is 8 in decimal representation.

2	0	0	0	0	0	0	1	0
2 << 1 = 4	0	0	0	0	0	1	0	0
4 << 1 = 8	0	0	0	0	1	0	0	0

$2 \ll 1 \rightarrow 2 \times 2$ $4 \ll 1 \rightarrow 4 \times 2$

Division by right shift:

The result of a Right Shift operation is a division by 2n , where n is the number of shifted bit positions.

Example:

If we have the binary number 01110101 (117 decimal) and we perform arithmetic right shift by 1 bit we get the binary number 00111010 (58 decimal). So we have divided the original number by 2.

117	0	1	1	1	0	1	0	1
117 >> 1 = 58	0	0	1	1	1	0	1	0
58 >> 1 = 29	0	0	0	1	1	1	0	1

117 >> 1 ➔ 117 / 2

58 >> 1 ➔ 58 / 2

```
#include <iostream.h>
void main() {
       // a = 5(00000101), b = 9(00001001)
       int a = 5, b = 9;

       cout<<"a = " << a <<","<< " b = " << b;
       cout << "a & b = " << (a & b);
       // The result is 00000001

       cout << "a | b = " << (a | b);
       // The result is 00001101

       cout << "a ^ b = " << (a ^ b);
       // The result is 00001100

       cout << "~(" << a << ") = " << (~a);
       // The result is 11111010

       cout<<"b << 1" <<" = "<< (b << 1);
       // The result is 00010010

       cout<<"b >> 1 "<<"= " << (b >> 1 );
       // The result is 00000100      }
```

C++ Programming

Ninevah University
College of Electronics Engineering
Department of Electronic Engineering
MEDICAL INSTRUMENTATION

2nd Year
2024 – 2025
8:30-10:30

Lecturer
Prof Dr. Qais Thanon
Lecture #3

All the lectures of this course will upload at the
Google classroom

Mathematical Functions

Mathematical calculations can be done in C++ programming language using the mathematical functions which are included in **math.h** library.
Let's learn each of them one by one :-

sin, cos, tan

Calling syntax

```
double x = sin(ang);
```

```
#include <iostream.h>
#include <math.h>
void main(){
double x =          y;
y = tan(x);
cout << y;
}
```

```
#include <iostream.h>
#include <math.h>
void main(){
double x = 45.3, y;
y = tan(x * M_PI/180.0);
cout << y;
}
```

The angle should be in RAD

Mathematical Functions

Power

The pow function is used to calculate the power of the base raised to the power of exponent.

Calling syntax

```
double y = pow(a, n);
```

$\rightarrow$ $y = a^n$

```
#include <iostream.h>
#include <math.h>
void main(){
double x = 2.8, y;
y = pow(x, 5);
cout << y;
}
```

$\rightarrow$ $y = 2.8^5$

y = 172.10368

$y = a^{n^m}$

$y = 2.8^{5^7}$

Mathematical Functions

Sqrt (square root)

sqrt function in C++ returns the square root of the double integer inside the parameter list.

Calling syntax `double y = sqrt(x);` $\Rightarrow$ $y = \sqrt{x}$

Log The logarithm function is used to find the natural log of the given number.

Calling syntax `double x = log(n);` `x = log(n)`

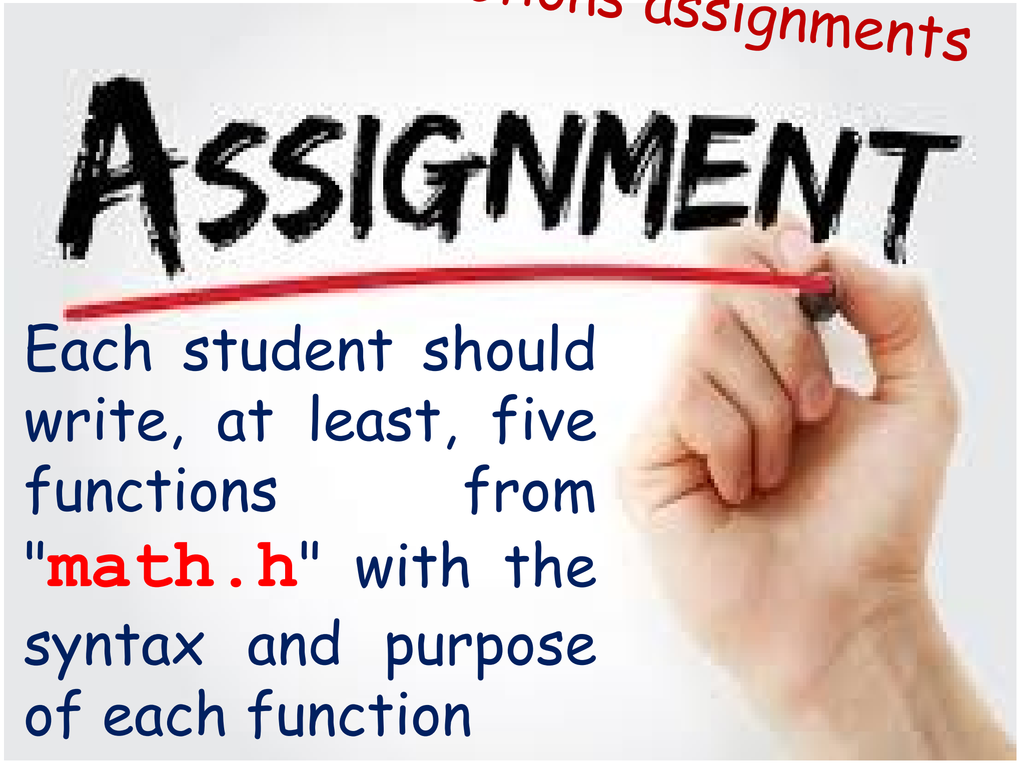
exp The exponential function is used to returns the (Euler's number) e (or 2.71828) raised to the given argument.

Calling syntax `double x = exp(n);` `x = e(n)`

abs The abs function returns the absolute value of the integer value.

Calling syntax `int x = abs(n);` `x =` n

Mathematical Functions assignments



Each student should write, at least, five functions from "**math.h**" with the syntax and purpose of each function

Conditional Statements

if **Selection statements:**

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

Relational operators

==

!=

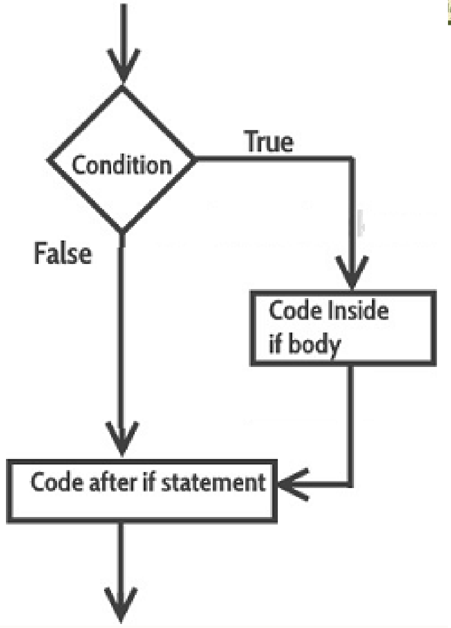
>

>=

<

<=

One-Way (if) Selection



```
if (A == 5)

if (A != B)

if (A > B)

if (A >= B)

if (A < 50)

if (A <= 50)
```

if ~~A > B~~

Ex: The following code fragment prints x is 100 only if the value stored in the x variable is indeed 100:

```
if (x == 100)
cout << "x is 100";
```

If we want more than a single statement to be executed in case that the condition is true we can specify a block using braces { }:

```
if (x == 100)
{
cout << "x is ";
cout << x;
}
```

If there are more than one relational operators logical operators should used.

Ex: Write a C++ program to enter two Boolean numbers then, print phrase "A and B“ if A and B equal to 1, or print phrase "A Or B" if A equal to 1 and B equal to 0.

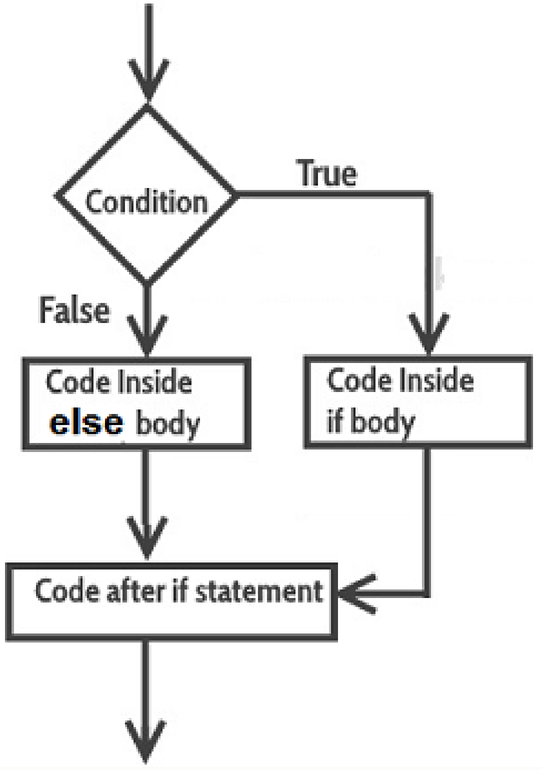
```
#include <iostream.h>
void main () {
int A,B;
cin >>A ;
cin >>B ;
if ((A==1)&&(B==1))
{cout << "A And B"<<'\n';}
if ((A==1)||(B==0))
{cout << "A or B"<<'\n';}

}
```

&&

if-else Selection statements:

```
if (expression)
     statement1;
else
     statement2;
```

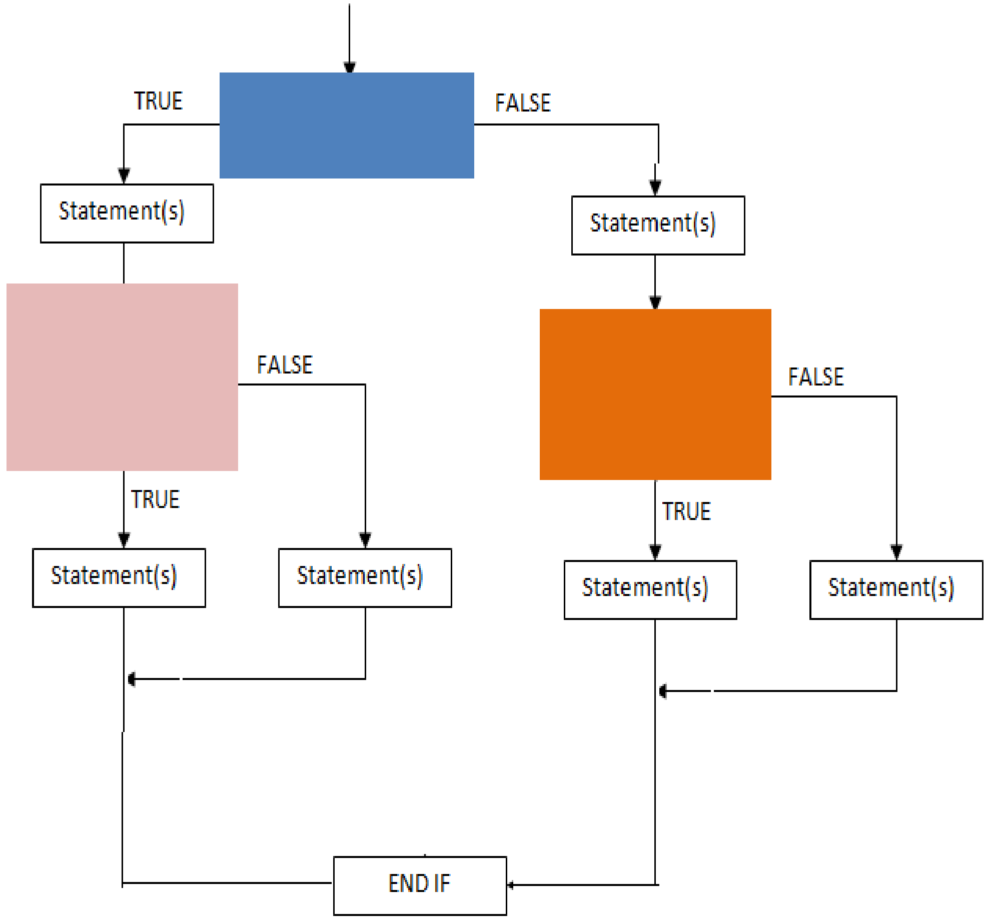
Two-Way (if…else) Selection



Ex: The following code fragment prints x is 100 only if the value stored in the x variable is indeed 100 , but if it has not –and only if not- it prints out x is not 100.

```
if (x == 100)
cout << "x is 100";
else
cout << "x is not 100";
```

Nested `if-else` **Selection statements:**



}

C++ Programming

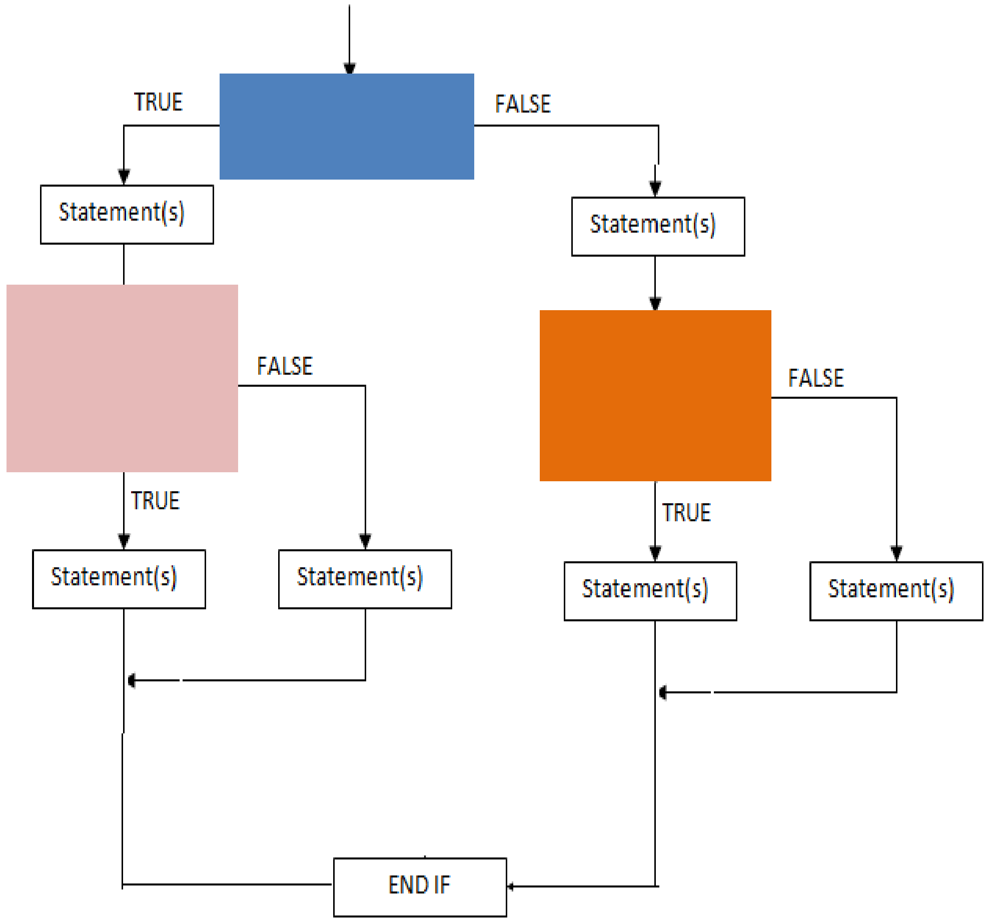
Ninevah University
College of Electronics Engineering
Department of Electronic Engineering
MEDICAL INSTRUMENTATION

2nd Year
2024 – 2025

Lecturer
Prof Dr. Qais Thanon
Lecture #4

All the lectures of this course will upload at the
Google classroom

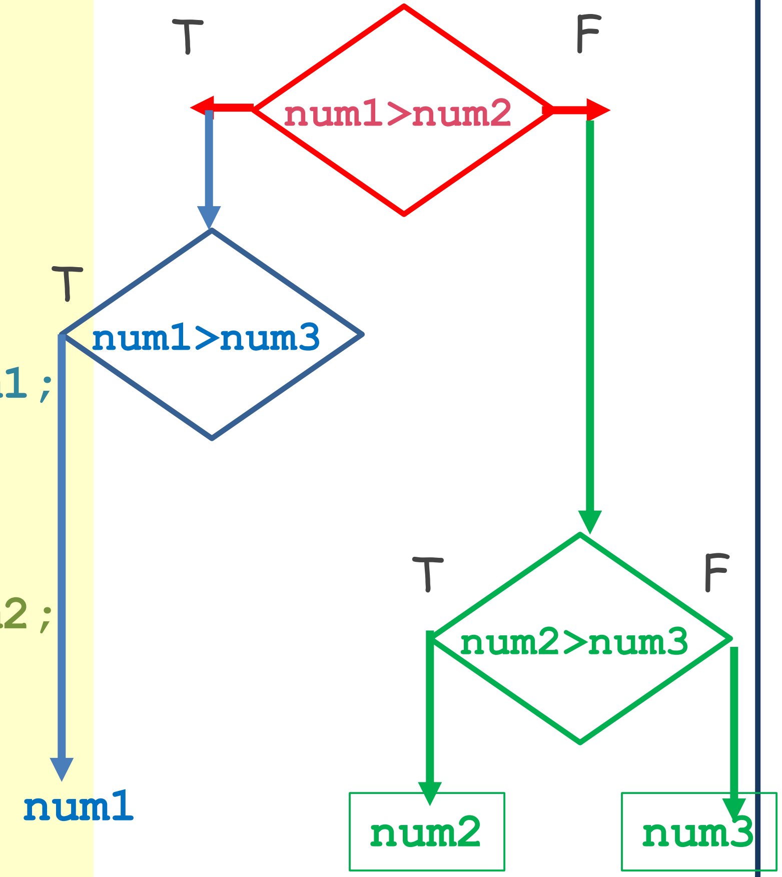
Nested `if-else` Selection statements:



}

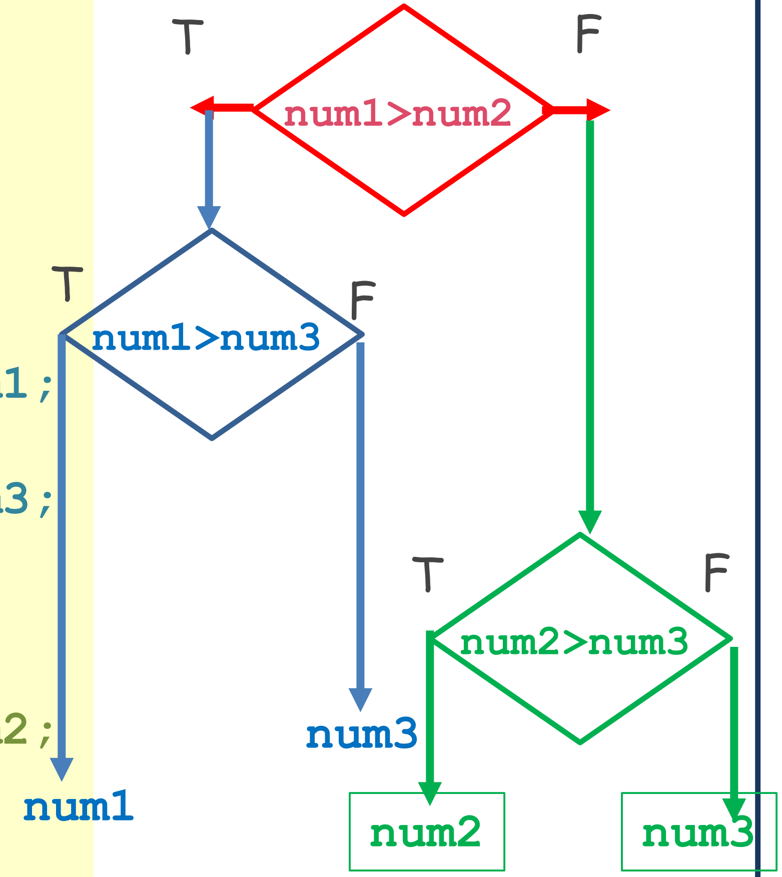
Write a C++ program to find the maximum number between three numbers

```
#include <iostream.h>
void  main(){
int num1, num2, num3;
cout<< "Enter three numbers:"<<"\n ;"
cin >> num1>> num2>> num3;
if (num1 > num2){
   if ( num1 > mun3 )
    cout << "The largest no is:"<< num1;
}
else {
 if (num2 > num3)
   cout << "The largest no is:" << num2;
 else
 cout << "The largest no is:" << num3;
  }
}
```



Write a C++ program to find the maximum number between three numbers

```
#include <iostream.h>
void  main(){
int num1, num2, num3;
cout<< "Enter three numbers:"<<"\n ;"
cin >> num1>> num2>> num3;
if (num1 > num2){
   if ( num1 > mun3 )
    cout << "The largest no is:"<< num1;
   else
    cout << "The largest no is:"<< num3;
}
else {
 if (num2 > num3)
   cout << "The largest no is:" << num2;
 else
 cout << "The largest no is:" << num3;
 }
}
```



Write C++ program to enter a number represents a centigrade degree. Find degree in Fahrenheit that generated from the first degree according to the relation:

F= (9/5) * C +32.

Then display the below phrases according to their equivalent Fahrenheit degree:

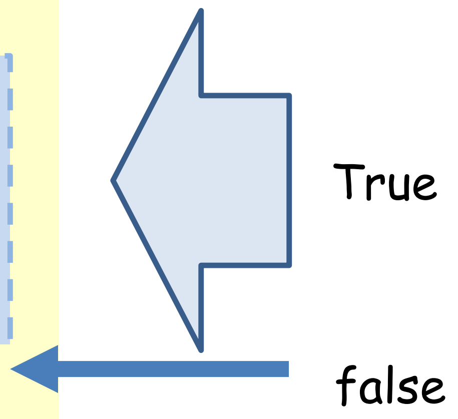
```
#include <iostream.h>
void main () {
float C,F;
cin >> C;
F = (9 / 5) * C + 32;
cout << "F="<<F<<'\n';
if (F <= 41)
{cout << "Cold"<<'\n';}
else if (F > 41 && F <= 77)
{cout << " Nice"<<'\n';}
     else {cout << Hot"<<'\n';}
}
```

1. “Cold” when F ≤ 41.
2. “Nice” when 41< F ≤ 77.
3. “Hot” when F >77.

