Computer Science

References:

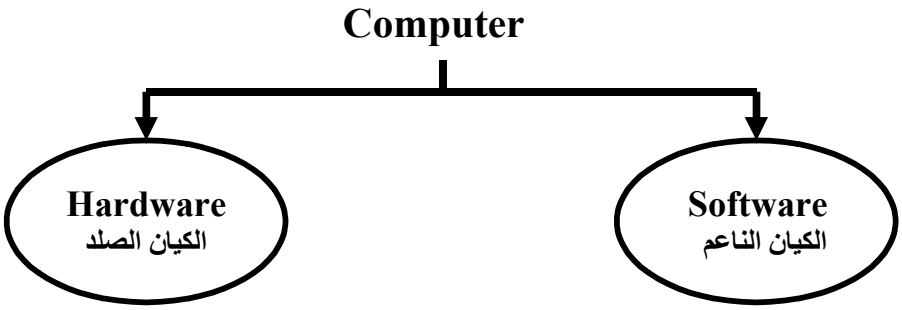
1- Borland C++ Builder By Herbert Schmidt

2- البرمجة بلغة C++ سلسلة ملخصات شـــوم

3- C / C++ Programmers References
By Herbert Schmidt

4- C++ By Dissection
By Ira Pohl

2010-2011



Hardware: Is the physical components of computer.

ويطلق على جميع مكونات الحاسبة الالكترونية وملحقاتها وتشمل على سبيل المثال:

Screen, Keyboard, CD Drive, Printer, Mouse…,etc.

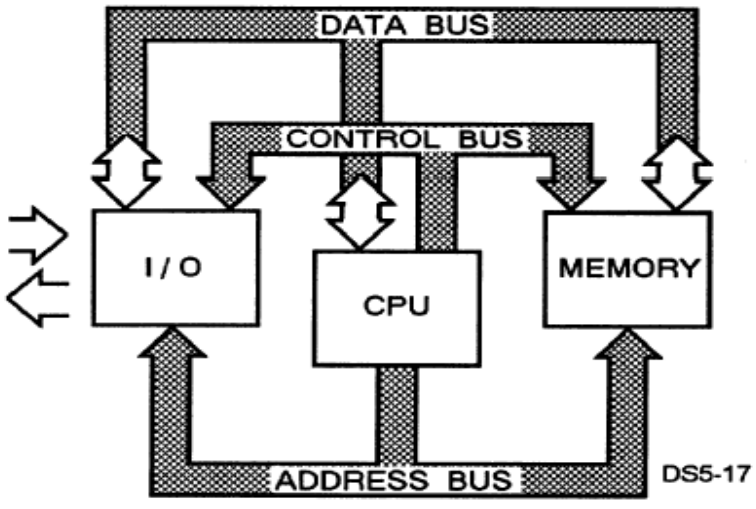
Software: Is computer programs; the set of all programs available on a computer.

ويطلق على البرامج المتنوعة التي تـــتـقبـلها الحاسبة وتشمل على سبيل المثال:

Basic, Fortran, Pascal, C, C++, ……etc

Architecture of a computer system: *معمارية الحاسبة*

The architecture of computer is shown below:



There are three main units:

1- The Central Processing Unit (CPU): وحدة المعالجة المركزية

The tasks of this unit are:

*Execute instructions of the program and processing data

تنفيذ ايعازات البرامج و معالجة المعلومات

*Data transfer between the CPU, memory and I/O ports

نقل المعطيات بين وحدة المعالجة ووحدة الخزن ومنافذ الدخل والخرج

*Controls overall system operations

السيطرة على العمليات لكل أجزاء نظام الحاسبة

2- Memory Unit: وحدة الذاكرة (الخزن)

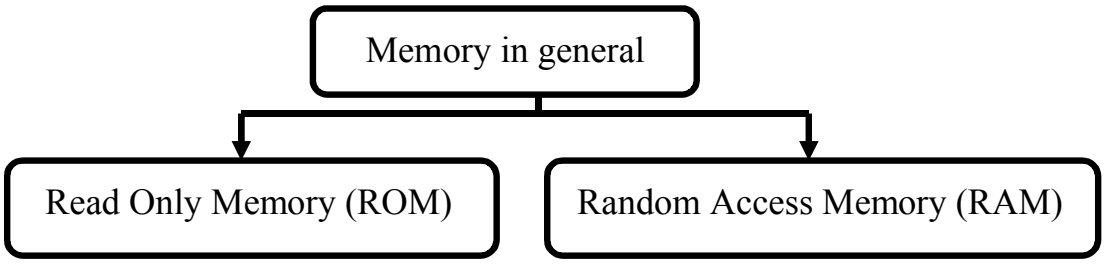
is a collection of storage registers used to transfer information in and out of the unit. The information stored in the memory as binary code in groups of bits called word. The binary is two logic levels:

*Logic (1) represented as (+5) volt in the memory cell.