Ex: Write a C++ program to find the value of Z where:

$$Z = \begin{cases} x + y & \text{if} \quad i = 1 \\ x - y & \text{if} \quad i = 2 \\ x * y & \text{if} \quad i = 3 \\ x / y & \text{if} \quad i = 4 \end{cases}$$

```
#include <iostream.h>
void main () {
int i;
float x, y, Z;
cin >> i >> x >> y;

if ( i >= 1 && i <= 4){




else cout <<“wronh no.”;
}
```



Write a C++ program to input student name and marks of three subjects ,and calculate average, and print grade according to the following conditions:

Grade A	If average >= 90
Grade B	If average >= 80
Grade C	If average >= 70
Grade D	If average >= 60
Grade E	If average >= 50
Grade F	If average < 50

```
#include <iostream.h>
#include <string>
void main() {
  int d1, d2, d3, sum = 0;
  float avg
  string name;
  cin >> name >> d1 >> d2 >> d3;
  avg = (d1 + d2 + d3) / 3.0;
    if ( average >= 90 ) cout << " Grade A ";
    else if (average >= 80)
         cout << " Grade B ";
         else if (average >= 70)
                  cout << " Grade C ";
              else if (average >= 60)
                   cout << " Grade D ";
                   else if (average >= 50)
                         cout << " Grade E ";

}}
```

Iteration loops in C++

There may be a situation, when you need to execute a block of code several number of times. For example if you want to print the numbers from 1 to 10 then the program would be:

```
#include <iostream.h>
void main(){
int x=1;
cout << x++; // the out put would be 1
cout << x++; // the out put would be 2
cout << x++; // the out put would be 3
cout << x++; // the out put would be 4
cout << x++; // the out put would be 5
cout << x++; // the out put would be 6
cout << x++; // the out put would be 7
cout << x++; // the out put would be 8
cout << x++; // the out put would be 9
cout << x++; // the out put would be 10
}
```

Initial value

Update

Condition to stop

goto statement

The goto statement is a jump statement which is sometimes also referred to as unconditional jump statement. The goto statement can be used to jump from anywhere to anywhere within a function.

Syntax:

```
goto label;
Statement 1;
...
Statement n;
label:
```

Flag

```
Label:
Statement 1;
…..
Statement n;
goto label;
```

```
// C++ program to print numbers
from 1 to 10

#include <iostream.h>
void main() {
int n = 0;
label:
cout << n++ << " ";
if (n <= 10)
goto label;
}
```

initial value

Update

condition

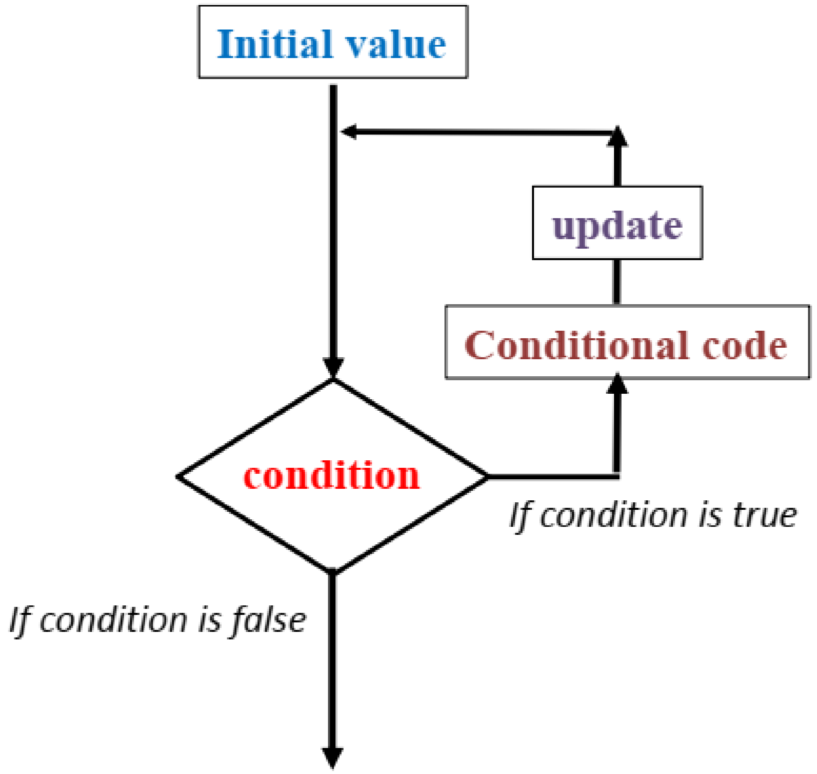
Iteration loops in C++

C++ programming language provides the following type of loops to handle looping requirements.

Sr. No.	Loop Type & Description
1	for loop Execute a sequence of statements multiple times and abbreviates the code that manages the loop variable.
2	while loop Repeats a statement or group of statements while a given condition is true. It tests the condition before executing the loop body.
3	do...while loop Like a 'while' statement, except that it tests the condition at the end of the loop body.
4	nested loops You can use one or more loop inside any another 'while', 'for' or 'do..while' loop.

A loop statement allows us to execute a statement or group of statements multiple times and following is the general from of a loop statement in most of the programming languages -

The initial statement, loop condition, and update statement are called: loop control statements

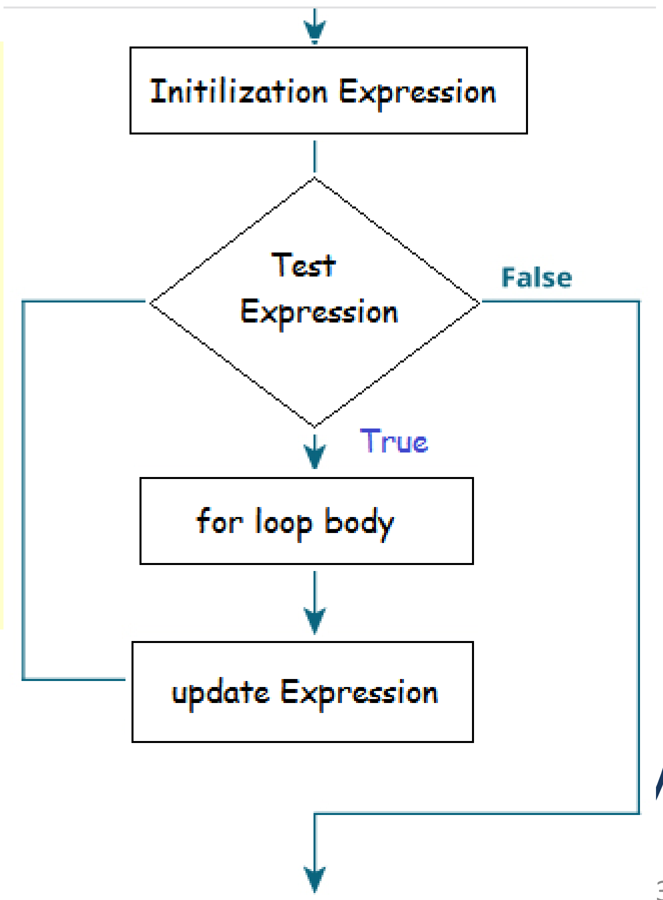


The for Loop

The general form of the for statement is:

```
for (initial statement; loop condition; update statement){
loop statements;
}
```

```
// C++ program to print numbers from 1 to 10
#include <iostream.h>
void main() {
int n;
for ( n = 1; n <= 10; n++)
cout << n << " ";
}
```



The `for` Loop

The `for` loop executes as follows:

1. The initial statement executes.
2. The loop condition is evaluated. If the loop condition evaluates to true
    i. Execute the `for` loop statement.
    ii. Execute the update statement (the third expression in the parentheses).
3. Repeat Step 2 until the loop condition evaluates to false.

The initial statement usually initializes a variable (called the for **loop control**, or for **indexed**, **variable**).

In C++, for is a reserved word

The `for` Loop (comments)

The following are some comments on `for` loops:

✓ If the loop condition is initially `false,` the loop body does not execute.

✓ The update expression, when executed, changes the value of the loop control variable (initialized by the initial expression), which eventually sets the value of the loop condition to false. The `for` loop body executes indefinitely if the loop condition is always `true.`

✓ C++ allows you to use fractional values for loop control variables of the `double` type (or any real data type). Because different computers can give these loop control variables different results, you should avoid using such variables.

The for Loop (comments)

- ✓ A semicolon at the end of the for statement (just before the body of the loop) is a semantic error. In this case, the action of the for loop is empty.

```
for (int x=0; x<100; x++);
```

- ✓ In the for statement, if the loop condition is omitted, it is assumed to be true

```
    for (int x=0; ; x++)
```

- ✓ In a for statement, you can omit all three statements—initial statement, loop condition, and update statement. The following is a legal for loop:

```
for ( ; ; )
cout << "Hello ";
```

Example: Assume the following specification:

Input: read a number N > 0

Output: write the sequence 1 2 3 … N (one number per

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

N = 6

```
1
2
3
4
5
6
```

Assume the following specification:

Input: read a number N < 0

Output: write the sequence -1 -2 -3 … -N (one number per line)

```
#include <iostream.h>
void main() {
int N;
cin >> N;
for ( int i = -1; i ● N; i--)
cout << i << "\n ";}
```

N = -6

```
-1
-2
-3
-4
-5
-6
```

Example: Program to find the factorial of an integer number

n! = 1 × 2 × 3 × 4 × ……. × n

```
#include <iostream.h>
#include <conio.h>
void main(){
clrscr();
int num, i, fac=1;
cout <<“Enter the number”;
cin >> num;
for (i=1; i<=num; i++)
fac*=i;
cout <<“\n”<<“The factorial is:”<< fac;

getch();
}
```

Example: Write a C++ program to count the number of digits of any integer.

202211101.

```cpp
#include <iostream.h>
#include <conio.h>
void main(){
clrscr();
long N;
int ndigits = 0;
cin >> N;
for ( ; N > 9; ) {
ndigits++;
N = N/10; // extracts one digit
}
cout << ndigits + 1;
getch();
}
```

C++ Programming

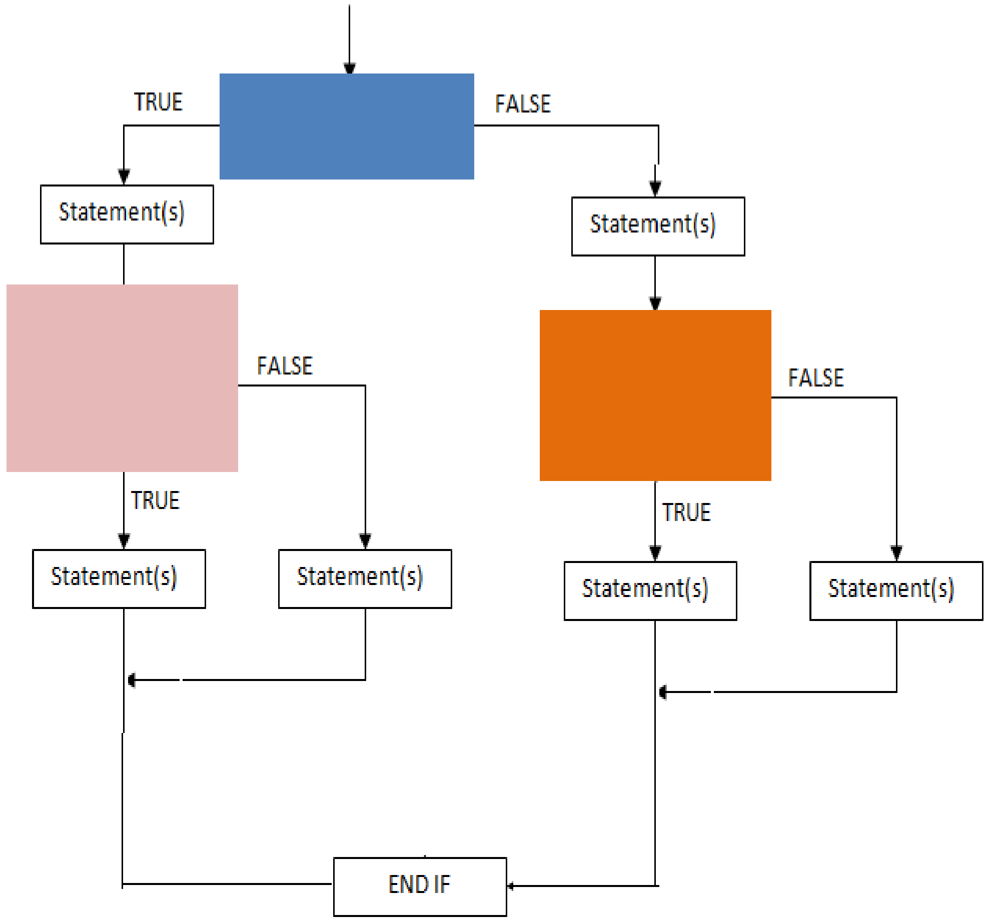
Ninevah University
College of Electronics Engineering
Department of Electronic Engineering
MEDICAL INSTRUMENTATION

2nd Year
2024 – 2025

Lecturer
Prof Dr. Qais Thanon
Lecture #4

All the lectures of this course will upload at the Google classroom

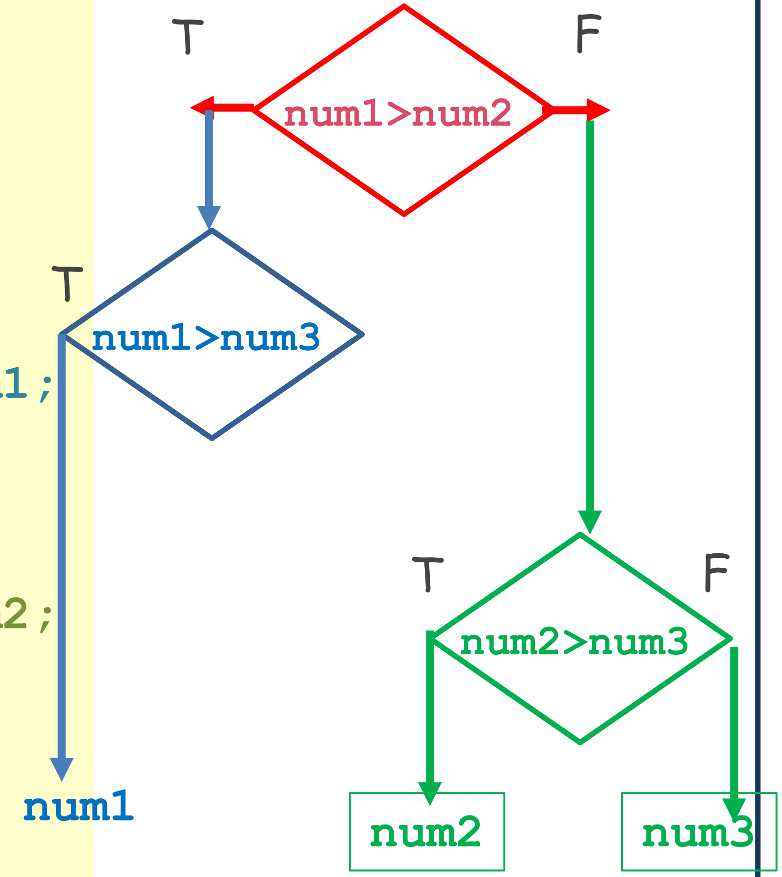
Nested `if-else` **Selection statements:**



}

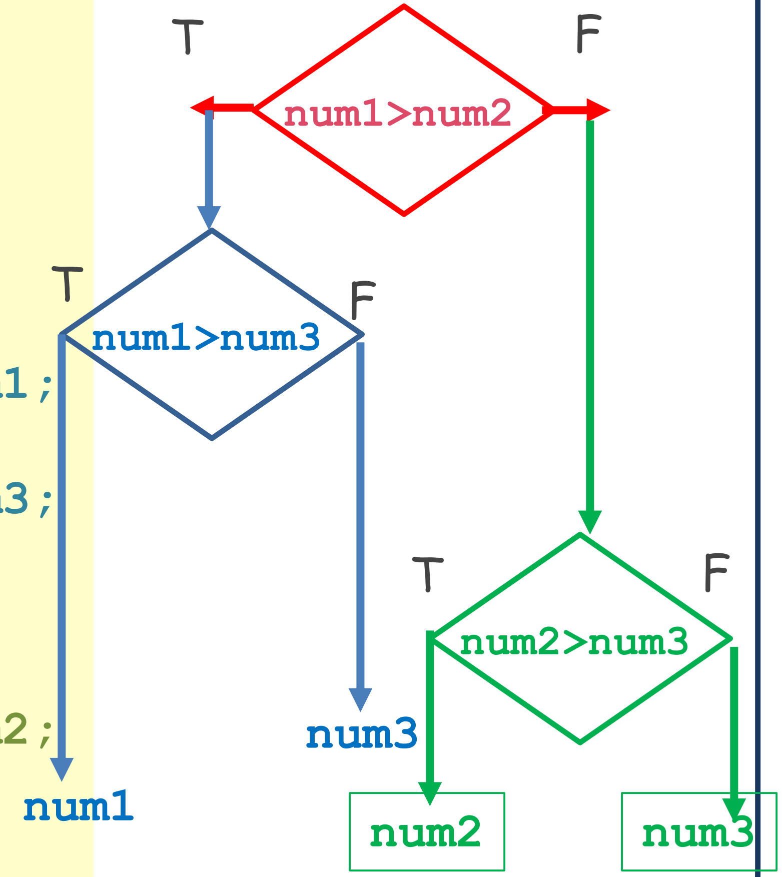
Write a C++ program to find the maximum number between three numbers

```
#include <iostream.h>
void  main(){
int num1, num2, num3;
cout<< "Enter three numbers:"<<"\n ;"
cin >> num1>> num2>> num3;
if (num1 > num2){
   if ( num1 > mun3 )
    cout << "The largest no is:"<< num1;
}
else {
 if (num2 > num3)
   cout << "The largest no is:" << num2;
 else
 cout << "The largest no is:" << num3;
  }
}
```



Write a C++ program to find the maximum number between three numbers

```
#include <iostream.h>
void  main(){
int num1, num2, num3;
cout<< "Enter three numbers:"<<"\n ;"
cin >> num1>> num2>> num3;
if (num1 > num2){
   if ( num1 > mun3 )
    cout << "The largest no is:"<< num1;
   else
    cout << "The largest no is:"<< num3;
}
else {
 if (num2 > num3)
   cout << "The largest no is:" << num2;
 else
 cout << "The largest no is:" << num3;
 }
}
```



Write C++ program to enter a number represents a centigrade degree. Find degree in Fahrenheit that generated from the first degree according to the relation:

F= (9/5) * C +32.

Then display the below phrases according to their equivalent Fahrenheit degree:

```
#include <iostream.h>
void main () {
float C,F;
cin >> C;
F = (9 / 5) * C + 32;
cout << "F="<<F<<'\n';
if (F <= 41)
{cout << "Cold"<<'\n';}
else if (F > 41 && F <= 77)
{cout << " Nice"<<'\n';}
     else {cout << Hot"<<'\n';}
}
```

1. “Cold” when F ≤ 41.
2. “Nice” when 41< F ≤ 77.
3. “Hot” when F >77.

Ex: Write a C++ program to find the value of Z where:

$$Z = \begin{cases} x + y & \text{if} \quad i = 1 \\ x - y & \text{if} \quad i = 2 \\ x * y & \text{if} \quad i = 3 \\ x / y & \text{if} \quad i = 4 \end{cases}$$

```
#include <iostream.h>
void main () {
int i;
float x, y, Z;
cin >> i >> x >> y;

if ( i >= 1 && i <= 4){




else cout <<"wronh no.";
}
```

True

false

Write a C++ program to input student name and marks of three subjects ,and calculate average, and print grade according to the following conditions:

Grade A	If average >= 90
Grade B	If average >= 80
Grade C	If average >= 70
Grade D	If average >= 60
Grade E	If average >= 50
Grade F	If average < 50

```
#include <iostream.h>
#include <string>
void main() {
  int d1, d2, d3, sum = 0;
  float avg
  string name;
  cin >> name >> d1 >> d2 >> d3;
  avg = (d1 + d2 + d3) / 3.0;
    if ( average >= 90 ) cout << " Grade A ";
    else if (average >= 80)
         cout << " Grade B ";
         else if (average >= 70)
                  cout << " Grade C ";
              else if (average >= 60)
                   cout << " Grade D ";
                   else if (average >= 50)
                         cout << " Grade E ";


}}
```

Iteration loops in C++

There may be a situation, when you need to execute a block of code several number of times. For example if you want to print the numbers from 1 to 10 then the program would be:

```
#include <iostream.h>
void main(){
int x=1;
cout << x++; // the out put would be 1
cout << x++; // the out put would be 2
cout << x++; // the out put would be 3
cout << x++; // the out put would be 4
cout << x++; // the out put would be 5
cout << x++; // the out put would be 6
cout << x++; // the out put would be 7
cout << x++; // the out put would be 8
cout << x++; // the out put would be 9
cout << x++; // the out put would be 10
}
```

Initial value

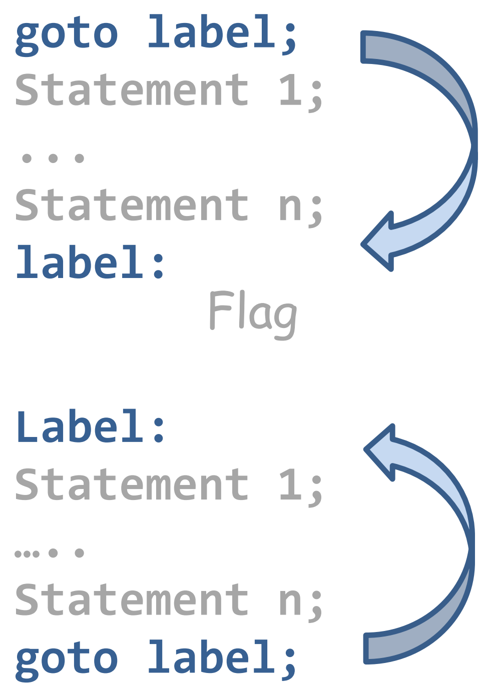
Update

Condition to stop

goto statement

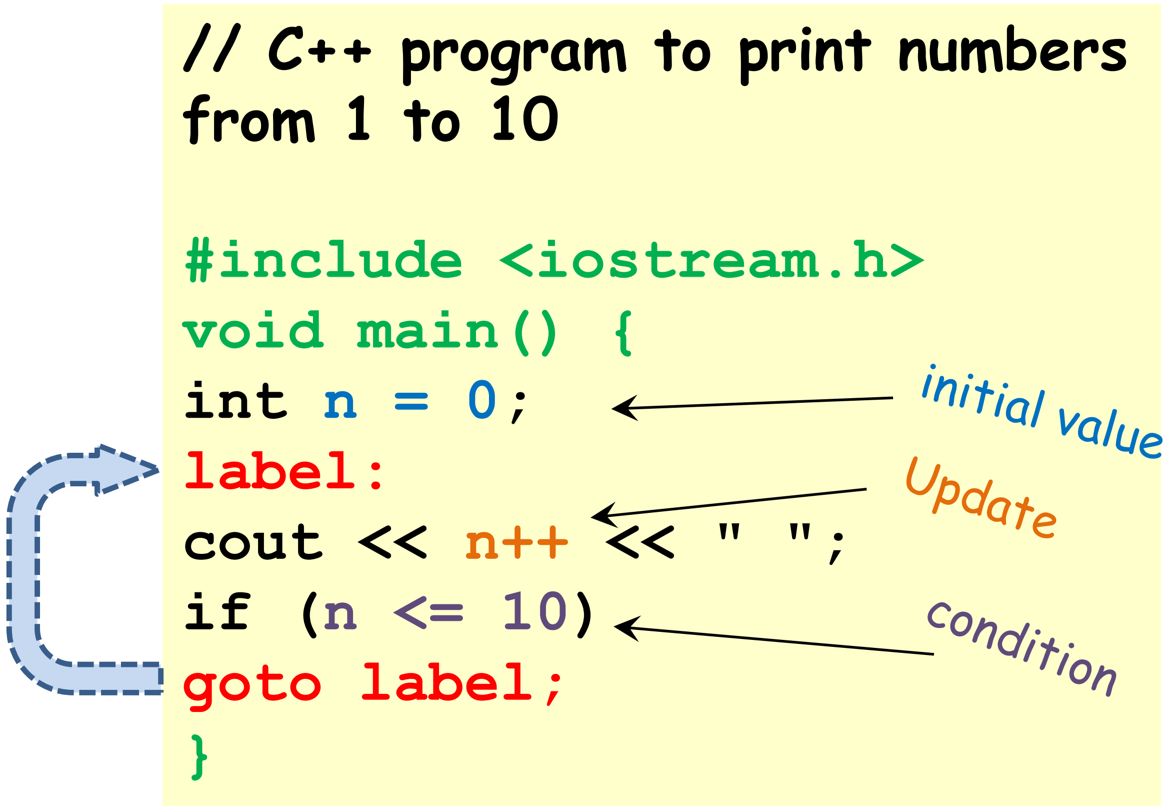
The goto statement is a jump statement which is sometimes also referred to as unconditional jump statement. The goto statement can be used to jump from anywhere to anywhere within a function.

Syntax:



```
goto label;
Statement 1;
...
Statement n;
label:
```

```
Label:
Statement 1;
…..
Statement n;
goto label;
```



```
// C++ program to print numbers
from 1 to 10

#include <iostream.h>
void main() {
int n = 0;
label:
cout << n++ << " ";
if (n <= 10)
goto label;
}
```

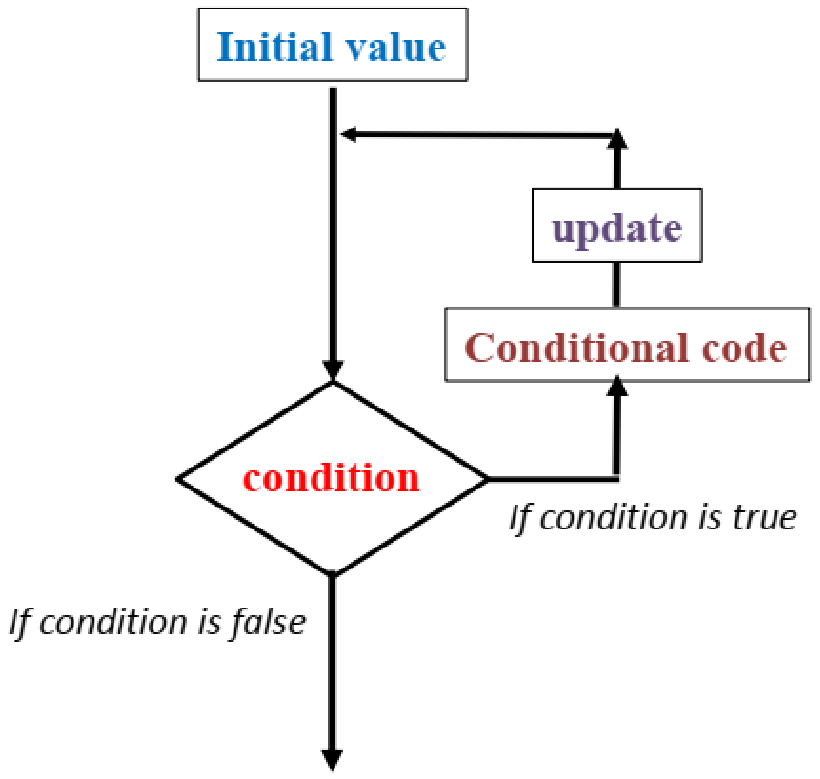
Iteration loops in C++

C++ programming language provides the following type of loops to handle looping requirements.

Sr. No.	Loop Type & Description
1	for loop Execute a sequence of statements multiple times and abbreviates the code that manages the loop variable.
2	while loop Repeats a statement or group of statements while a given condition is true. It tests the condition before executing the loop body.
3	do...while loop Like a 'while' statement, except that it tests the condition at the end of the loop body.
4	nested loops You can use one or more loop inside any another 'while', 'for' or 'do..while' loop.

A loop statement allows us to execute a statement or group of statements multiple times and following is the general from of a loop statement in most of the programming languages -

The initial statement, loop condition, and update statement are called: loop control statements

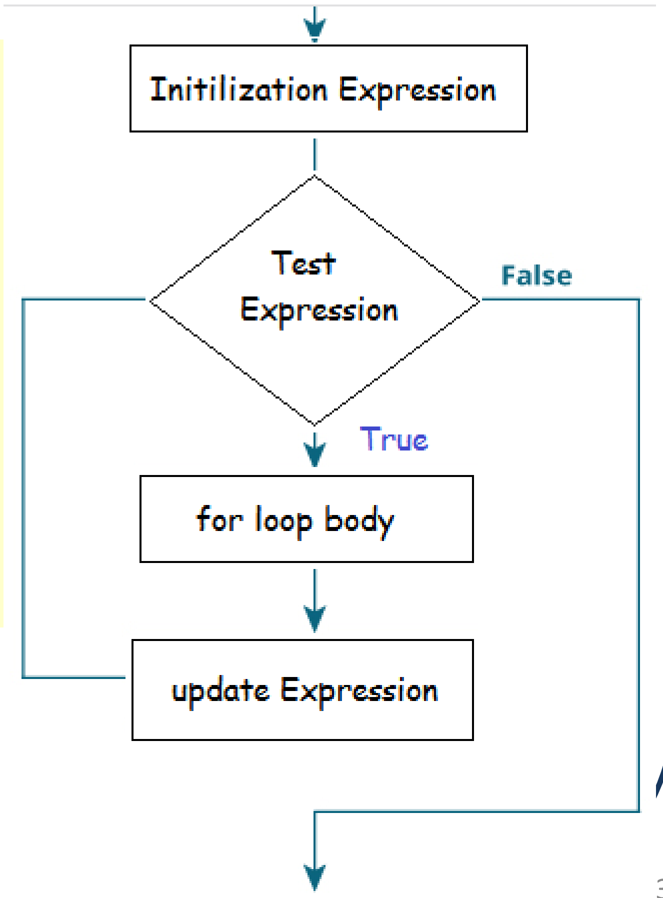


The for Loop

The general form of the for statement is:

```
for (initial statement; loop condition; update statement){
loop statements;
}
```

```
// C++ program to print numbers from 1 to 10
#include <iostream.h>
void main() {
int n;
for ( n = 1; n <= 10; n++)
cout << n << " ";
}
```



The `for` Loop

The `for` loop executes as follows:

1. The initial statement executes.
2. The loop condition is evaluated. If the loop condition evaluates to true
   - i. Execute the `for` loop statement.
   - ii. Execute the update statement (the third expression in the parentheses).
3. Repeat Step 2 until the loop condition evaluates to false.

The initial statement usually initializes a variable (called the for **loop control**, or for **indexed**, **variable**).

In C++, for is a reserved word

The `for` Loop (comments)

The following are some comments on `for` loops:

✓ If the loop condition is initially `false,` the loop body does not execute.

✓ The update expression, when executed, changes the value of the loop control variable (initialized by the initial expression), which eventually sets the value of the loop condition to false. The `for` loop body executes indefinitely if the loop condition is always `true.`

✓ C++ allows you to use fractional values for loop control variables of the `double` type (or any real data type). Because different computers can give these loop control variables different results, you should avoid using such variables.

The `for` Loop (comments)

- ✓ A semicolon at the end of the `for` statement (just before the body of the loop) is a semantic error. In this case, the action of the `for` loop is empty.

```
for (int x=0; x<100; x++);
```

- ✓ In the `for` statement, if the loop condition is omitted, it is assumed to be `true`

```
    for (int x=0; ; x++)
```

- ✓ In a `for` statement, you can omit all three statements—initial statement, loop condition, and update statement. The following is a legal `for` loop:

```
for ( ; ; )
cout << "Hello ";
```

Example: Assume the following specification:

Input: read a number N > 0

Output: write the sequence 1 2 3 … N (one number per

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

N = 6

```
1
2
3
4
5
6
```

Assume the following specification:

Input: read a number N < 0

Output: write the sequence -1 -2 -3 … -N (one number per line)

```
#include <iostream.h>
void main() {
int N;
cin >> N;
for ( int i = -1; i ● N; i--)
cout << i << "\n ";}
```

N = -6

```
-1
-2
-3
-4
-5
-6
```

Example: Program to find the factorial of an integer number

n! = 1 × 2 × 3 × 4 × ……. × n

```
#include <iostream.h>
#include <conio.h>
void main(){
clrscr();
int num, i, fac=1;
cout <<"Enter the number";
cin >> num;
for (i=1; i<=num; i++)
fac*=i;
cout <<"\n"<<"The factorial is:"<< fac;

getch();
}
```

Example: Write a C++ program to count the number of digits of any integer.

202211101.

```
#include <iostream.h>
#include <conio.h>
void main(){
clrscr();
long N;
int ndigits = 0;
cin >> N;
for ( ; N > 9; ) {
ndigits++;
N = N/10; // extracts one digit
}
cout << ndigits + 1;
getch();
}
```

C++ Programming

Ninevah University
College of Electronics Engineering
Department of Electronic Engineering
MEDICAL INSTRUMENTATION

2nd Year
2024 – 2025
8:30-10:30

Lecturer
Prof Dr. Qais Thanon
Lecture #3

All the lectures of this course will upload at the Google classroom

Mathematical Functions

Mathematical calculations can be done in C++ programming language using the mathematical functions which are included in **math.h** library.
Let's learn each of them one by one :-

sin, cos, tan

Calling syntax

```
double x = sin(ang);
```

```
#include <iostream.h>
#include <math.h>
void main(){
double x =          y;
y = tan(x);
cout << y;
}
```

```
#include <iostream.h>
#include <math.h>
void main(){
double x = 45.3, y;
y = tan(x * M_PI/180.0);
cout << y;
}
```

The angle should be in RAD

Mathematical Functions

Power

The pow function is used to calculate the power of the base raised to the power of exponent.

Calling syntax

```
double y = pow(a, n);
```

$\rightarrow$ $y = a^n$

```
#include <iostream.h>
#include <math.h>
void main(){
double x = 2.8, y;
y = pow(x, 5);
cout << y;
}
```

$\rightarrow$ $y = 2.8^5$

y = 172.10368

$y = a^{n^m}$

$y = 2.8^{5^7}$

Mathematical Functions

Sqrt (square root)

`sqrt` function in C++ returns the square root of the double integer inside the parameter list.

Calling syntax `double y = sqrt(x);` $\Rightarrow$ $y = \sqrt{x}$

`Log` The logarithm function is used to find the natural log of the given number.

Calling syntax `double x = log(n);` `x = log(n)`

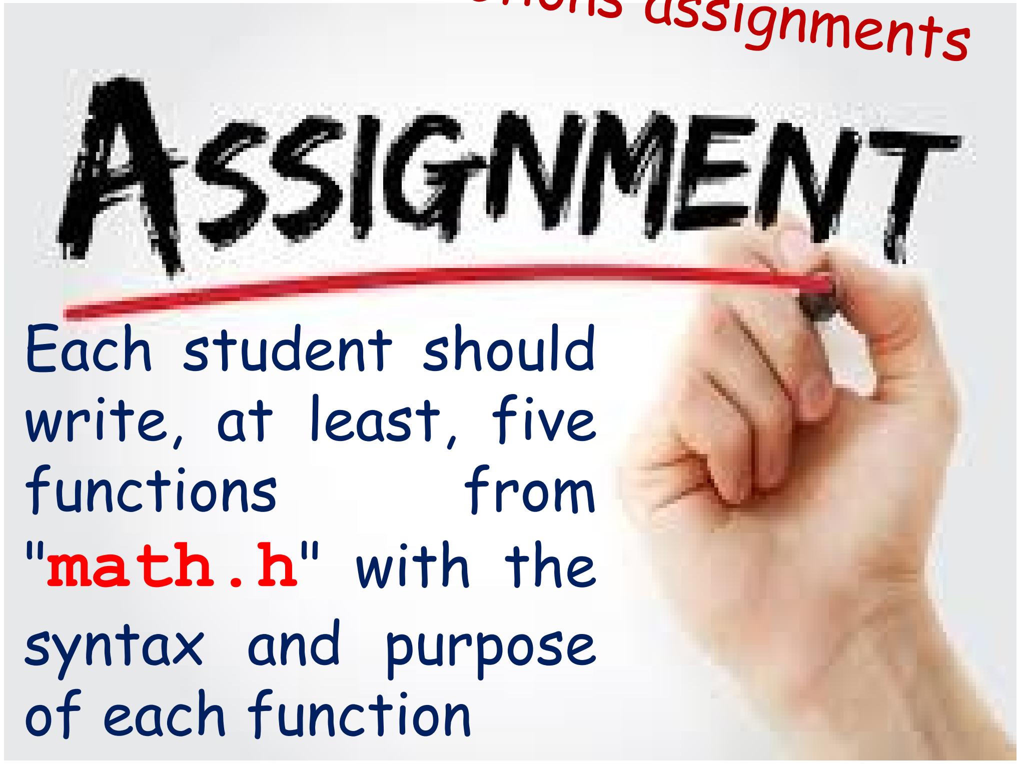
`exp` The exponential function is used to returns the (Euler's number) e (or 2.71828) raised to the given argument.

Calling syntax `double x = exp(n);` `x = e(n)`

`abs` The abs function returns the absolute value of the integer value.

Calling syntax `int x = abs(n);` `x =` n

Mathematical Functions assignments



Each student should write, at least, five functions from "**math.h**" with the syntax and purpose of each function

Conditional Statements

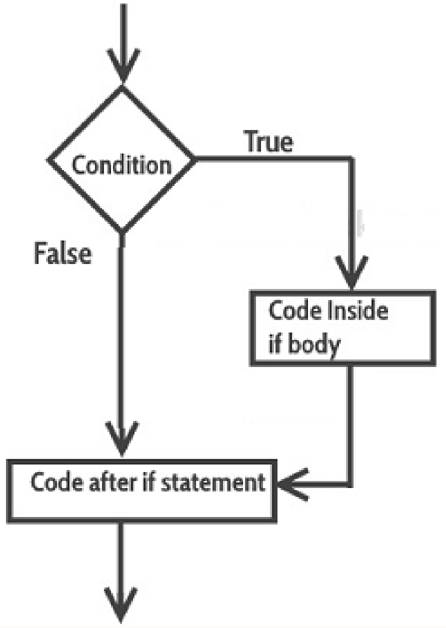
if **Selection statements:**

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

Relational operators

```
==
!=
>
>=
<
<=
```

One-Way (if) Selection



```
if (A == 5)
if (A != B)
if (A > B)
if (A >= B)
if (A < 50)
if (A <= 50)
```

if ~~A > B~~

Ex: The following code fragment prints x is 100 only if the value stored in the x variable is indeed 100:

```
if (x == 100)
cout << "x is 100";
```

If we want more than a single statement to be executed in case that the condition is true we can specify a block using braces { }:

```
if (x == 100)
{
cout << "x is ";
cout << x;
}
```

If there are more than one relational operators logical operators should used.

Ex: Write a C++ program to enter two Boolean numbers then, print phrase "A and B" if A and B equal to 1, or print phrase "A Or B" if A equal to 1 and B equal to 0.

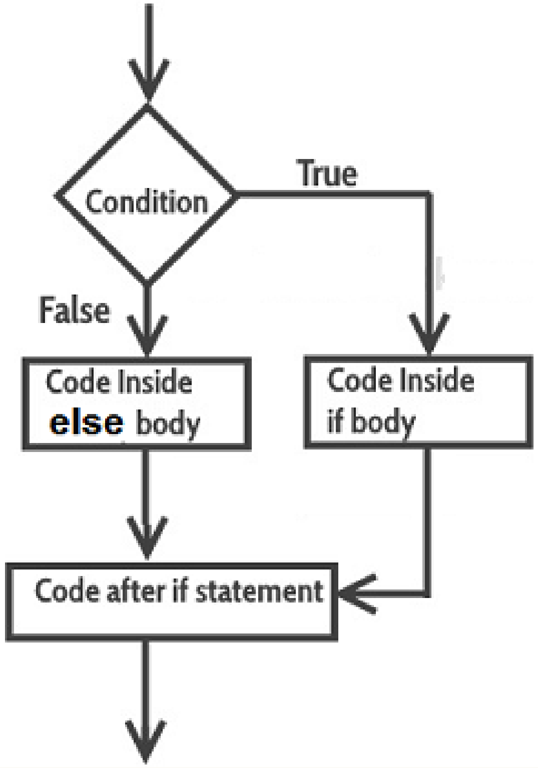
```
#include <iostream.h>
void main () {
int A,B;
cin >>A ;
cin >>B ;
if ((A==1)&&(B==1))
{cout << "A And B"<<'\n';}
if ((A==1)||(B==0))
{cout << "A or B"<<'\n';}

}
```

&&

if-else Selection statements:

Two-Way (if…else) Selection

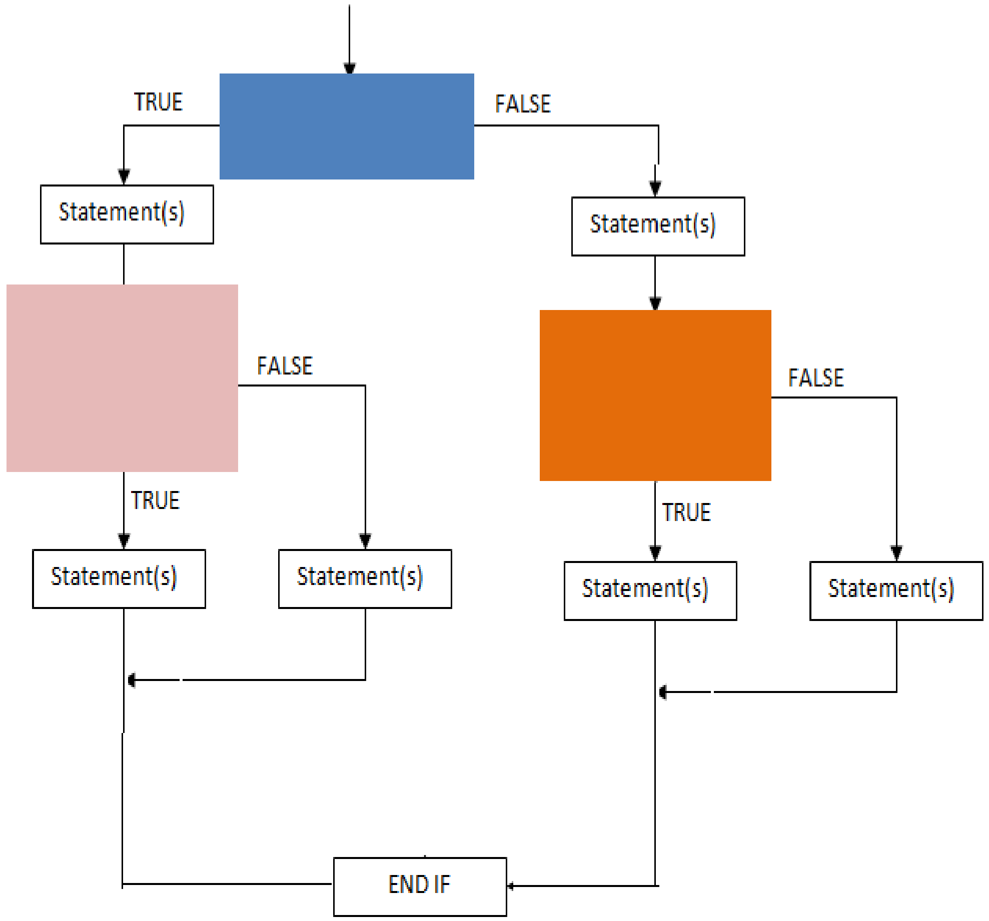
```
if (expression)
     statement1;
else
     statement2;
```



Ex: The following code fragment prints x is 100 only if the value stored in the x variable is indeed 100 , but if it has not -and only if not- it prints out x is not 100.

```
if (x == 100)
cout << "x is 100";
else
cout << "x is not 100";
```

Nested `if-else` **Selection statements:**



}

C++ Programming

Ninevah University
College of Electronics Engineering
Department of Electronic Engineering
MEDICAL INSTRUMENTATION

2nd Year
2024 – 2025
10:30 – 12:30

Lecturer
Prof Dr. Qais Thanon
Lecture #2

All the lectures of this course will upload at the Google classroom

The cout Object:

Use the cout<< object to display information on the computer's screen.