*Logic (0) represented as (0) volt in the memory cell.

Bit: is binary digit (one) or (zero), and Byte: is a group of eight bits.

Store the binary data is take different manners in the compact disk (CD), magnetic tape (floppy disk), flash disk….etc.



Memory in general is divided into two general categories:

1- Read only memory: that does not change their contents, and used for various memory applications, such as fixed program storage, look-up tables, and code conversions as:

* Integrated Circuit IC 2716 (2k×8) bits.

* Compact Disk CD 700MB.

***2- Random Access Memory*:** that does not change their contents, and the data can be retrieved from memory, altered, and written back into memory. as:

*IC 2111A(2k34) Bits

*Magnetic Tape

*Floppy Disk (1.44 Mbytes)

A memory address is a particular location of a larger memory array. Usually one memory address contains one word of data. A word is one packet of information for the computer and is usually composed of many bits. Computers exist that use 1-bit words, 8-bit words, 16-bit words, 32-bit words, and 64-bit words. Handling computer data in 8-bit words is so common that the 8-bit word has its own name, the byte.

Capacity (memory size) is an important aspect of system performance; it is a useful and convenient way to describe the size of memory. At the individual part level, a computer's memory may be described as containing 65,536 bits; or, alternately, it may be called an "8K × 8" memory. Most computer memory sizes are described as a specific number of words. It is assumed that the word size is geared to the particular computer that is used with the memory. Computer memory sizes are given in Kilo, Mega, or Gega The exact size of a 1K block is 1,024, or 2^{10} . and 1M blocks is 1048576 or 2^{20} , and 1G blocks is 1073741824 or 2^{30} .

Example: Certain semiconductor memory chip is specified as: 2k × 8.

a- How many words can be stored on the chip?

b- What is the word size?

c- How many total bits can the chip store?

d- How many address lines in the chip?

Ans:

a- $k = 2^{10} = 1024$ ⇨ 2k=2×1024=2048 word.

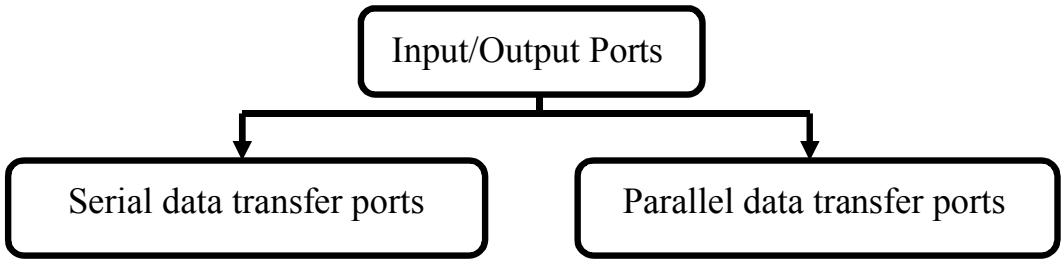
b- Word=8bits=1byte.

c- Total bits=2048×8=16384 bits.

d- Address lines=$\log_2 2k = \log_2[(2 \times 2]^{10}) = 11$

3- Input/Output Ports Unit: وحدة منافذ الادخال والاخراج

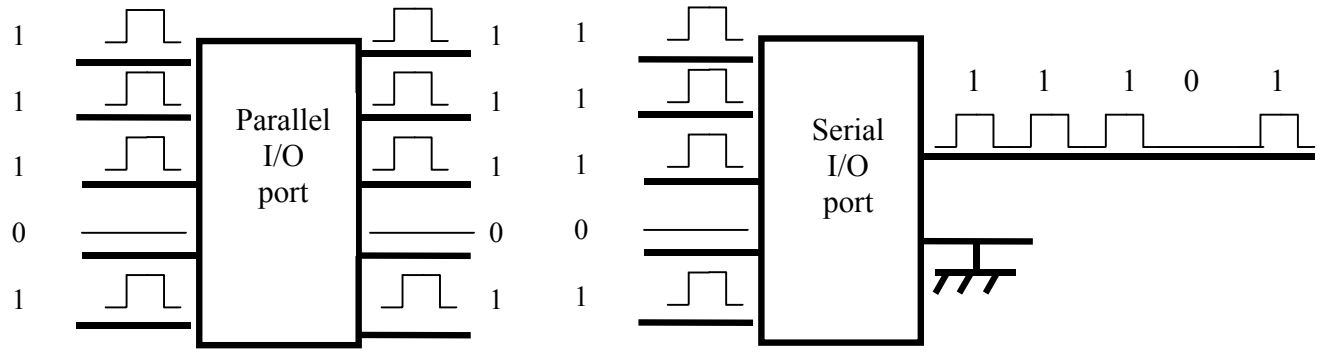
Allows the computer to communicate with and/or control other computers, peripheral devices, other subsystems (display and communication). The different types of computers vary in their organization of parallel and serial I/O data operations.