- Its job is to output information using the standard output device.
- The << operator is used to send the string like "NINEVAH UNIVERSITY" to cout.
- cout does not produce a newline at the end of a statement

```
# include <iostream.h >
void main ()  {
 cout << " *** University of NINEVAH ***";
}
```

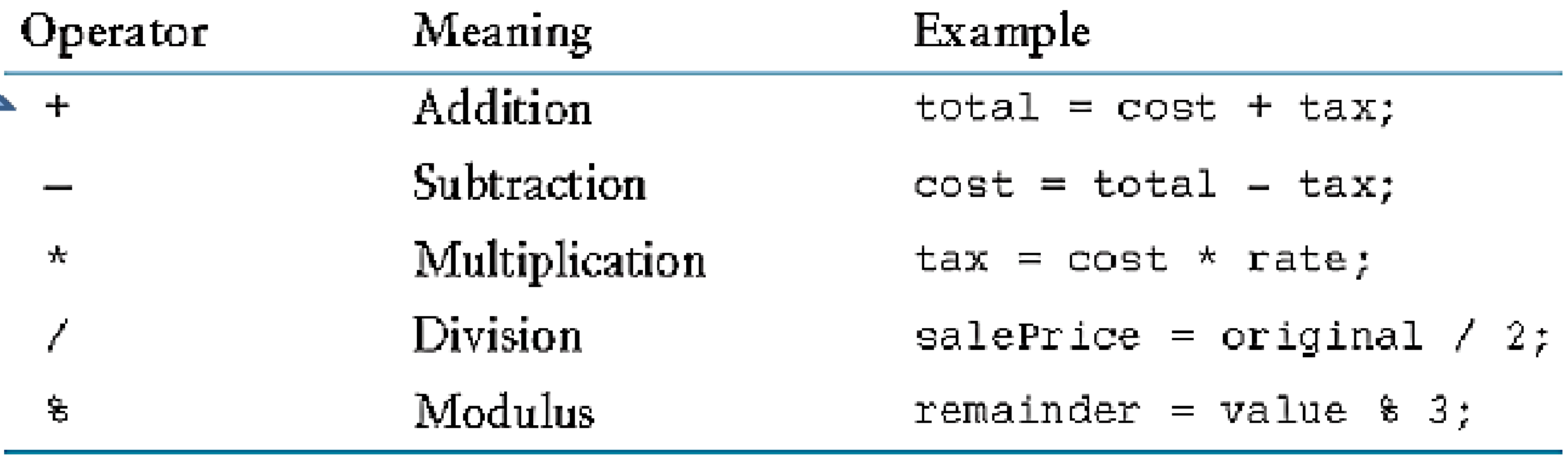
```
*** University of NINEVAH ***
```

The cin Object

- The cin>> object reads information types at the keyboard.
- Notice the >> and << operators appear to point in the direction information is flowing.

Arithmetic Operators

•There are many operators for manipulating numeric values and performing arithmetic operations



Operator	Meaning	Example
+	Addition	`total = cost + tax;`
–	Subtraction	`cost = total - tax;`
*	Multiplication	`tax = cost * rate;`
/	Division	`salePrice = original / 2;`
%	Modulus	`remainder = value % 3;`

```
# include <iostream.h >

 void main () {
 int x;
 x = 4 + 3;
 cout << x / 3.0 << " " << x * 2;
 }
```

```
x = 7
7/3 = 2
7 * 2 =14
```

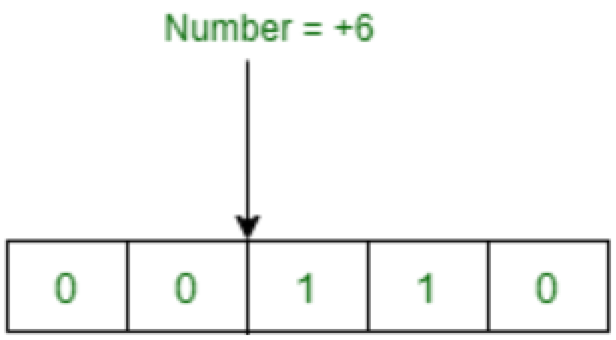
```
x = 7
7/3.0 = 2.33
7 * 2 =14
```

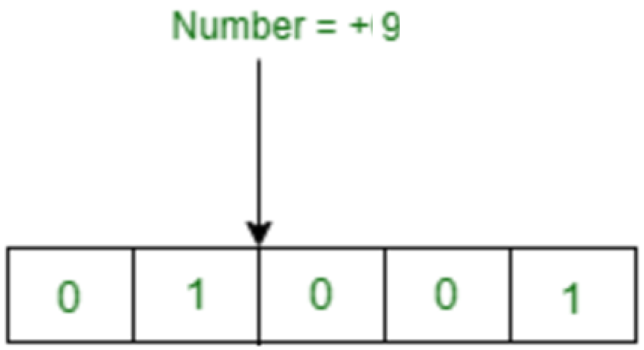
Calculations can be performed in a output statement

int / int = int
int / float = float

Bitwise operators

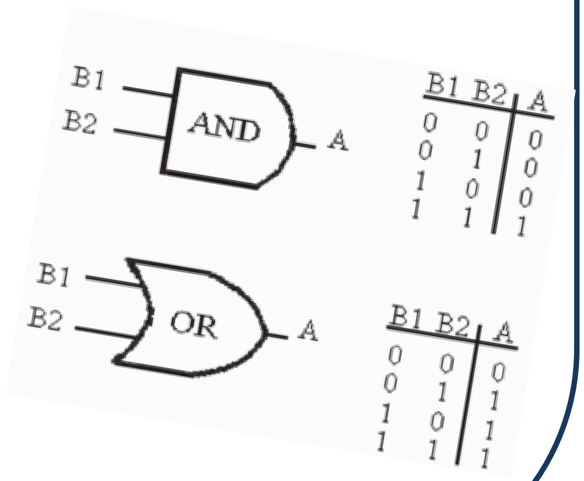
C++ provides bitwise operators, which provide bit-level control.



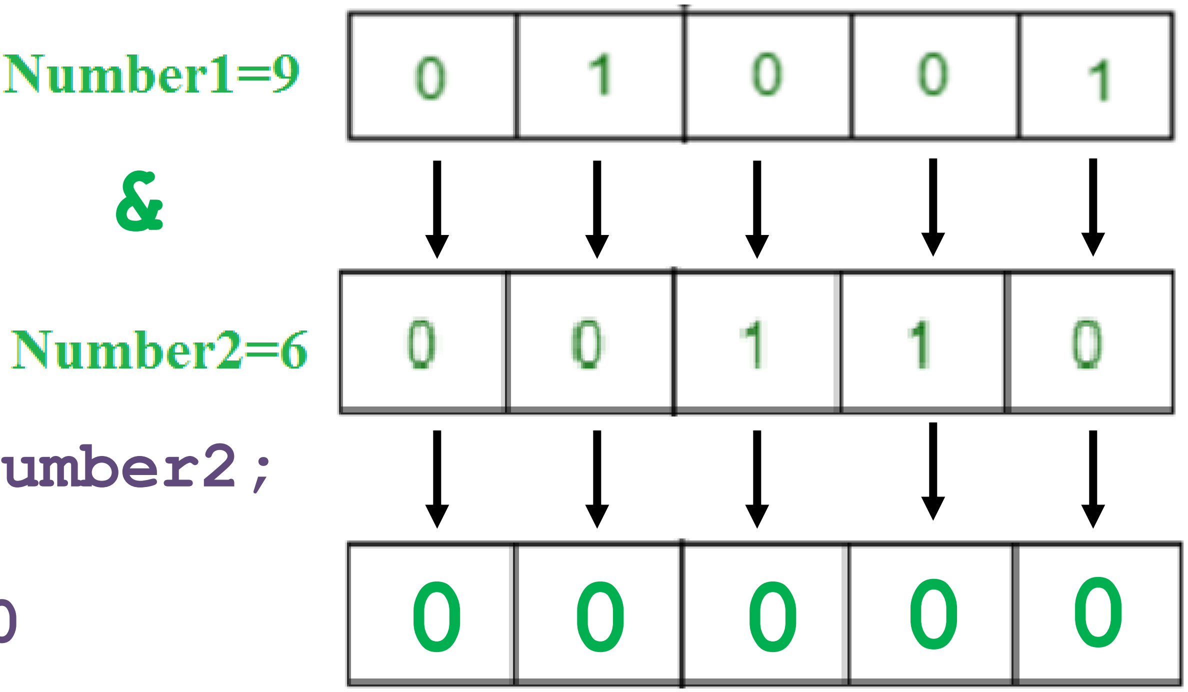


The following list describes these operations:

`& and`	`| or`
`^ xor`	`~ not`
`>> left shift`	`<< right shift`



AND (&)



Number1 & Number2;

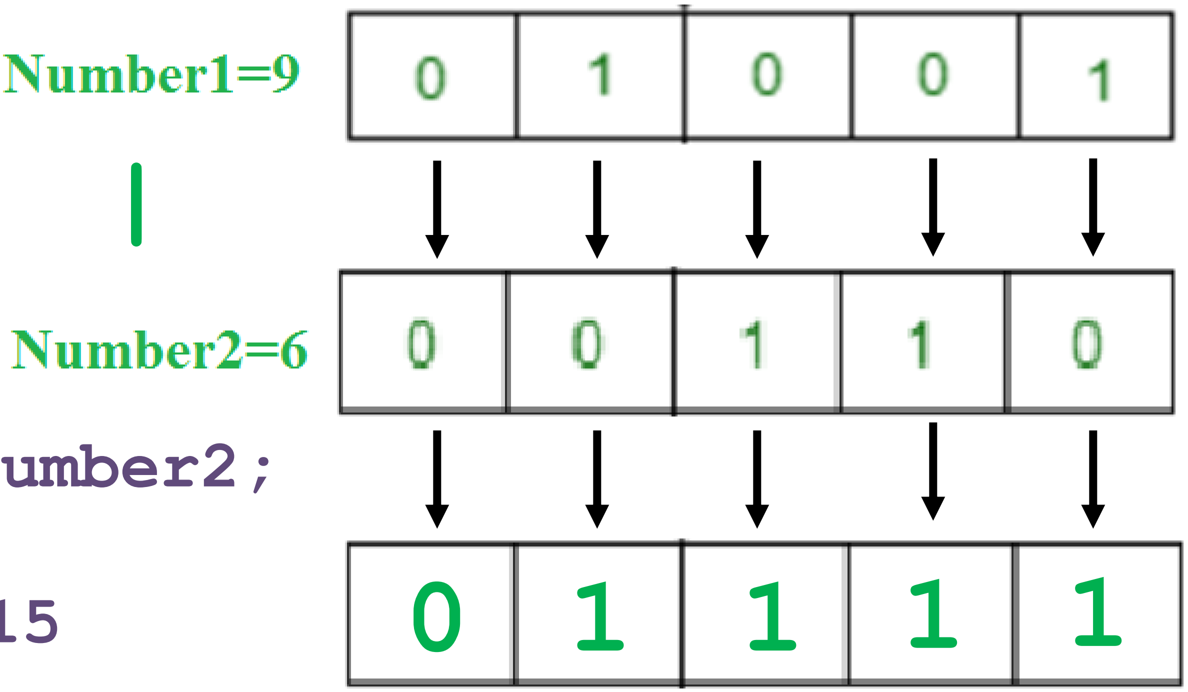
9 & 6 = 0

```
# include <iostream.h >
void main () {
 int N1, N2;
 cin >> N1 >> N2;
 cout << N1 & N2;
}
```

Let N1 = 7 , N2 = 3

```
  0 1 1 1
& 0 0 1 1
  =======
  0 0 1 1
```

OR (|)



Number1 | Number2;

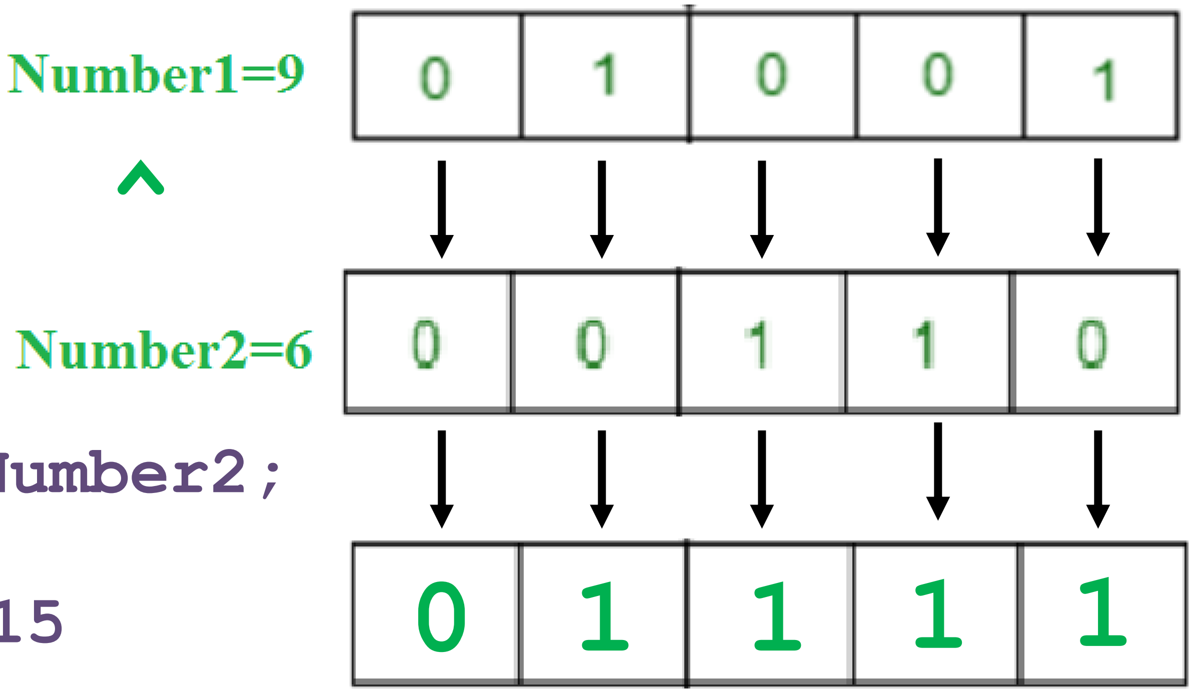
9 | 6 = 15

```
# include <iostream.h >
void main () {
 int N1, N2;
 cin >> N1 >> N2;
 cout << N1 | N2;
}
```

Let N1 = 7 , N2 = 3

```
   0 1 1 1
|  0 0 1 1
   =======
   0 1 1 1
```

XOR (^)



Number1 ^ Number2;

9 ^ 6 = 15

```
# include <iostream.h >
void main () {
 int N1, N2;
 cin >> N1 >> N2;
 cout << N1 ^ N2;
}
```

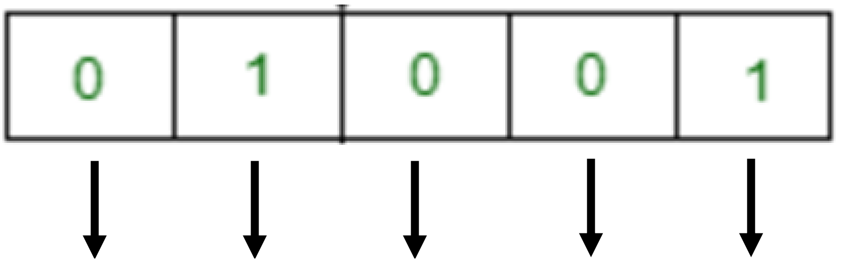
Let N1 = 7 , N2 = 3

```
   0 1 1 1
^  0 0 1 1
   =======
   0 1 0 0
```

NOT (~)

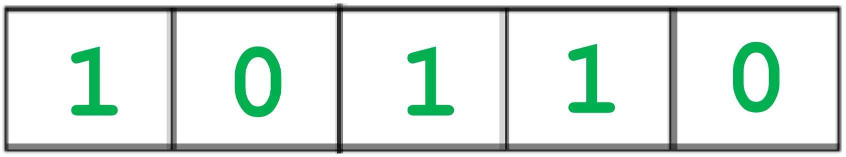
Number1=9

~



~Number1

~9 = 22



```
# include <iostream.h >
void main () {
 int N1, R;
 cin >> N1;
 R = ~ N1 ;
 cout << R;
}
```

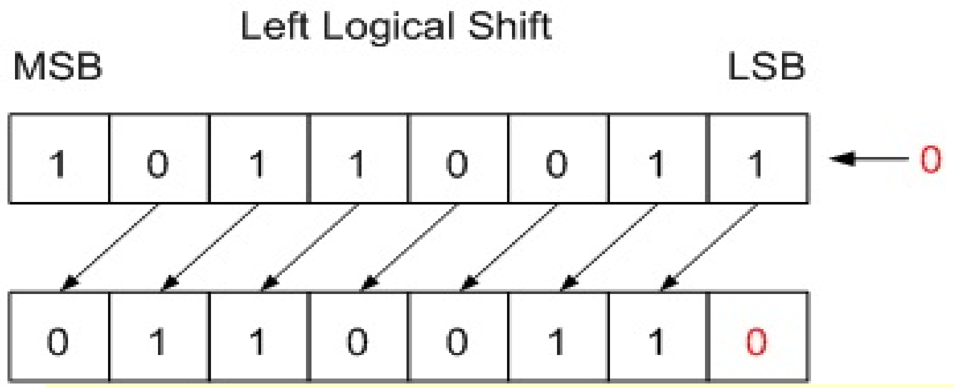
Let N1 = 7

```
~ 0 1 1 1
  =======
  1 0 0 0
```

Left Shift and Right Shift Operators in C++

Left Shift

A Left Logical Shift of one position moves each bit to the left by one. The vacant least significant bit (LSB) is filled with zero and the most significant bit (MSB) is discarded.



R = NUMBER << 1;

```
# include <iostream.h >
void main () {
 int N1, R;
 cin >> N1;
 R = N1 << 3;
 cout << R;
}
```

25 << 3

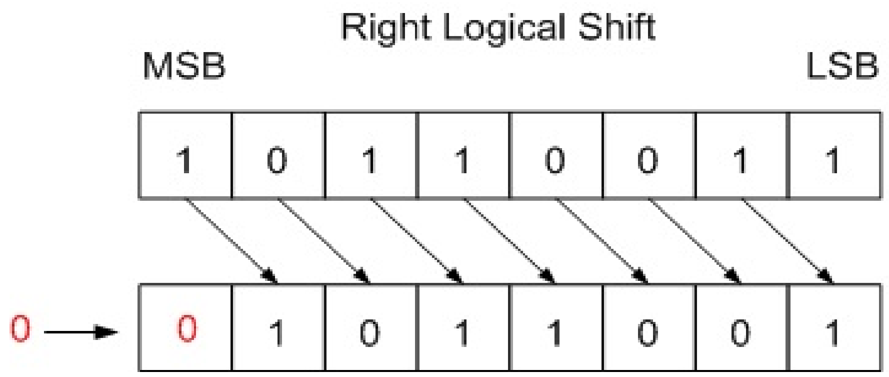
```
25 => 1 0 1 0 1
 <<3
      ============
  1 0 1 0 1 0 0 0
```

168

Left Shift and Right Shift Operators in C++

Right Shift

A Right Logical Shift of one position moves each bit to the right by one. The least significant bit is discarded and the vacant MSB is filled with zero.



R = NUMBER >> 1;

```
# include <iostream.h >
void main () {
 int N1, R;
 cin >> N1;
 R = N1 >> 3;
 cout << R;
}
```

25 >> 3

```
25 => 1 0 1 0 1
 >>3
     ============
       0 0 0 1 0 1 0 1
```

2

Multiplication by left shift:

The result of a Left Shift operation is a multiplication by 2^n , where n is the number of shifted bit positions.

Example:

Let's take the decimal number 2 represented as 8 bit binary number *00000010*. By shifting in to the left with one position we get *00000100* which is 4 in decimal representation. If we shift it once more we get binary value *00001000* which is 8 in decimal representation.

2	0	0	0	0	0	0	1	0
2 << 1 = 4	0	0	0	0	0	1	0	0
4 << 1 = 8	0	0	0	0	1	0	0	0

$2 \ll 1 \rightarrow 2 \times 2$ $\qquad$ $4 \ll 1 \rightarrow 4 \times 2$

Division by right shift:

The result of a Right Shift operation is a division by 2n , where n is the number of shifted bit positions.

Example:

If we have the binary number 01110101 (117 decimal) and we perform arithmetic right shift by 1 bit we get the binary number 00111010 (58 decimal). So we have divided the original number by 2.

117	0	1	1	1	0	1	0	1
117 >> 1 = 58	0	0	1	1	1	0	1	0
58 >> 1 = 29	0	0	0	1	1	1	0	1

117 >> 1 ➔ 117 / 2   58 >> 1 ➔ 58 / 2

```
#include <iostream.h>
void main() {
       // a = 5(00000101), b = 9(00001001)
       int a = 5, b = 9;

       cout<<"a = " << a <<","<< " b = " << b;
       cout << "a & b = " << (a & b);
       // The result is 00000001

       cout << "a | b = " << (a | b);
       // The result is 00001101

       cout << "a ^ b = " << (a ^ b);
       // The result is 00001100

       cout << "~(" << a << ") = " << (~a);
       // The result is 11111010

       cout<<"b << 1" <<" = "<< (b << 1);
       // The result is 00010010

       cout<<"b >> 1 "<<"= " << (b >> 1 );
       // The result is 00000100      }
```

C++ Programming

Ninevah University
College of Electronics Engineering
Department of Electronic Engineering
MEDICAL INSTRUMENTATION

2nd Year
2024 – 2025

Lecturer
Prof Dr. Qais Thanon

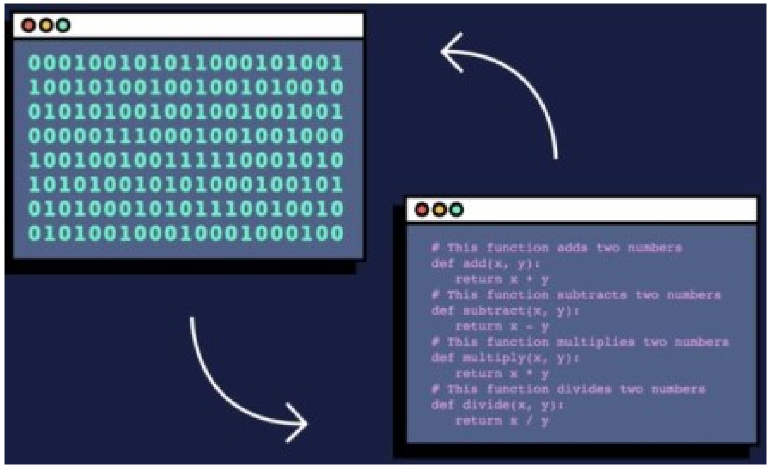
Fundamentals of C++

All the lectures of this course will upload at the
Google classroom

Introduction

What is programming language?

A programming language is a vocabulary and set of grammatical rules for instructing a computer or computing device to perform specific tasks.



```
000100101011000101001
100101001001001010010
010101001001001001001
000001110001001001000
100100100111110001010
101010010101000100101
010100010101110010010
010100100010001000100
```

```
# This function adds two numbers
def add(x, y):
    return x + y
# This function subtracts two numbers
def subtract(x, y):
    return x - y
# This function multiplies two numbers
def multiply(x, y):
    return x * y
# This function divides two numbers
def divide(x, y):
    return x / y
```

The term programming language usually refers to high-level languages, such as BASIC, **C**, **C++**, COBOL, Java, FORTRAN, Ada, and Pascal.