In parallel port the computer exchanges information using a parallel configuration, all bits of information represented by a byte or word are input or output simultaneously. In serial port the computer exchanges information using a serial configuration, all bits of information are input or output one at a time.

Example: Sketch a figure to show the data transfer in parallel and serial ports unit for number $(29)_{10}$ which is equal to $(11101)_2$.

Ans:



To transfer information internally between units, the computers use buses. **Buses** are groups of conductors that connect the units to one another. A ***bus*** is a parallel data communication path over which information is transferred a byte or word at a time. Therefore computer has uses three main types of buses. These are:

1- Data Bus

The bidirectional data bus, handles the transfer of all **data** and **instructions** between the units of the computer. The bidirectional data bus can only transmit in one direction at a time.

2- Address Bus

The unidirectional address bus consists of all the signals necessary to define any of the possible memory address locations within the computer.

3- Control Bus

The unidirectional control bus is used by the CPU to direct and monitor the actions of the other functional areas of the computer.

Software: is computer programs; the set of all programs available on a computer.

Computer program**: is** a list of instructions to be performed by a computer

Programming language: is a set of rules, symbols, and special words used to construct a program.

Computer Program Language:

Can be divided into three levels, these levels are:

1- **Low Level Language:** as assembly language in which a mnemonic represents each of the machine language instructions for a particular computer.
2- **Middle Level Language:** as C++ and Java languages which are combination between low level language instructions and high level languages instructions.
3- **High Level Language:** as Basic, Pascal, Fortran,…etc which a single statement translates into one or more machine language instructions.

The high level language and middle level language are translated into machine language which is made up of binary-coded instructions, that is used directly by the computer called compiler program.

The Programming Process:

1- Specify the task.
2- Discover an algorithm for its solution.
3- Code the algorithm in C++ language.
4- Test the code.

The algorithm is a logical sequence of discrete steps that describes a complete solution to a given problem.

Example: Write algorithm to find the average of five integer numbers, and then print the positive result only?

Answer:

1- Start.
2- Input the numbers as q, w, e, t. y.
3- Output is M.

4- M=(q+w+e+t+y)/5

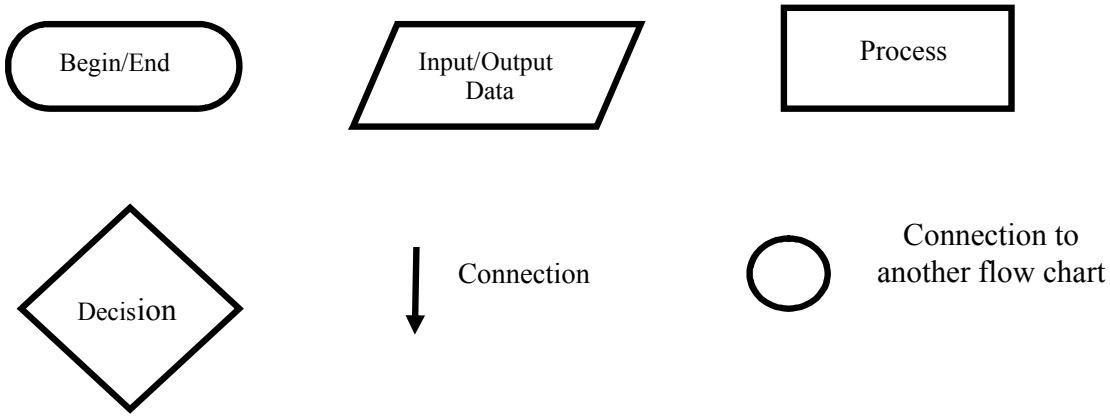
5- If (M≥ 0) print M, otherwise finish.

6- End.