C++
Java
Top Programming Languages
PHP
Python

What is the C++ programming language used for?

It is used when a low-level programming language is necessary. While C++ is commonly used for develop Desktop based applications, Games and Gaming Engines, 2D and 3D animations, Developing Web Browsers, Database Software, Media Access Software, Compilers, Operating Systems, Printing and Scanning Applications, Engineering and Medical Applications, Embedded and Real-time Applications.

Why C++ language get the most interest?

C++ is an object-oriented programming language which gives a clear structure to programs and allows code to be reused, lowering development costs.

LANGUAGE CHARACTER SET AND TOKENS

types of tokens:

1. Reserved words (keywords)
2. Identifiers
3. Constants
4. String literals
5. Punctuators
6. Operators

1. Reserved words :

Identify language entities, they have special meanings to the compiler. C reserved words must be typed fully in lowercase. Some examples of reserved words from the program are const, double, int, and return.

2. Identifiers

Programmer-defined words. Needed for program variables, functions, and other program constructs. Must be unique within the same scope

1. A to Z , a to z , 0 to 9 , and the underscore “_”
2. The first character must be a letter
3. Only the first 32 characters as significant.
4. There can be no embedded blanks.
5. Reserved words cannot be used as identifiers.
6. Identifiers are case sensitive.

3. Constants :

fixed values

```
MAX_V = 12.175
```

4. String Literals

characters surrounded by double quotation marks.

"NINEVAH UNIVERSITY"

5. Punctuators

[] () {} , ; : ………* #

6. Operators

result in some kind of computation or action

```
Final_R= int_value * n ;
```

THE STRUCTURE OF a C++ PROGRAM

C++ program consists of following components:

1. Program comments

   use /* and */ to surround comments, or // to begin comment lines.

2. Preprocessor directives

   Lines that begin with a pound sign, #,

3. Type declarations

   int data_in;

4. Named constants

   const double CITY_TAX_RATE = 0.0175;

5. Statements

A statement is a specification of an action to be taken by the computer as the program executes.

Compound Statements is a list of statements enclosed in braces, { }

6. Function declarations (prototypes)
7. Function definitions
8. Function calls

Program structure in C++

The Basic Structure of C++ Program

```
#include<XXXX.h>
void main(){
statement 1;
statement 2;
statement 3;
……
statement n;
}
```

The program begins with the including the libraries using

```
#include < >
```

Between the two tags the name of the directive file (library) the required in the code is written

It can be use more than one directive file (library)

The code start with main function `main()`

The code begin with { and end with }

- ✓ All the language statements and functions are written in lowercase
- ✓ Each line should end with semicolon ;
- ✓ Comments can be written with backslash \\

Variables in C++

A variable is a name given to a memory location. It is the basic unit of storage in a program.
The value stored in a variable can be changed during program execution.
A variable is only a name given to a memory location, all the operations done on the variable effects that memory location.

How to declare variables?

A typical variable declaration is of the form:

```
// Declaring a single variable
type variable_name;

// Declaring multiple variables:
type variable1_name, variable2_name, variable3_name;
```

In C++, all the variables must be declared before use. A variable name can consist of alphabets (both upper and lower case), numbers and the underscore '_' character. However, the name must not start with a number.

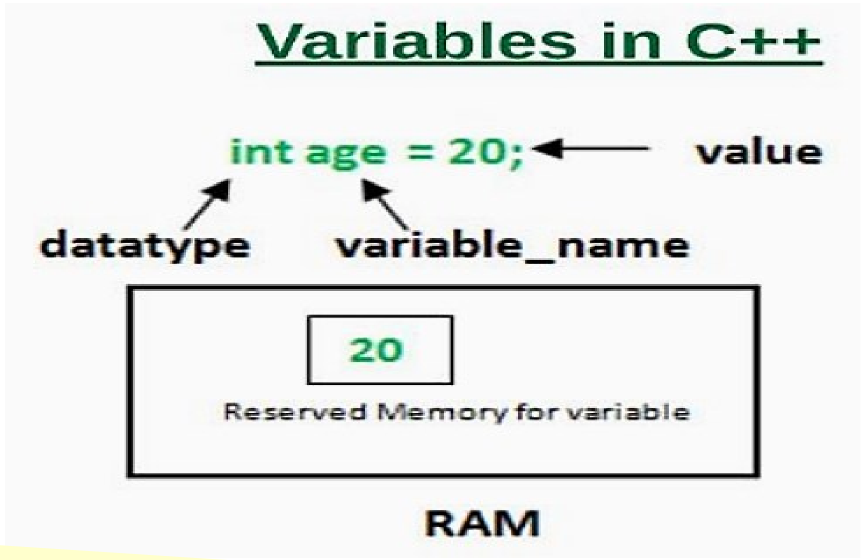
Fundamental Variable Types

There are following basic types of variable in C++

1 char	Typically a single octet (one byte). This is an integer type.
int	The most natural size of integer for the machine.
3 float	A single-precision floating point value.
4 double	A double-precision floating point value.

Example:

```
int i, j, k_1;
char c, ch;
float f, salary;
double d;
```



DON'T use C++ keywords as variable names.

Type	*Size*	*Values*
`unsigned short int`	2 bytes	0 to 65,535
`short int`	2 bytes	–32,768 to 32,767
`unsigned long int`	4 bytes	0 to 4,294,967,295
`long int`	4 bytes	–2,147,483,648 to 2,147,483,647
`int` (16 bit)	2 bytes	–32,768 to 32,767

Determining the Size of a Data Type

• The sizeof operator may be used to determine the size of a data type on any system.

Example **(sizeof (data type)**

```
#include <iostream.h>
void main() {
cout << "Size of char : " <<     sizeof(char);
cout << "Size of int : " <<      sizeof(int);
cout <<"Size of short int : "<<  sizeof(short int);
cout << "Size of long int : " << sizeof(long int);
cout << "Size of float : " <<    sizeof(float);
cout << "Size of double : " <<   sizeof(double);
}
```

C++ Programming

Ninevah University
College of Electronics Engineering
Department of Electronic Engineering

2nd Year
2024 – 2025

Lecturer No.9
Prof Dr. Qais Thanon

FUNCTIONS IN C++

All the lectures of this course will upload at the Google classroom

✓ What is the function in C++?

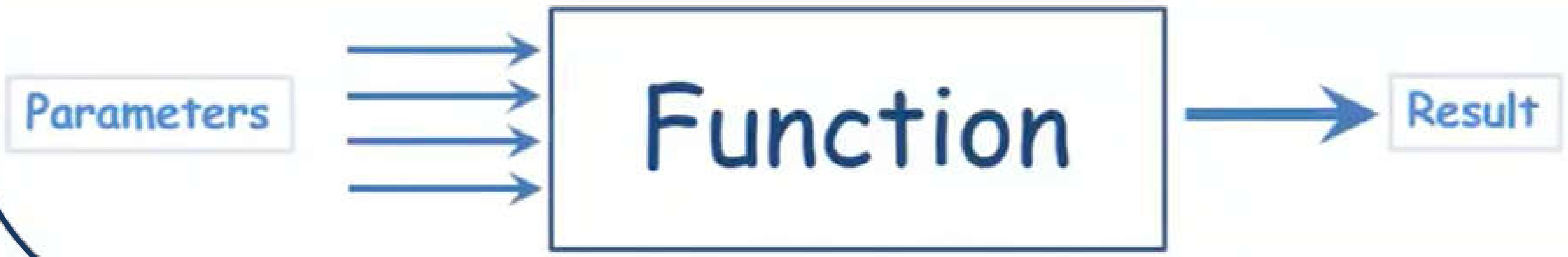
A function is block of code which is used to perform a particular task.

✓ Why we should use the function in C++?

- A program may need to repeat the same piece of code at various places.
- It may be required to perform certain task repeatedly.
- The program may become very large if functions are not used.

- Easier to Code
- Easier to Modify
- Easier to Maintain
- Reusability
- Less Programming Time
- Easier to Understand

The real reason for using function is to divide program into different parts.



There are two types of function:

1.Standard Library Functions: Predefined in C++

2.User-defined Function: Created by users

In this lecture, we will focus mostly on user-defined functions.

C++ allows the programmer to define their own function.

A user-defined function groups code to perform a specific task and that group of code is given a name (identifier).

When the function is invoked from any part of the program, it all executes the codes defined in the body of the function.

There are 3 aspects in each C++ function. They are,

- ✓ Function declaration or prototype – This informs compiler about the function name, function parameters and return value's data type.
- ✓ Function call – This calls the actual function
- ✓ Function definition – This contains all the statements to be executed.

The syntax to declare a function is:

```
returnType functionName (parameter1, parameter2,...)
        {
// function body
        }
```

- Return Type
  - void
  - value
    - char
    - int
    - float

A function is *called* by specifying its name followed by its arguments.

Non-value returning functions:

```
    function_name (data passed to function);
```

Value returning functions:

```
    results = function_name (data passed to function);
```

Before using any function it must be defined in the program. Function definition has three principal components: the first line, the parameter declarations and the body of the functions.

The first line of a function definition contains the data type of the information return by the function	function name*	set of arguments or **parameters**, separated by commas and enclosed in parentheses
int	***FACT***	**(a)**
float	***D1***	**(x1, x2)**
long	***COM_2***	**(m, n, k)**
double	***Calc***	**(y_1)**
data-type	*function-name*	(formal argument 1, formal argument 2…formal argument n)

Example: Write a C++ program to find the maximum between two numbers.

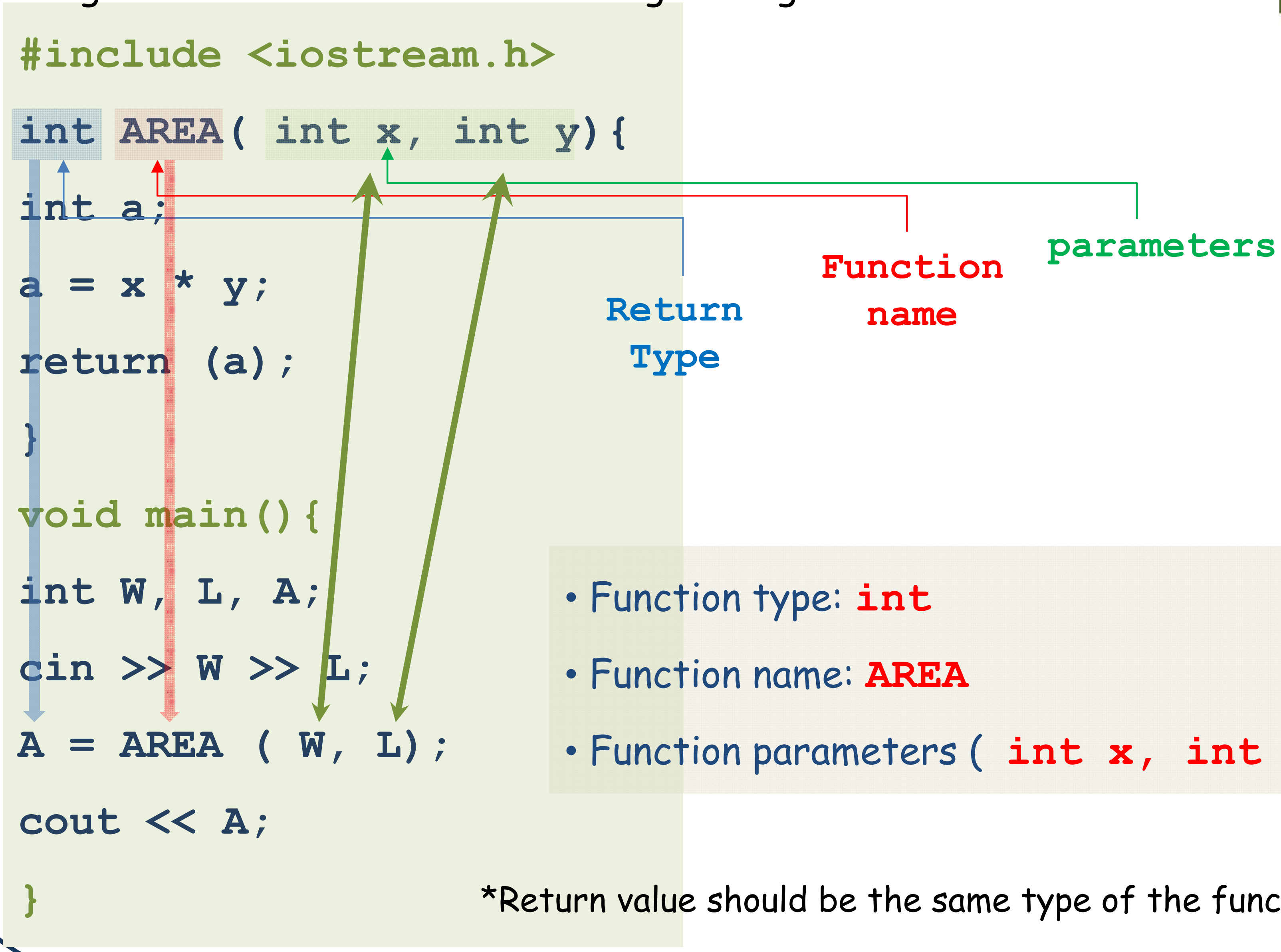
```
#include <iostream.h>

void main(){
 int a , b , c;
 cin >> a >> b;
 if(a > b) c = a;
 else c = b;
 cout << c;
}
```

Without function

```
#include <iostream.h>

int maxi (int x, int y){
 int m;
if(x > y) m = x;
 else m = y;
return m;
}

void main(){
 int a , b , c;
 cin >> a >> b;
 c = maxi (a, b);
 cout << c;
}
```

Value Return Type

Program to find the area of rectangle using function.



```
#include <iostream.h>
int AREA( int x, int y){
int a;
a = x * y;
return (a);
}
void main(){
int W, L, A;
cin >> W >> L;
A = AREA ( W, L);
cout << A;
}
```

- Function type: **int**
- Function name: **AREA**
- Function parameters (**int x, int y**)

*Return value should be the same type of the function

Example: Write a C++ program to find the value of the following series.

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots + \frac{x^n}{n!} + \ldots$$

```
#include<iostream.h>
#include<math.h>

long FACT(int n){
long f = 1;
for( int i=1; i<=n; ++i)
  f* = i;
 return (f);
}
void main(){
        int N, x;
        float Res =0.0;
        cout <<"Enter the number of treams :";
        cin >> N;
        cin >> x;
        for( int i = 0; i <= N; i++)
        Res+= pow(x,i)/FACT(i);
        cout<< Res;
        getch();}
```

Can function return more than one value?

In terms of the keyword return, no.

But it is possible to function contains two or more return statements but only one can return value to the main program.

Example: write a C++ program to find the maximum between two float variables using function.

```
#include<iostream.h>
float COMP (float N1, float N2){
 if(N1 > N2) return N1;
 else return N2;
}
```

```
void main(){
 float x , y , z;
 cin >> x >> y;
 z = COMP (x, y);
 cout << z;
}
```

⇔

```
void main(){
 float x , y;
 cin >> x >> y;
 cout << COMP (x, y);
}
```

Example: write a C++ program to find the maximum between three float variables using function.

```
#include<iostream.h>

float COMP (float N1, float N2){
 if(N1 > N2) return N1;
 else return N2;
}
```

```
void main(){
 float x , y , z, M;
 cin >> x >> y >> z;
 M = COMP (x, y);
 M = COMP (M, z);
 cout << M;
}
```

⇔

```
void main(){
 float x , y , z, M;
 cin >> x >> y >> z;
 M = COMP (x, COMP(y,z));
 cout << M;
}
```

```
cout << COMP (x, COMP(y,z));
```

Mathematical calculation can be made in return statements

Example: Write C++ program, using function, to find the area of: [1] Circle. [2] Triangle. [3] Rectangle.

```
#include<iosream.h>
#include<conio.h>
#include<math.h>
// FUNCTIONS SHOULD BE HERE
//
float CIRCLE(float R){
return (R * R * M_PI); }
```

Circle function

```
int TRIANGLE(int BASE, int
HIGHT){
return (BASE * HIGHT/2); }
```

Triangle function

```
int Rectangle(int WIDTH, int LENGTH){
return (WIDTH * LENGTH); }
```

Rectangle function

```
void main(){
int S, x, y ;
float Area, r;
clrscr();
cout<<"\n For circle enter 1 \n For triangle enter
2 \n For rectangle enter 3 ");
cin>> S;
switch(S){
case 1: cin >> r;
Area = CIRCLE(r); break;
case 2: cin>> x >> y;
Area = TRIANGLE(x, y); break;
case 3: cin>> x >> y;
Area = Rectangle(x, y);
}
cout << Area;
getch();}
```

YouTube IQ https://www.youtube.com/watch?v=-H-lUbMLy00

C++ Programming

Ninevah University
College of Electronics Engineering
Department of Electronic Engineering

2nd Year
2024 – 2025

Lecturer No.8
Prof Dr. Qais Thanon

All the lectures of this course will upload at the Google classroom

Two-Dimensional Arrays

- Arrays that we have consider up to now are one dimensional arrays, a single line of elements.

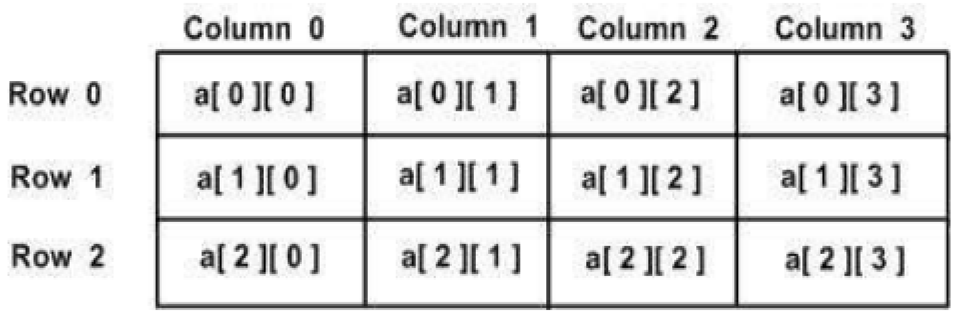
- Often data come naturally in the form of a table, e.g., spreadsheet, which need a two-dimensional array. The simplest form of the multidimensional array is the two-dimensional array.

- A two-dimensional array is, in essence, a list of one-dimensional arrays. To declare a two-dimensional integer array of size x, y, you would write something as follows:

```
type arrayName [ x ][ y ];
```

where x and y should be integers

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



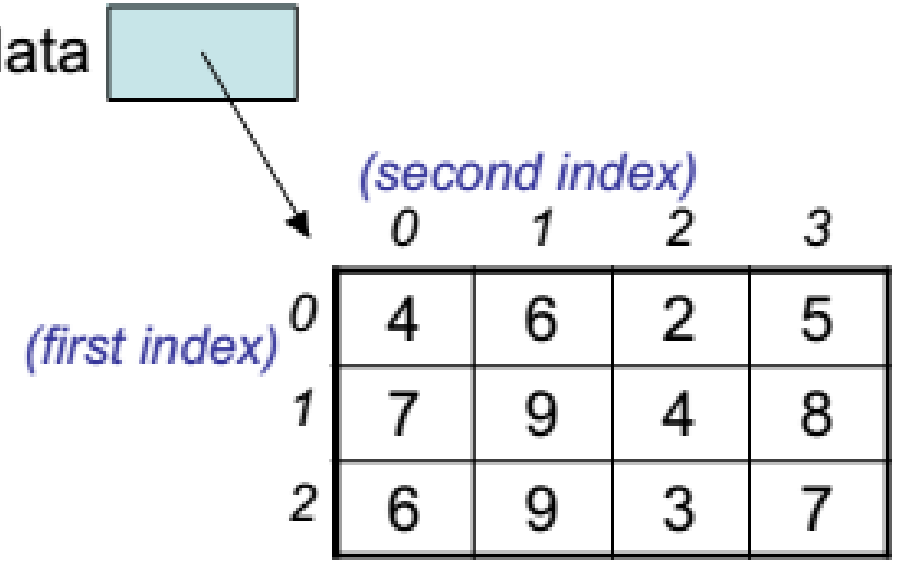
	Column 0	Column 1	Column 2	Column 3
Row 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]

Two-Dimensional Arrays

- Two-dimensional (2D) arrays are indexed by two subscripts, one for the row and one for the column.
- Example:

row *col*

```
rating[0][2] = 2
rating[1][3] = 8
```



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

Similarity with 1D Arrays

- Each element in the 2D array must by the same type,
- Subscripted variables can be use just like a variable:

```
rating[0][3] = 10;
```

- Array indices must be of type `int` and can be a variable, or expression.

```
rating[3][j] = j;
```

Initializing Two-Dimensional Arrays

Multi-dimensioned arrays may be initialized by specifying bracketed values for each row. Following is an array with 3 rows and each row have 4 columns.

```
int a[3][4] = { {0, 1, 2, 3} ,   /* initializers for row indexed by 0 */
                {4, 5, 6, 7} ,   /*initializers for row indexed by 1 */
                 {8, 9, 10, 11}  /*initializers for row indexed by 2 */
                      };
```

The nested braces are optional

```
int a[3][4] = {0,1,2,3,4,5,6,7,8,9,10,11};
```

How to enter data in a Two Dimensional Arrays

Indirect Initializing

Nested loop is used to enter data in 2-D arrays.

Suppose you want to fill an array with dimension 5x5 with number 10

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

void main(){

int matrix [5][5];
for (int m1=0 ; m1<5 ; m1++)
                  {
for (int m2=0 ; m2<5 ; m2++)
                     {
           matrix [m1][m2] = 10 ;

                     }

                 }
getch();
}
```

10	10	10	10	10
10	10	10	10	10
10	10	10	10	10
10	10	10	10	10
10	10	10	10	10

Suppose the 5×5 array as shown below, write a C++ program count:

1. How many positive number in this array?
2. The average of odd numbers?

```
#include<iostream.h>
#include<conio.h>
 void main(){
 int i, j, N=0, SUM=0, d=0,m[5][5] =
{7,-3,1,17,3,5,9,5,-21,11,1,-5,12,10,
-2,-21,9,3,-6,8,8,-7,-12,-3,11};

for (i = 0 ; i < 5 ; i++){
for (j = 0 ; j < 5 ; j++){
if (m[i][j]>0) N++;
if( m[i][j]%2 != 0) {SUM+=m[i][j];
                       d++;}
                        }
                        }
 cout << "\n\n the number positive values is"<<N;
 cout << "\n\n the average of odd numbers is"<<SUM/d;

 getch();
 }
```

7	-3	1	17	3
5	9	5	-21	11
1	-5	12	10	-2
-21	9	3	-6	8
8	-7	-12	-3	11

Can a Two-Dimensional Array used for Character?

```
#include<iostream.h>
#include<conio.h>
 void main(){
char cmatrix [3][3];
int q1, m2;
for (q1=0 ; q1<3 ; q1++){
for (m2=0 ; m2<3 ; m2++){
    cout<<"Enter name :";
    cin>>cmatrix [q1][m2];
                                }
}
 // For displaying elements of a matrix on a screen //
for (q1=0 ; q1<3 ; q1++){
for (m2=0 ; m2<3 ; m2++){
cout<<cmatrix [q1][m2] << "\t";
}
cout<<"\n";
}
 getch();
}
```

R	m	a
K	z	v
T	E	Q

C++ Programming

Ninevah University
College of Electronics Engineering
Department of Electronic Engineering
Medical Instrumentation

2nd Year
2024 – 2025

Lecturer
Prof Dr. Qais Thanon

Lecture #6 and #7

All the lectures of this course will upload at the
Google classroom

C++ NESTED LOOPS

A loop can be nested inside of another loop. C++ allows at least 256 levels of nesting

Syntax:

The syntax for a **nested for loop** statement in C++ is as follows:

```
for ( init; condition; update )
{
   for (init; condition; update )
   {
    statement(s);
   }
statement(s); // you can put more
}
```

Example:

What do you think the output of the following program would be

```
#include <iostream.h>
void main (){
int R = 5, C = 3, i, j;
for(i=0; i < R; i++) {
   for(j=0; j < C; j++){
   cout << "@"<<"\t";
                        }
   cout << "\n";}
               }
```

```
@    @    @
@    @    @
@    @    @
@    @    @
@    @    @
```

Example:

What do you think the output of the following program would be

```
#include <iostream.h>
void main (){
int R = 5, C = 3, i, j, z=0;
for(i=0; i < R; i++) {
   for(j=0; j < C; j++){
   cout << z++ <<"\t";
                        }
   cout << "\n";}
               }
```

```
0     1     2
3     4     5
6     7     8
9     10    11
12    13    14
```

C++ NESTED LOOPS

Syntax:

The syntax for a **nested while loop** statement in C++ is as follows:

```
init1;
while (condition1)
{
 update1;
statement(s); // you can put more
}
```

Example: Write a program to print half pyramid of numbers as shown :

```
#include<iostream.h>
void main(){
int N, M, Z=1;
for (N=1;N<=5; N++){

   for (M=1; M <= N; M++)
    {
     cout << Z++ << "  ";
}
cout << "\n";
}
}
```

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

Example: Write a program to print half pyramid of numbers as shown using while statement:

```
#include<iostream.h>
void main(){
int N=100, M;
while (N<=500){
 M = 100;
  while (M <= N)
   {
    cout << M << "  ";
   M+=100;
   }
cout << "\n";
 N+=100;
                 }
}
```

```
100
100  200
100  200  300
100  200  300  400
100  200  300  400  500
```

Example: Write a program to print the main diagonal of 5x5 array:

```
#include<iostream.h>
void main(){
int N, M, Z=1;
for (N=1;N<=5; N++){

   for (M=1; M <= 5; M++) {
    if (N==M) cout << N<< M<< "\t";
}
cout << "\n";
}
}
```

```
11
     22
          33
               44
                    55
```

C++ NESTED LOOPS

Syntax:

The syntax for a **nested do-while loop** statement in C++ is as follows:

```
init1;
do{
 init2;
[redacted]
 update1;
statement(s);

  } while (condition1);
```

Example: Write program to print half pyramid of numbers as shown :

```
#include<iostream.h>
void main(){
int N = 1, M, K;
do{




 N++;
  } while (N <= 5);
  }
```

```
▭▭▭▭1
▭▭▭22
▭▭333
▭4444
55555
```

Write a program in C++ to print the Floyd's Triangle

```
#include <iostream.h>
void main (){
int R = 5, C = 5, i, j, z = 0;
for(i=0; i < R; i++) {
    for(j=0; j < C; j++){

    cout << z <<"\t";
    z = !z;
                        }
    cout << "\n";}
               }
```

```
0
1 0
0 1 0
1 0 1 0
0 1 0 1 0
```

i j

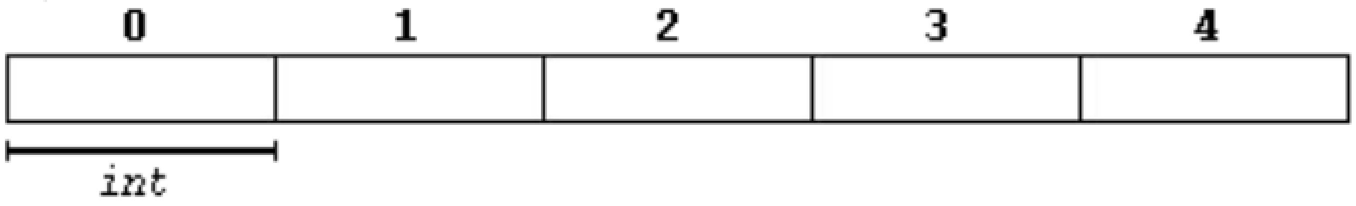
	C				
R	0 0	0 1	0 2	0 3	0 4
	1 0	1 1	1 2	1 3	1 4
	2 0	2 1	2 2	2 3	2 4
	3 0	3 1	3 2	3 3	3 4
	4 0	4 1	4 2	4 3	4 4

One diamention ARRAY

Rules for Declaring One Dimensional Array

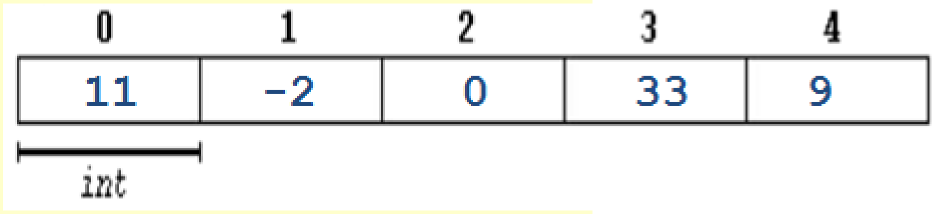
- ✓ An array variable must be declared before being used in a program.
- ✓ The declaration must have a data type(int, float, char, double, etc.), variable name, and subscript.
- ✓ The subscript represents the size of the array. ...
- ✓ An array index always starts from 0.

```
int c[ 5 ];
```



Example: Write a C++ Program to define one dimension array with 5 integer elements and print them?

```
#include<iostream.h>
#include<conio.h>
void main(){
int a[5], i;
clrscr();
cout<<"Enter any 5 num in array: \n";
```



11	-2	0	33	9

Loop for array elements reading

Loop for array elements printing

```
getch();
}
```

Example: Write a C++ Program to find the average of even numbers in one dimension N elements array?

```
#include<iostream.h>
#include<conio.h>
void main(){
 int i , N, a[10], sum=0, E_N=0;
 clrscr();
 cout<<"Enter the number of array elements:";
 cin>>N;
 for(i=0;i<N;i++)     cin>>a[i];
 for(i=0;i<N;i++) {
   if(a[i]%2==0) {sum+=a[i];
                E_N++;
             }
                }
 cout<<"\n The average is:"<< (sum *1.0/E_N);
   getch();
                }
```

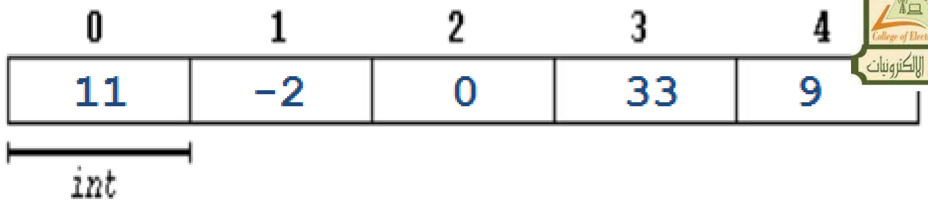
Example: Write a C++ Program to find the max. **even** number in one dimension array?

0	1	2	3	4
111	-2	0	33	9

```
#include<iostream.h>
#include<conio.h>
void main(){
int i , a[5], max;
clrscr();
cout<<"Enter any 5 num in array: \n";
for(i=0;i<5;i++) cin>>a[i];
max = a[0];
for(i=1;i<5;i++) {
if(a[i]%2 ==0 &&a[i]>max )  max = a[i];
                    }
cout<<"\n The max number is:"<< max;
getch();
}
```

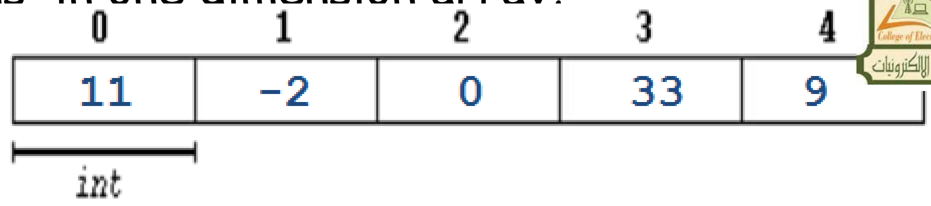
```
for(int n=0;n<5;n++)  {
if (a[n]%2==0){
 max =a[n]; break;}
}
```

Example: Write a C++ Program to find the max. number and its position in one dimension array?



```
#include<iostream.h>
#include<conio.h>
void main(){
int i , a[5], max, ind;
clrscr();
cout<<"Enter any 5 num in array: \n";
for(i=0;i<5;i++) cin>>a[i];
max = a[0];  ind=0;
for(i=1;i<5;i++) {
    if(a[i]>max) { max = a[i]; ind=i;}
}

cout << max <<“\t”<< ind;
getch();
 }
```

Example: Write a C++ Program to find the max. number and min number and swap there positions in one dimension array?



```
#include<iostream.h>
#include<conio.h>
void main(){
int i , a[5], max, min, ind1, ind2;
clrscr();
cout<<"Enter any 5 num in array: \n";
for(i=0;i<5;i++) cin>>a[i];
max = a[0];  ind1=0;
min = a[0];  ind2=0
for(i=1;i<5;i++) {
if(a[i]>max) { max = a[i]; ind1=i;}
if(a[i]<min) { min = a[i]; ind2=i;}
                  }
a[ind2] = max;    a[ind1]=min;
for(i=0;i<5;i++) cin>>a[i];
getch();
 }
```

C++ Programming

Ninevah University
College of Electronics Engineering
Department of Electronic Engineering
MEDICAL INSTRUMENTATION

2nd Year
2024 – 2025

Lecturer
Prof Dr. Qais Thanon

Lecture #5

All the lectures of this course will upload at the Google classroom

Example: Write a program in C++ to find the sum of the series

$$sum = 1 + \frac{1}{2^2} + \frac{1}{3^3} + \frac{1}{4^4} + \cdots + \frac{1}{n^n}$$

```
#include <iostream.h>
#include <math.h>
void main() {
double sum = 0, a;
int n, i;
cout << " Input the value for nth term: ";
cin >> n;

for (i = 1; i <= n; ++i) {
     a = 1 / pow(i, i);
     sum += a; }

cout << " The sum of the above series is: " << sum;
}
```

Example: Write a program in C++ to find the sum of first and last digit of a number.

Sample output:

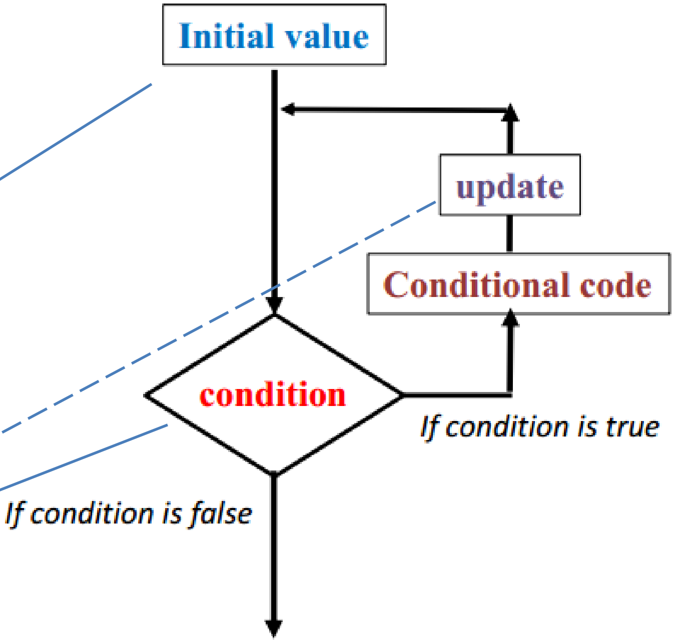
```
Input any number 1357
The first digit of the number is 7
The last digit of the number is 1
```

```
#include <iostream.h>
void main() {
int n, first, last;
cout << " Input any number: ";
cin >> n;
first = n;
last=n % 10;

for(first=n; first>=10; first=first/10);

cout<<" The first digit of "<<n<<" is: "<<first;
cout<<" The last digit of "<<n<<" is: "<<last;
cout<<" The sum is: "<<first+last;
}
```

The `while` Loop

The general form of the **while** statement is:

```
initial statement;
while (testExpression){
//loop control statements;
update statement;
}
```



```
// C++ program to print numbers from 1 to 10
#include <iostream.h>
void main()  {

int n=1;
while (n <= 10){
cout << n << " ";
n++;}
 }
```

Example: Write a program in C++ to count number of digits of an integer using while loop.

5723

```
#include <iostream.h>
#include <conio.h>
void main() {
clrscr();
long n;
int count=0;

cout << " Enter an integer: ";
cin >> n;

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

cout << " The number of digits: " << count;
}
```

Initial statement

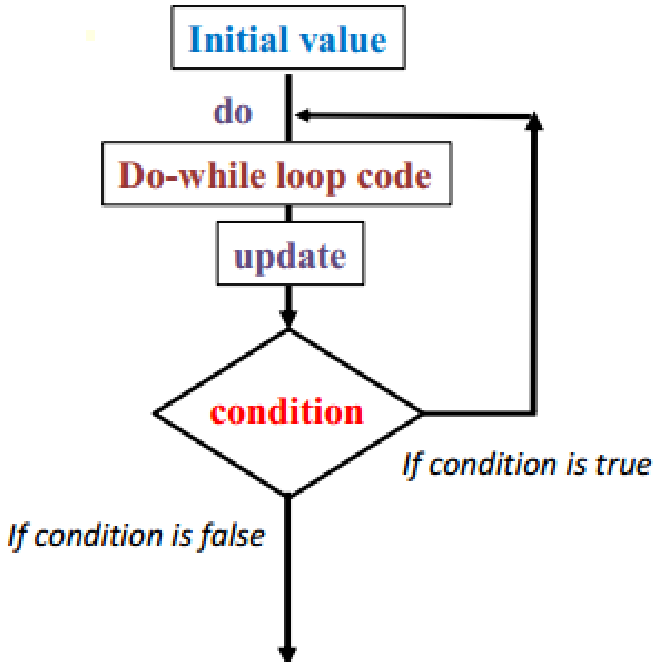
Condition tested

Update statement

The do…while Loop

The general form of a do…while statement is:

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



- The statement executes first, and then the expression is evaluated
- If the expression evaluates to true, the statement executes again
- As long as the expression in a do…while statement is true, the statement executes

Example: Write a program to calculate the summation of

10, 10.5, 11, 11.5, 12, ... -→ .., 19, 19.5, 20

```
#include<iostream.h>
void main(){
float n=10.0, sum=0;
do {
sum += n;
n+= 0.5;
} while(n<=20);
cout<<sum;
}
```

```
#include<iostream.h>
void main(){
float n, sum=0;
for(n=10; n<=20; n+= 0.5)
sum += n;
cout<<sum;
}
```

Initial statement

Condition tested

Update statement

Notes on iteration loops

- ✓ To avoid an infinite loop, the loop body must contain a statement that makes the expression `false`

- ✓ The statement can be simple or compound

- ✓ If compound, it must be in braces

- ✓ `do…while` loop has an exit condition and always iterates at least once (unlike `for` and `while`)

```
a.  i = 11;
    while (i <= 10)
    {
        cout << i << " ";
        i = i + 5;
    }
    cout << i << " ";
```

```
b.  i = 11;
    do
    {
        cout << i << " ";
        i = i + 5;
    }
    while (i <= 10);
    cout << i << " ";
```

`i = ?`

`i = ?`

The break statement

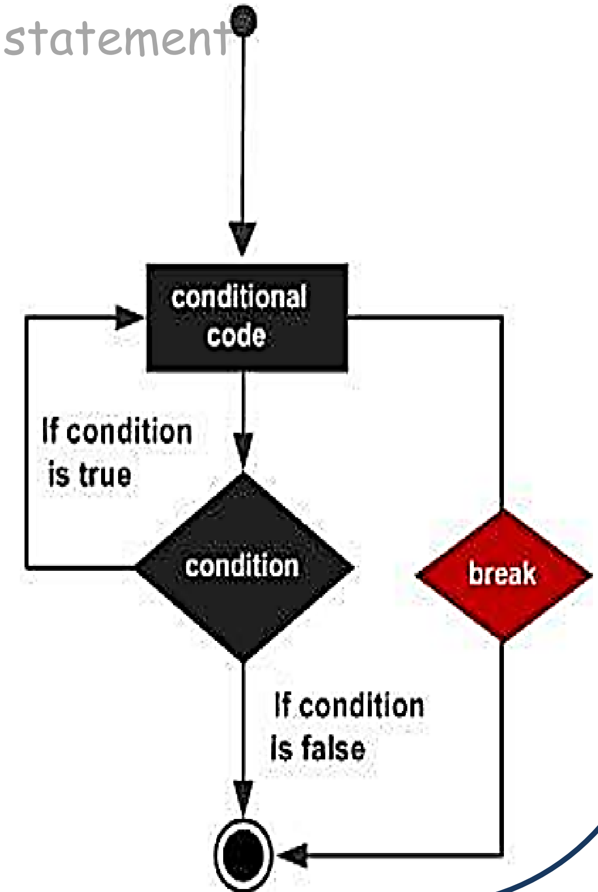
```
break;
```

The **break** statement has the following two usages in C++:

- When the **break** statement is encountered inside a loop, the loop is immediately terminated
- and program control resumes at the next statement following the loop.

It can be used to terminate a case in the **switch** statement

```
#include <iostream.h>
#include <conio.h>
void main(){
clrscr();
 int a;

for (a=10; a <= 20; a++) {
 cout << "the value of a="<< a;

                        }

    getch();
 }
```



The break statement

```
break;
```

The **break** statement has the following two usages in C++:

- When the **break** statement is encountered inside a loop, the loop is immediately terminated
- and program control resumes at the next statement following the loop.

It can be used to terminate a case in the **switch** statement

```
#include <iostream.h>
#include <conio.h>
void main(){
clrscr();
 int a = 10;
 do {
  cout << "the value of a="<< a;
  a++;

  }while (a <= 20);
  getch();
  }
```

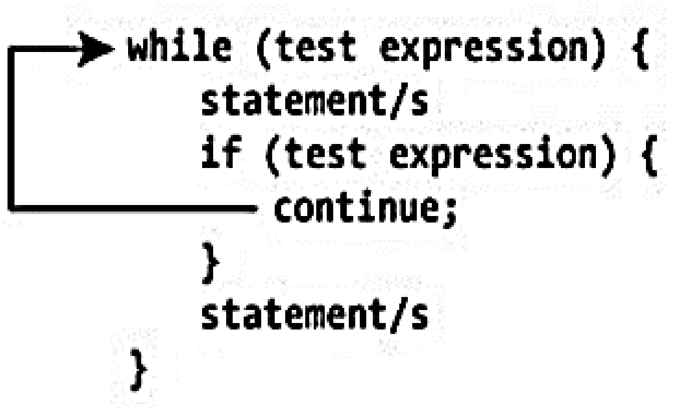
The continue statement

- It is sometimes necessary to skip a certain test condition within a loop. In such case, continue; statement is used in C++ programming.
- In practice, continue; statement is almost always used inside a conditional statement.

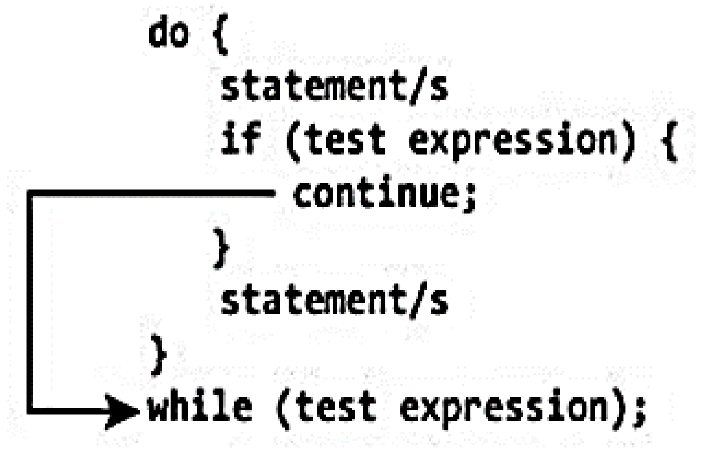
Syntax:

The syntax of a break statement in C++ is:

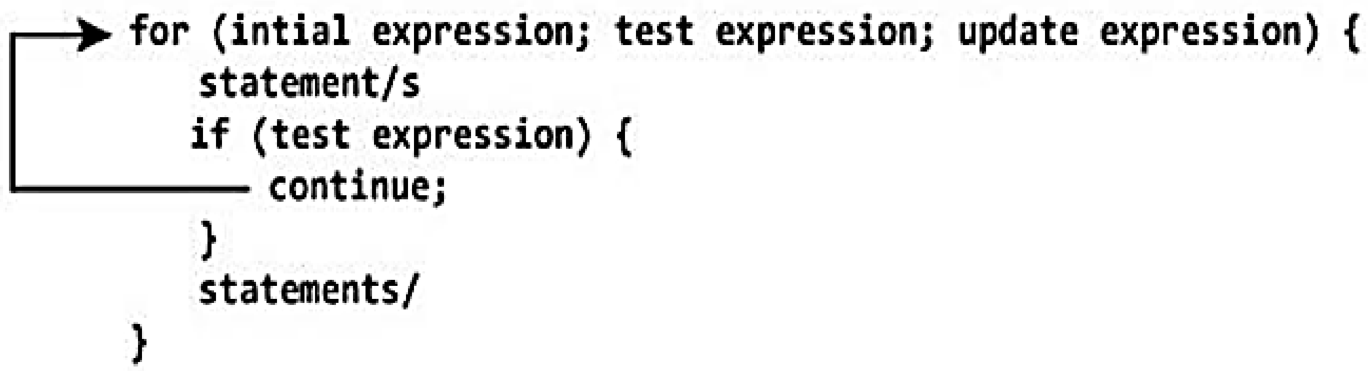
```
continue;
```



```
while (test expression) {
    statement/s
    if (test expression) {
        continue;
    }
    statement/s
}
```



```
do {
    statement/s
    if (test expression) {
        continue;
    }
    statement/s
}
while (test expression);
```



```
for (intial expression; test expression; update expression) {
    statement/s
    if (test expression) {
        continue;
    }
    statements/
}
```

Example: Write program to display integer from 1 to 10 except 6 and 9.

```
#include <iostream.h>
void main (){
for (int a = 1; a <= 10; a++){
 if( a ==6 || a==9 )continue;
 cout << "value of a: " << a ;
                              }
          }
```

```
#include <iostream.h>
void main (){
int a = 1;
do {
 if( a ==6 || a==9 )continue;
cout << "value of a: " << a;
a++;
}while( a <= 10 );
         }
```

Continue statement

The continue statement breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.

```
 #include <iostream.h>

 void main(){
 clrscr();
for (int i = 0; i < 10; i++) {



       cout << i << "\n";
                                 }
 getch();
          }
```

This example skips the value of 4:

/ C++ program to print the sum of the odd numbers between 1 and 100

```
#include <iostream.h>
void main()  {
int n, sum = 0;
for ( n = 1; n <= 100; n++)
if (n % 2 == 1) sum+=n;

 cout << "\n " <<sum;
}
```

/ C++ program to print the average of the even numbers between 100 and -100

```
#include <iostream.h>
void main()  {
int i, sum = 0, n=0;
for ( i = 100; i >= -100; i--)
if (i % 2 == 0) {sum+=i;
                  n++;}
 cout << "\n " <<sum/n;
}
```

C++ Programming

Ninevah University
College of Electronics Engineering
Department of Electronic Engineering

2nd Year
2024 – 2025

Lecturer No.11

Prof Dr. Qais Thanon

Functions in C++ language   Part III

All the lectures of this course will upload at the Google classroom

Non value-returning functions void function

There is another type of function which is called void function. This functions with no type. A void() cannot return a value that can be used.

The syntax shown below for functions:

```
void name ( argument1, argument2 ...)
{ statements; }


void main(){
// ACCESSEMENT OF A FUNCTION //
name ( actual argument1, actual argument2 ...)
}
```

Calling a function

Example: Write a C++ program to find the maximum between two numbers.

```
#include <iostream.h>
/*function definition*/
int maxi(int x,int y) {
int z;
if(x>=y)z=x;
else z=y;
return (z); }

void main() {
int a, b, M;
cin >> a >> b;
/*call function*/
M = maxi(a, b);
Cout << R;
}
```

```
#include <iostream.h>
/*function definition*/
void maxi(int x,int y) {
int z;
if(x>=y)z=x;
else z=y;
cout << z; }

void main() {
int a, b;
cin >> a >> b;
/*call function*/
maxi(a, b);
}
```

Example :Write a C/C++ program using function to swap two integer numbers?

```
#include<iostream.h>
#include<conio.h>
void SWAP(int x1, int x2){
int TEMP;
TEMP = x2;
x2 = x1;
x1 = TEMP;
cout << "first no.=" << x1;
cout << "Second no.=" << x2;
}
void main(){
int N1, N2;
clrscr();
cout<"Enter the numbers to be swapped :";
cin>> N1 >> N2;
SWAP(N1, N2);
getch();
}
```

Can the void function be recursive?

Example: Write a program in C to print first 50 natural numbers using recursion

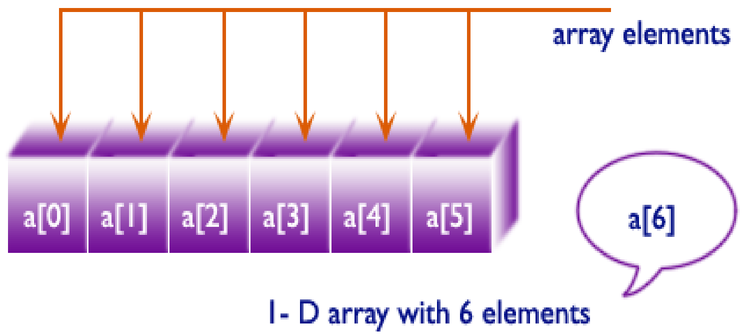
```
#include <iostream.h>

void nat_pnt(int z){
if(z<=50){
cout<< z;
 nat_pnt (z+1);}
}
void main(){

int n=1;

nat_pnt( n );
}
```

First 50 natural numbers

```
1  2  3  4  5  6  7  8  9  10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30 31
32 33 34 35 36 37 38
39 40 41 42 43 44 45
46 47 48 49 50
```

Example: Write a program in C to print the array elements using recursion.

```
#include <iostream.h>

void ArrayEle(int a[6], int n)
{
if(n<6){
cout<< a[n]<<"\t";
 ArrayEle(a,n+1);}
}


void main (){
int arr[6], i;
for (i=0; i<6; i++)
cin >> arr[i];
ArrayEle (arr,0);
}
```



```
Input the number of elements to be stored
in the array :6
Input 6 elements in the array:
element- 0 => -4
element- 1=>   7
element- 2 =>  9
element -3 =>  0
element -4 => 11
element 5 => -1
```

Expected Output:

```
The elements in the array are : -4 7 9 0 11
-1
```

Example: Write a program in C to convert a decimal number to binary using recursion.

```
#include <iostream.h>

void DIG(int z){
cout<< z%2;
 if (z!=0)
  DIG (z/2);
}

void main (){
int n;
cin>> n;
DIG( n );
}
```

```
Input any decimal number : 66
```

Expected Output:

```
The Binary value of decimal no. 66 is :
 1000010
```

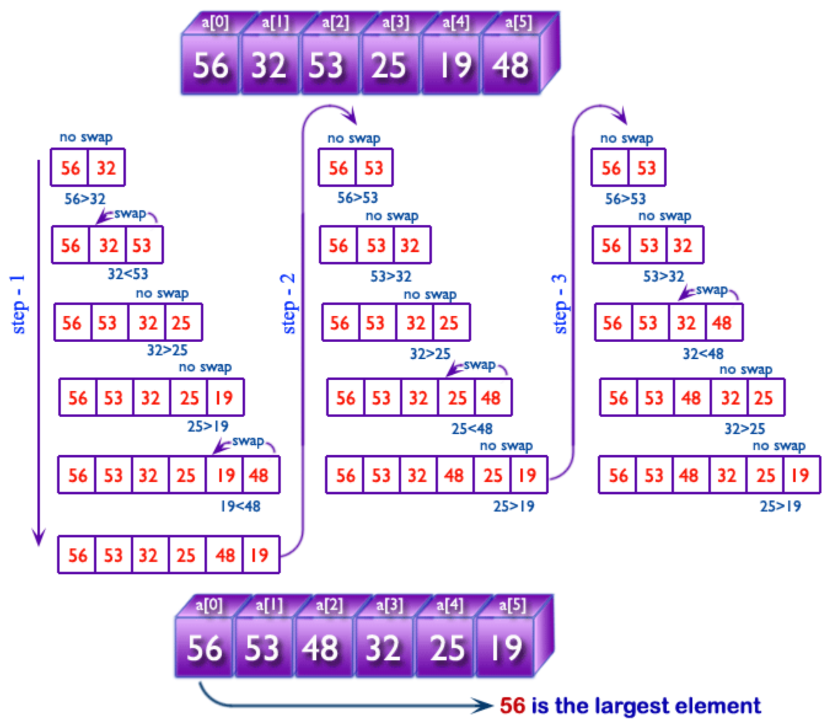
Example: Write a program in C to find the sum of digits of a number using recursion.

```
#include <iostream.h>

int cal(int N){
if(N == 0) return 0;
return ((N % 10) + cal(N / 10));
}

void main (){
int number, sum;
cin >> number;

sum = cal (number);
cout << sum;
}
```

The Sum of digits of 371=11

Example: Write a program in C to get the largest element of an array using recursion

```
#include <iostream.h>
int MaxE (int arr1[5], int n)
{
static int i=0,High =-9999;
if(i < n) {
      if(High<arr1[i])
        High=arr1[i];
        i++;
        MaxE (arr1, 5); }
       return High;
}
void main (){
int arr1[5], i, M;
for (i=0; i<5; i++)
cin >> arr1[i];
M=MaxE (arr1,5);
cout << M;
}
```



What is the static variable?
Static variables have a property of preserving their value even after they are out of their scope

```
#include<iostream.h>
int fun() {
   int count = 0;
   count++;
   return count;
}

void main(){
 cout << fun();
 cout << fun();
}
```

```
1  1
```

```
#include<iostream.h>
int fun() {
   static int count = 0;
   count++;
   return count;
}

void main(){
 cout << fun();
 cout << fun();
}
```

```
1  2
```

Example: Write a program in C to find the power of any integer number using recursion.

```
#include <iostream.h>

int POWR(int B, int P){
if(P == 0) return 0;
return (B * POWR(B , (P-1)));
}

void main (){
int Result, m, n;
cin >> m >> n;

Result = POWR (m, n);
cout << Result;
}
```

Result = m^n

Solve this program using a void function

C++ Programming

Ninevah University
College of Electronics Engineering
Department of Electronic Engineering

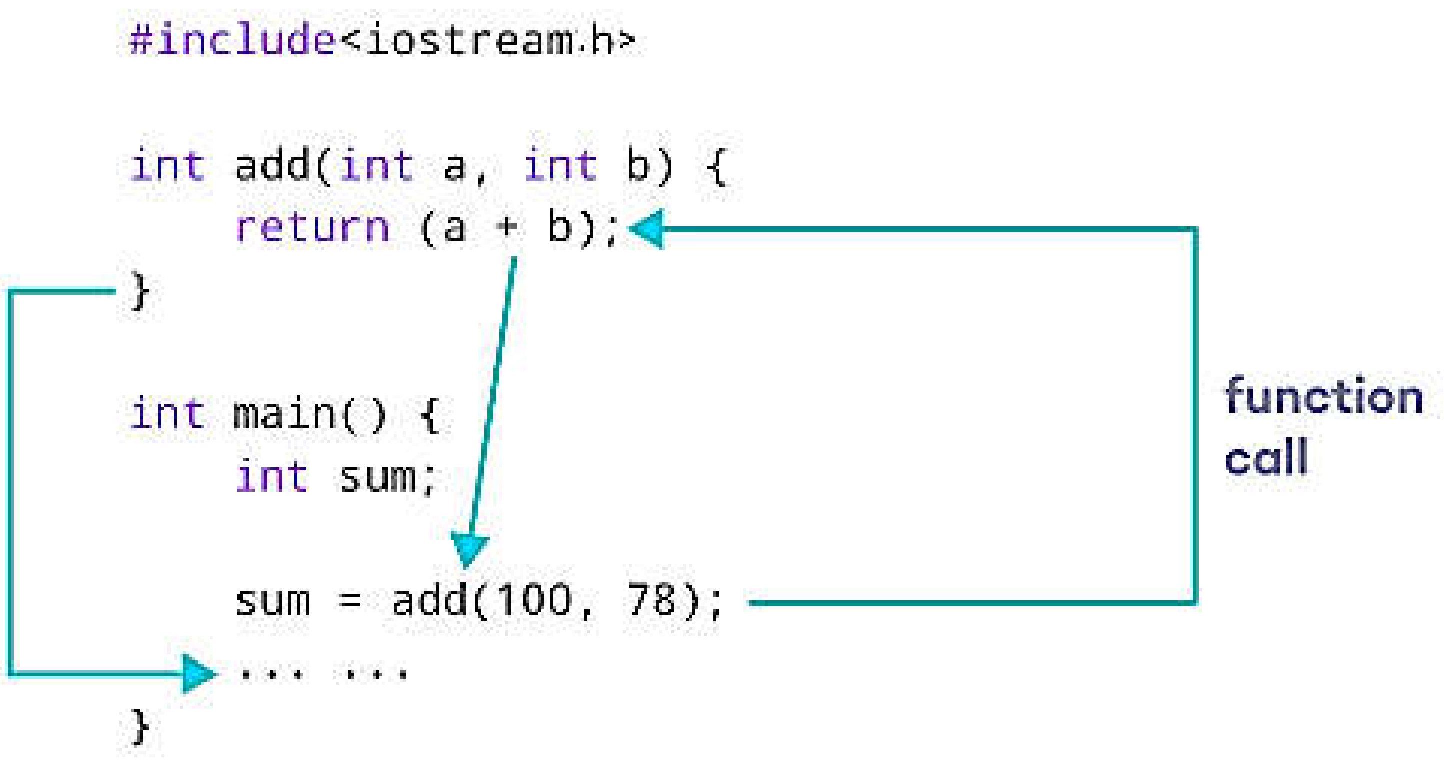
2nd Year
2024 – 2025

Lecturer No.10
Prof Dr. Qais Thanon

FUNCTIONS IN C++ (Part II)

All the lectures of this course will upload at the Google classroom

Functions in C++



```
#include<iostream.h>

int add(int a, int b) {
    return (a + b);
}

int main() {
    int sum;

    sum = add(100, 78);
    ... ...
}
```

Example: write a C++ program to find the maximum between three float variables using function.

```
#include<iostream.h>

float COMP (float N1, float N2){
 if(N1 > N2) return N1;
 else return N2;
}
```

```
void main(){
 float x , y , z, M;
 cin >> x >> y >> z;
 M = COMP (x, y);
 M = COMP (M, z);
 cout << M;
}
```

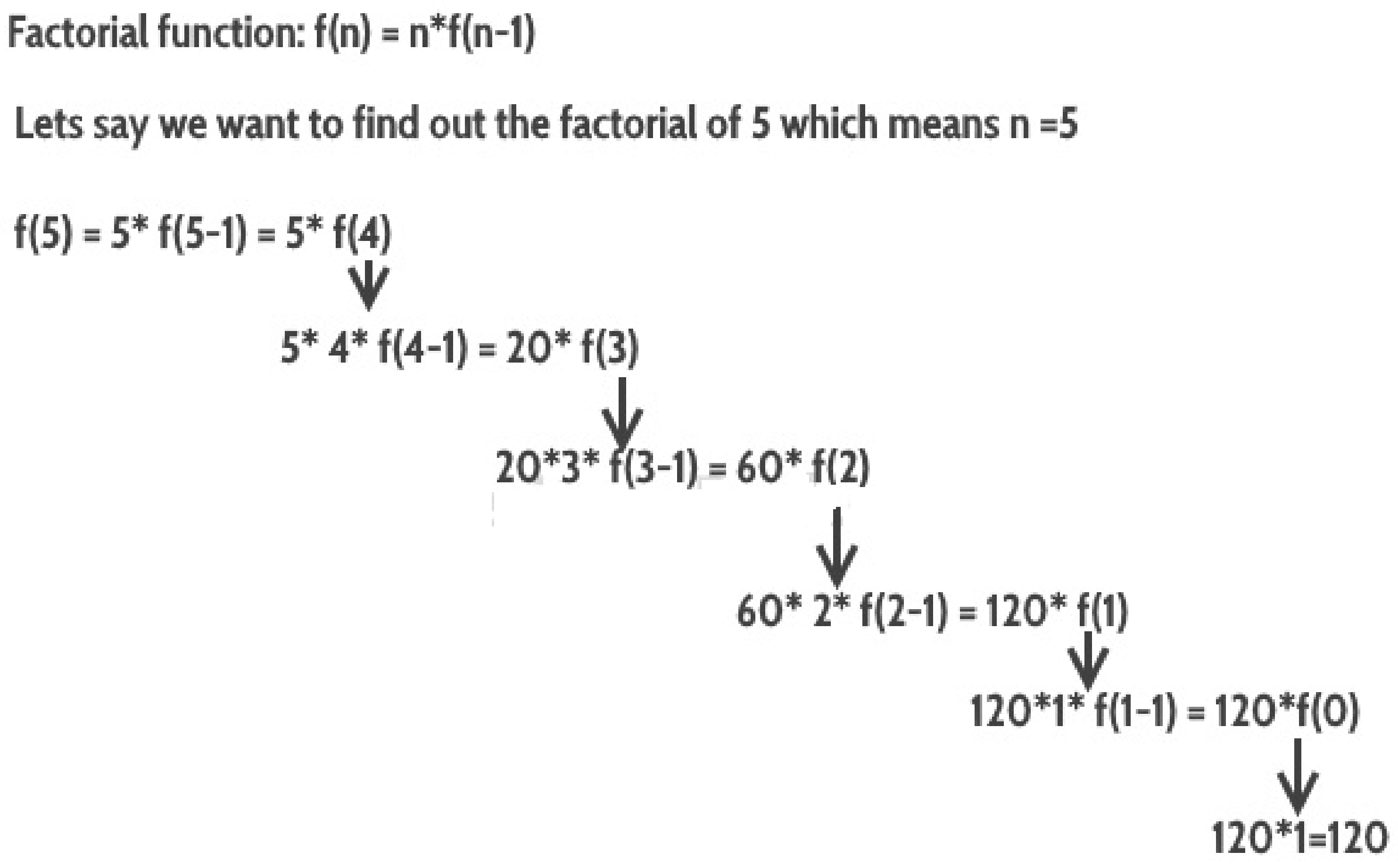
⇔

```
void main(){
 float x , y , z, M;
 cin >> x >> y >> z;
 M = COMP (x, COMP(y,z));
 cout << M;
}
```

```
cout << COMP (x, COMP(y,z));
```

Recursive function

The process in which a function calls itself is known as recursion. The popular example to understand the recursion is factorial function.



Factorial function: f(n) = n*f(n-1)

Lets say we want to find out the factorial of 5 which means n =5

f(5) = 5* f(5-1) = 5* f(4)

↓

5* 4* f(4-1) = 20* f(3)

↓

20*3* f(3-1) = 60* f(2)

↓

60* 2* f(2-1) = 120* f(1)

↓

120*1* f(1-1) = 120*f(0)

↓

120*1=120

Example: Write C++ program to find the factorial of any integer number using function.

```
#include <iostream.h>
```

```
double fa(int n){
double F=1;
for(int i=1; i<=n; i++)
F*= i;
return (F);
}
```

```
double fa(int n){
if(n<=1) return 1;
else
return n*fa(n-1);
}
```

```
void main (){
double FACT;
int k;
cin>> k;
FACT = fa( k );
cout<< FACT<<"\n";}
}
```

Pass Array to Function

In C++, we can pass arrays as an argument to a function.

The syntax for passing an array to a function is:

```
returnType functionName (dataType arrayName[arraySize])
 {
Code;
}
```

Let's see an example,

```
int total(int marks[5])
{
Code ;
}
  void main(){
  //  ACCESSEMENT OF A FUNCTION //
  var = functionName (arrayName);
  }
```

Example: Write a C++ program, using function, find mean value of N entered integers?

$$m = \frac{\text{sum of the terms}}{\text{number of terms}}$$

```
#inculde<iostream.h>
  double mean(double arr[100], int n){
  double sum = 0.0;
  for (int i = 0; i < n; i++)
  sum += arr[i];
  return sum/n;
  }
void main(){
double arr[100], sampleMean;
int size;
cout << "Enter the value of N \n";
cin >> size;
cout << "Enter %d real numbers \n";
for (int i = 0; i < size; i++)
cin>> arr[i];
// Call functions //
sampleMean = mean(arr, size);
cout<< "Mean ="<< sampleMean;
}
```

Example: Write a C++ program, using function, find count the odd number in 5X5 integer array?

```
#inculde<iostream.h>

  int ODD(int arr[5][5]){
  int sum = 0, i, j;
  for (i = 0; i < 5; i++)
  for (j = 0; j < 5; j++)
  if (arr[i][j]%2 !=0) sum ++;
  return sum;
  }
void main(){
int arr[5][5], i, j;
cout << "Enter the elements of the array\n";
for ( i = 0; i < 5; i++)
for (j = 0; j < 5; j++)
cin>> arr[i][j];
// Call functions //
cout<< "the odd numbers ="<< ODD(arr);
}
```

C++ Programming

Ninevah University
College of Electronics Engineering
Department of Electronic Engineering
MEDICAL INSTRUMENTATION

2nd Year
2024 – 2025

Lecturer
Prof Dr. Qais Thanon

Lecture #5

All the lectures of this course will upload at the Google classroom

Example: Write a program in C++ to find the sum of the series

$$sum = 1 + \frac{1}{2^2} + \frac{1}{3^3} + \frac{1}{4^4} + \cdots + \frac{1}{n^n}$$

```
#include <iostream.h>
#include <math.h>
void main() {
double sum = 0, a;
int n, i;
cout << " Input the value for nth term: ";
cin >> n;

for (i = 1; i <= n; ++i) {
     a = 1 / pow(i, i);
     sum += a; }

cout << " The sum of the above series is: " << sum;
}
```

Example: Write a program in C++ to find the sum of first and last digit of a number.

Sample output:

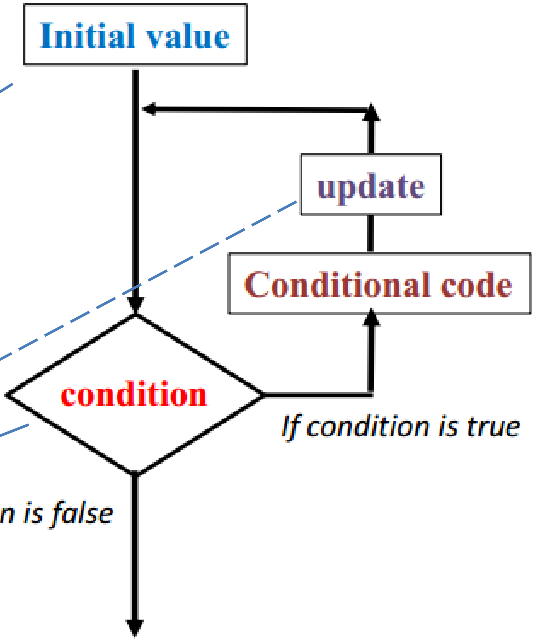
```
Input any number 1357
The first digit of the number is 7
The last digit of the number is 1
```

```
#include <iostream.h>
void main() {
int n, first, last;
cout << " Input any number: ";
cin >> n;
first = n;
last=n % 10;

for(first=n; first>=10; first=first/10);

cout<<" The first digit of "<<n<<" is: "<<first;
cout<<" The last digit of "<<n<<" is: "<<last;
cout<<" The sum is: "<<first+last;
}
```

The `while` Loop

The general form of the **while** statement is:

```
initial statement;
while (testExpression){
//loop control statements;
update statement;
}
```



```
// C++ program to print numbers from 1 to 10
#include <iostream.h>
void main()  {

int n=1;
while (n <= 10){
cout << n << " ";
n++;}
 }
```

Example: Write a program in C++ to count number of digits of an integer using while loop.

5723

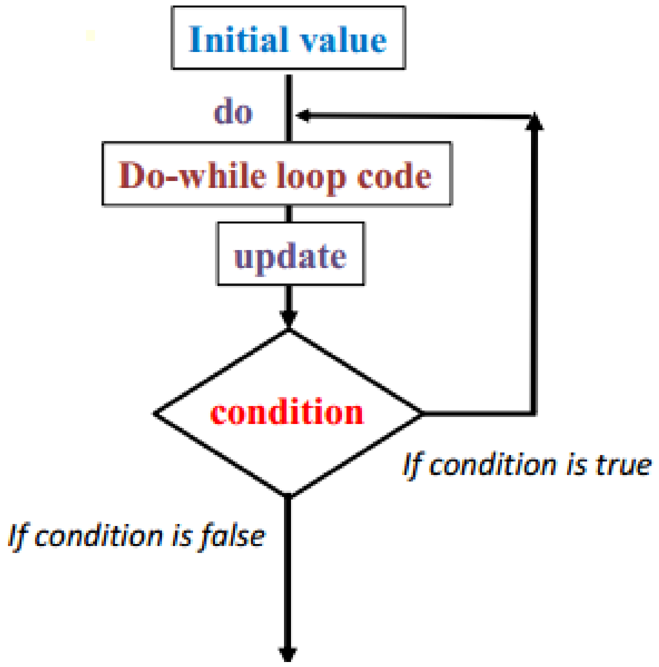
```
#include <iostream.h>
#include <conio.h>
void main() {
clrscr();
long n;
int count=0;

cout << " Enter an integer: ";
cin >> n;                Initial statement

while (n !=0) {          Condition tested
     n = n / 10;           Update statement
     count++; }

cout << " The number of digits: " << count;
}
```

The do...while Loop

The general form of a do...while statement is:

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

- The statement executes first, and then the expression is evaluated
- If the expression evaluates to true, the statement executes again
- As long as the expression in a do...while statement is true, the statement executes



Example: Write a program to calculate the summation of

10, 10.5, 11, 11.5, 12, … ……. -→ .., 19, 19.5, 20

```
#include<iostream.h>
void main(){
float n=10.0, sum=0;
do {
sum += n;
n+= 0.5;
} while(n<=20);
cout<<sum;
}
```

```
#include<iostream.h>
void main(){
float n, sum=0;
for(n=10; n<=20; n+= 0.5)
sum += n;
cout<<sum;
}
```

Initial statement

Condition tested

Update statement

Notes on iteration loops

- ✓ To avoid an infinite loop, the loop body must contain a statement that makes the expression `false`
- ✓ The statement can be simple or compound
- ✓ If compound, it must be in braces
- ✓ `do…while` loop has an exit condition and always iterates at least once (unlike `for` and `while`)

```
a.  i = 11;
    while (i <= 10)
    {
          cout << i << " ";
          i = i + 5;
    }
    cout << i << " ";
```

```
b.  i = 11;
    do
    {
          cout << i << " ";
          i = i + 5;
    }
    while (i <= 10);
   cout << i << " ";
```

`i = ?`

`i = ?`

The break statement

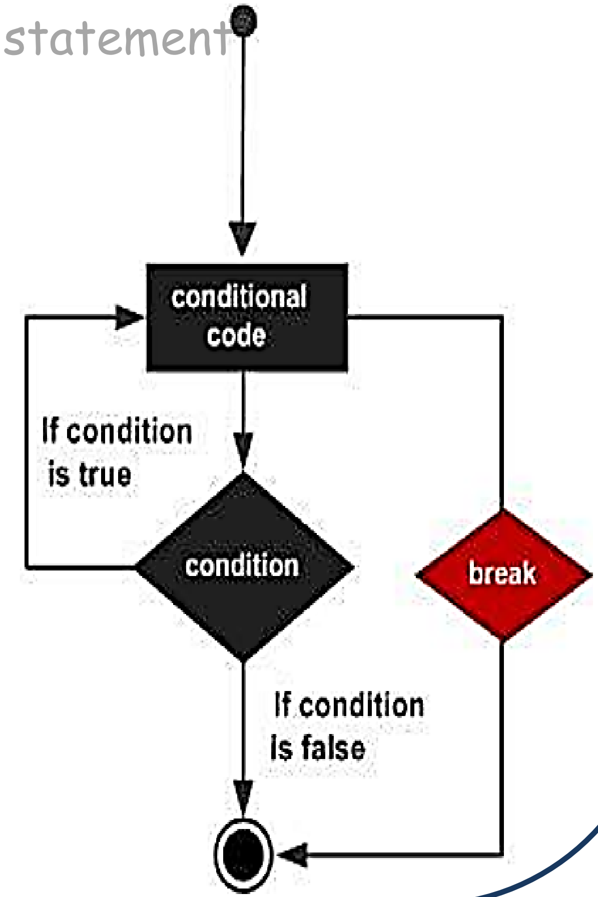
`break;`

The **break** statement has the following two usages in C++:

- When the **break** statement is encountered inside a loop, the loop is immediately terminated
- and program control resumes at the next statement following the loop.

It can be used to terminate a case in the **switch** statement

```
#include <iostream.h>
#include <conio.h>
void main(){
clrscr();
 int a;

for (a=10; a <= 20; a++) {
 cout << "the value of a="<< a;

                        }
    getch();
 }
```



The break statement

`break;`

The **break** statement has the following two usages in C++:

- When the **break** statement is encountered inside a loop, the loop is immediately terminated
- and program control resumes at the next statement following the loop.

It can be used to terminate a case in the **switch** statement

```
#include <iostream.h>
#include <conio.h>
void main(){
clrscr();
 int a = 10;
 do {
  cout << "the value of a="<< a;
  a++;

  }while (a <= 20);
  getch();
  }
```

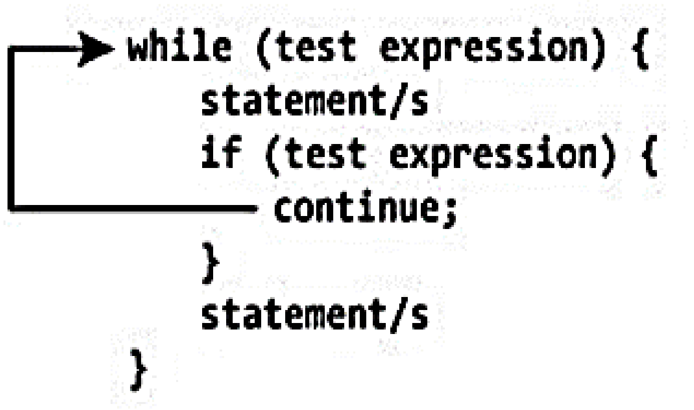
The continue statement

- It is sometimes necessary to skip a certain test condition within a loop. In such case, continue; statement is used in C++ programming.
- In practice, continue; statement is almost always used inside a conditional statement.

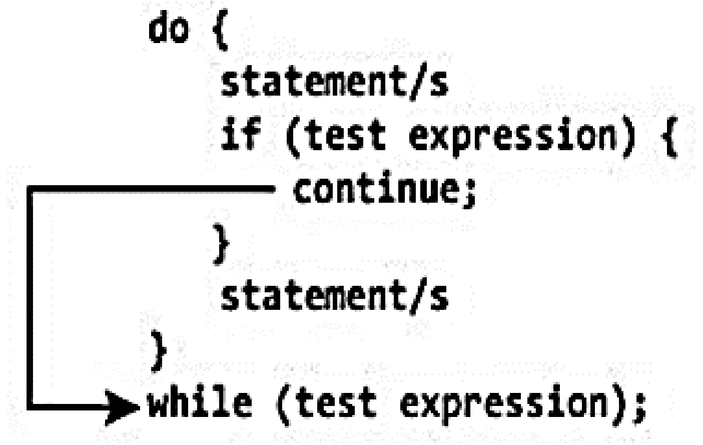
Syntax:

The syntax of a break statement in C++ is:

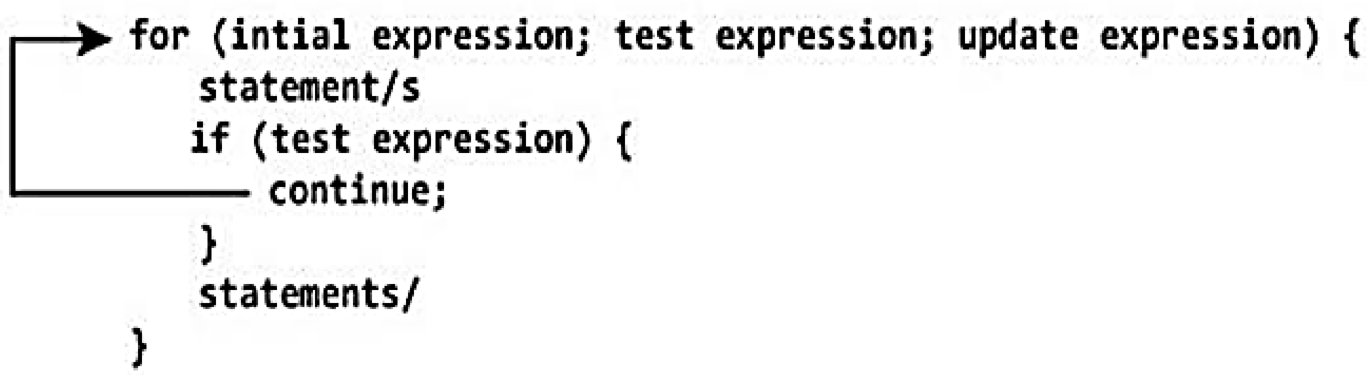
```
continue;
```



```
while (test expression) {
    statement/s
    if (test expression) {
        continue;
    }
    statement/s
}
```



```
do {
    statement/s
    if (test expression) {
        continue;
    }
    statement/s
}
while (test expression);
```



```
for (intial expression; test expression; update expression) {
      statement/s
     if (test expression) {
         continue;
      }
      statements/
}
```

Example: Write program to display integer from 1 to 10 except 6 and 9.

```
#include <iostream.h>
void main (){
for (int a = 1; a <= 10; a++){
 if( a ==6 || a==9 )continue;
 cout << "value of a: " << a ;
                              }
           }
```

```
#include <iostream.h>
void main (){
int a = 1;
do {
 if( a ==6 || a==9 )continue;
cout << "value of a: " << a;
a++;
}while( a <= 10 );
         }
```

Continue statement

The continue statement breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.

```
 #include <iostream.h>

 void main(){
 clrscr();
for (int i = 0; i < 10; i++) {



       cout << i << "\n";
                               }
 getch();
       }
```

This example skips the value of 4:

/ C++ program to print the sum of the odd numbers between 1 and 100

```
#include <iostream.h>
void main()  {
int n, sum = 0;
for ( n = 1; n <= 100; n++)
if (n % 2 == 1) sum+=n;

 cout << "\n " <<sum;
}
```

/ C++ program to print the average of the even numbers between 100 and -100

```
#include <iostream.h>
void main()  {
int i, sum = 0, n=0;
for ( i = 100; i >= -100; i--)
if (i % 2 == 0) {sum+=i;
                  n++;}
 cout << "\n " <<sum/n;
}
```

C++ Programming

Ninevah University
College of Electronics Engineering
Department of Electronic Engineering
Medical Instrumentation

2nd Year
2024 – 2025

Lecturer
Prof Dr. Qais Thanon

Lecture #6 and #7

All the lectures of this course will upload at the Google classroom

C++ NESTED LOOPS

A loop can be nested inside of another loop. C++ allows at least 256 levels of nesting

Syntax:

The syntax for a **nested for loop** statement in C++ is as follows:

```
for ( init; condition; update )
{
   for (init; condition; update )
   {
    statement(s);
   }
statement(s); // you can put more
}
```

Example:

What do you think the output of the following program would be

```
#include <iostream.h>
void main (){
int R = 5, C = 3, i, j;
for(i=0; i < R; i++) {
   for(j=0; j < C; j++){
   cout << “@”<<“\t”;
                        }
   cout << “\n”;}
               }
```

```
@       @       @
@       @       @
@       @       @
@       @       @
@       @       @
```

Example:

What do you think the output of the following program would be

```
#include <iostream.h>
void main (){
int R = 5, C = 3, i, j, z=0;
for(i=0; i < R; i++) {
   for(j=0; j < C; j++){
   cout << z++ <<"\t";
                    }
   cout << "\n";}
             }
```

```
0     1     2
3     4     5
6     7     8
9     10    11
12    13    14
```

C++ NESTED LOOPS

Syntax:

The syntax for a **nested while loop** statement in C++ is as follows:

```
init1;
while (condition1)
{



 update1;
statement(s); // you can put more
}
```

Example: Write a program to print half pyramid of numbers as shown :

```
#include<iostream.h>
void main(){
int N, M, Z=1;
for (N=1;N<=5; N++){

   for (M=1; M <= N; M++)
    {
     cout << Z++ << "  ";
}
cout << "\n";
}
}
```

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

Example: Write a program to print half pyramid of numbers as shown using while statement:

```
#include<iostream.h>
void main(){
int N=100, M;
while (N<=500){
 M = 100;
  while (M <= N)
   {
    cout << M << "  ";
   M+=100;
   }
cout << "\n";
 N+=100;
                 }
}
```

```
100
100  200
100  200   300
100  200   300   400
100  200   300   400  500
```

Example: Write a program to print the main diagonal of 5x5 array:

```
#include<iostream.h>
void main(){
int N, M, Z=1;
for (N=1;N<=5; N++){

   for (M=1; M <= 5; M++) {
    if (N==M) cout << N<< M<< "\t";
}
cout << "\n";
}
}
```

```
11
   22
      33
         44
            55
```

C++ NESTED LOOPS

Syntax:

The syntax for a **nested do-while loop** statement in C++ is as follows:

```
init1;
do{
 init2;
```
[redacted]
```
 update1;
statement(s);

   } while (condition1);
```

Example: Write program to print half pyramid of numbers as shown :

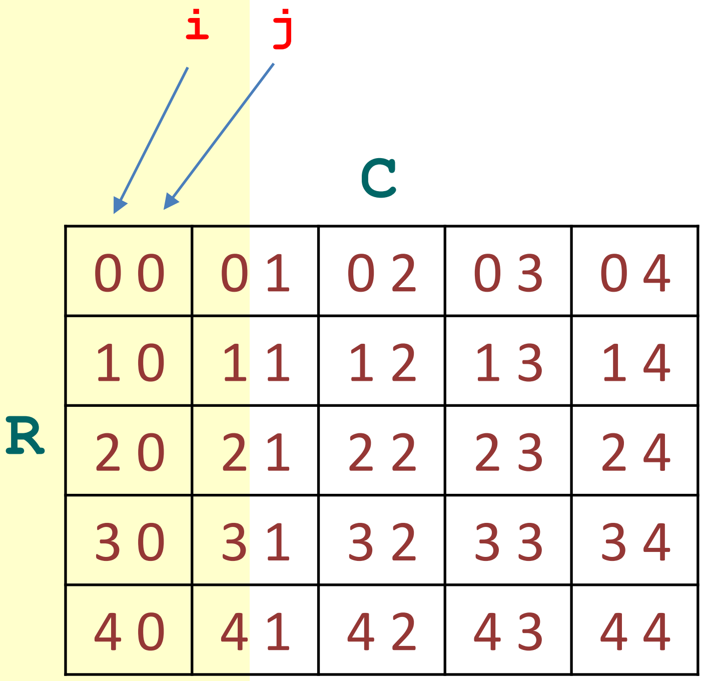
```
#include<iostream.h>
void main(){
int N = 1, M, K;
do{



 N++;
  } while (N <= 5);
  }
```

```
▭▭▭▭1
▭▭▭22
▭▭333
▭4444
55555
```

Write a program in C++ to print the Floyd's Triangle

```
#include <iostream.h>
void main (){
int R = 5, C = 5, i, j, z = 0;
for(i=0; i < R; i++) {
   for(j=0; j < C; j++){

   cout << z <<"\t";
   z = !z;
                        }
   cout << "\n";}
               }
```

```
0
1 0
0 1 0
1 0 1 0
0 1 0 1 0
```



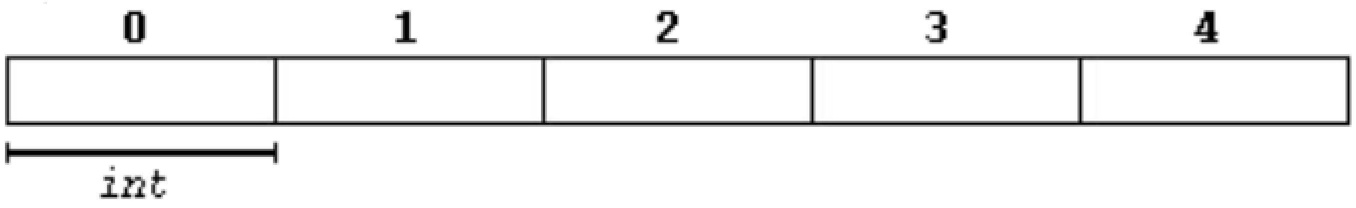
0 0	0 1	0 2	0 3	0 4
1 0	1 1	1 2	1 3	1 4
2 0	2 1	2 2	2 3	2 4
3 0	3 1	3 2	3 3	3 4
4 0	4 1	4 2	4 3	4 4

One diamention ARRAY

Rules for Declaring One Dimensional Array

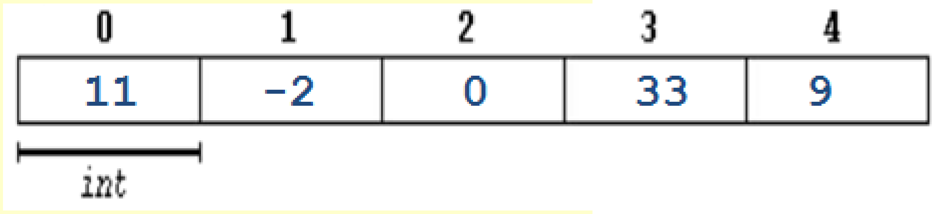
- ✓ An array variable must be declared before being used in a program.
- ✓ The declaration must have a data type(int, float, char, double, etc.), variable name, and subscript.
- ✓ The subscript represents the size of the array. ...
- ✓ An array index always starts from 0.

```
int c[ 5 ];
```



Example: Write a C++ Program to define one dimension array with 5 integer elements and print them?

```
#include<iostream.h>
#include<conio.h>
void main(){
int a[5], i;
clrscr();
cout<<"Enter any 5 num in array: \n";
```



Loop for array elements reading

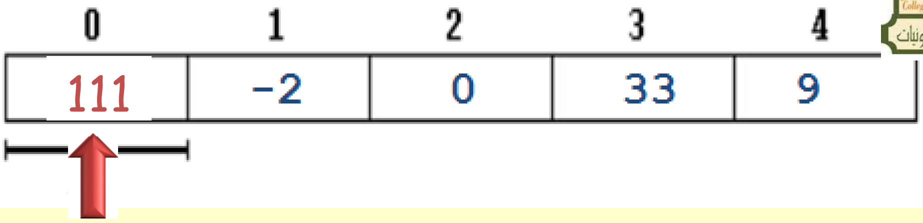
Loop for array elements printing

```
getch();
}
```

Example: Write a C++ Program to find the average of even numbers in one dimension N elements array?

```
#include<iostream.h>
#include<conio.h>
void main(){
 int i , N, a[10], sum=0, E_N=0;
 clrscr();
 cout<<"Enter the number of array elements:";
 cin>>N;
 for(i=0;i<N;i++)     cin>>a[i];
 for(i=0;i<N;i++) {
   if(a[i]%2==0) {sum+=a[i];
               E_N++;
             }
                 }
 cout<<"\n The average is:"<< (sum *1.0/E_N);
   getch();
                 }
```

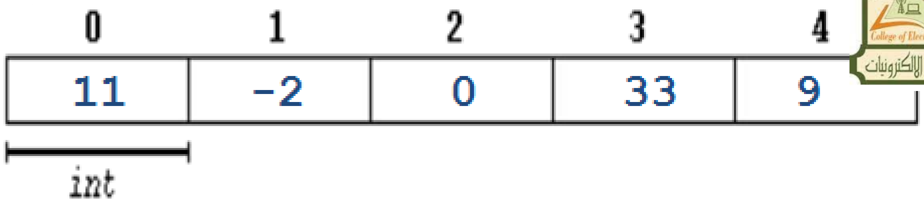
Example: Write a C++ Program to find the max. **even** number in one dimension array?



```
#include<iostream.h>
#include<conio.h>
void main(){
int i , a[5], max;
clrscr();
cout<<"Enter any 5 num in array: \n";
for(i=0;i<5;i++) cin>>a[i];
max = a[0];
for(i=1;i<5;i++) {
if(a[i]%2 ==0 &&a[i]>max )  max = a[i];
                      }
cout<<"\n The max number is:"<< max;
getch();
}
```

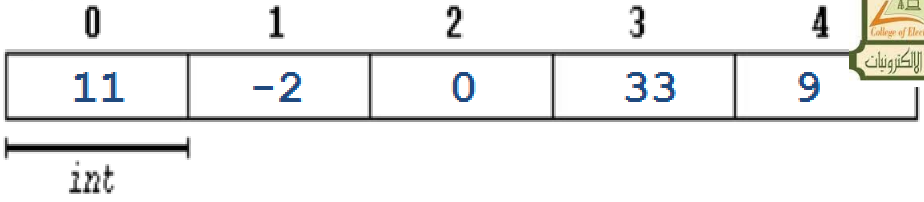
```
for(int n=0;n<5;n++) {
if (a[n]%2==0){
 max =a[n]; break;}
}
```

Example: Write a C++ Program to find the max. number and its position in one dimension array?



```
#include<iostream.h>
#include<conio.h>
void main(){
int i , a[5], max, ind;
clrscr();
cout<<"Enter any 5 num in array: \n";
for(i=0;i<5;i++) cin>>a[i];
max = a[0];  ind=0;
for(i=1;i<5;i++) {
    if(a[i]>max) { max = a[i]; ind=i;}
}

cout << max <<"\t"<< ind;
getch();
 }
```

Example: Write a C++ Program to find the max. number and min number and swap there positions in one dimension array?



```
#include<iostream.h>
#include<conio.h>
void main(){
int i , a[5], max, min, ind1, ind2;
clrscr();
cout<<"Enter any 5 num in array: \n";
for(i=0;i<5;i++) cin>>a[i];
max = a[0];   ind1=0;
min = a[0];   ind2=0
for(i=1;i<5;i++) {
if(a[i]>max) { max = a[i]; ind1=i;}
if(a[i]<min) { min = a[i]; ind2=i;}
                  }
a[ind2] = max;     a[ind1]=min;
for(i=0;i<5;i++) cin>>a[i];
getch();
  }
```