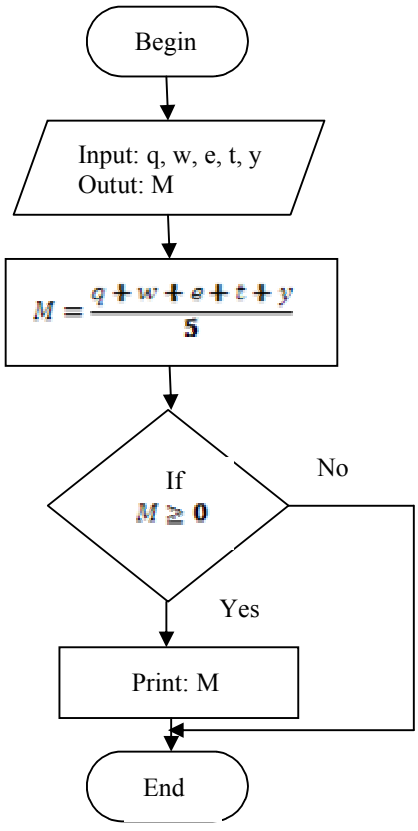
Flow Chart:

Flow chart is a set of symbols to identify both the operation required in the solution and the sequence in which they are performed.

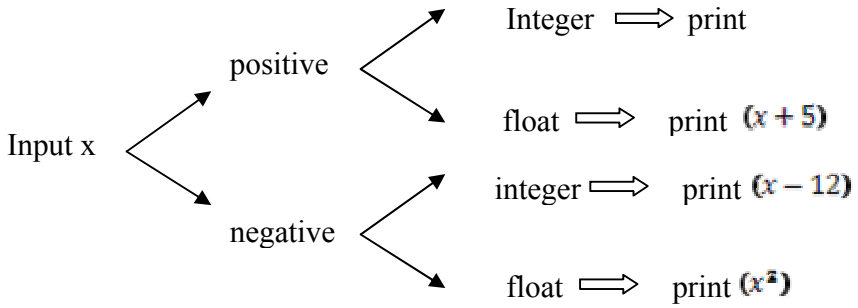
The commonly symbols are consisting of:



The flow chart with including algorithm for example above become as:



Question: Sketch flow chart including the algorithm to investigate the shape below.



Programming in C++ Language

C++ was created by Bjarne Stroustrup, beginning in 1979. The development and refinement of C++ was a major effort, spanning the 1980s and most of the 1990s. Finally, in 1998 an ANSI/ISO standard for C++ was adopted. In general terms, C++ is the object-oriented version of C. It soon expanded into being a programming language in its own right. Today, C++ is nearly twice the size of the C language. Needless to say, C++ is one of the most powerful computer languages ever devised.

Basic Source Character Set for C++ Language:

1. English Capital Letters الأبجدية الإنجليزية الكبيرة

A	B	C	D	E	F	G	H	I	J	K	L	M
N	O	P	Q	R	S	T	U	V	W	X	Y	Z

2. English Small Letters الأبجدية الإنجليزية الصغير

a	b	c	d	e	f	g	h	i	j	k	l	m
n	o	p	q	r	s	t	u	v	w	x	y	z

3- Arabic Numbers: الارقام العربية

0	1	2	3	4	5	6	7	8	9

4- Special Symbols (30):

-	+	*	/	_	(	)	{	}	[	]	#	<	>	%
:	;	?	^	&	|	~	!	=	“	‘	.	,	$	\

American Standard Code for Information Interchange (ASCII Table)										
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	nul	soh	stx	etx	eot	enq	ack	bel	bs	ht
1	nl	vt	np	cr	so	si	dle	dc1	dc2	dc3
2	dc4	nak	syn	etb	can	em	sub	esc	fs	gs
3	rs	us	sp	!	“	#	$	%	&	‘
4	(	)	*	+	,	-	.	/	0	1
5	2	3	4	5	6	7	8	9	:	;
6	<	=	>	?	@	A	B	C	D	E
7	F	G	H	I	J	K	L	M	N	O
8	P	Q	R	S	T	U	V	W	X	Y
9	Z	[	\	]	^	_	‘	a	b	c
10	d	e	f	g	h	i	j	k	l	M
11	n	o	p	q	r	s	t	u	v	w
12	x	y	z	{	|	}	~	del		

C++ Language Keywords						
asm	**else**	new	this	auto	enum	operator
bool	explicit	private	true	**break**	export	protected
case	extern	puplic	typedef	catch	false	register
char	**float**	Reinter_cast	**while**	typename	class	**for**
union	**const**	friend	**short**	**unsigned**	const_cast	**goto**
using	**continue**	throw	try	typeid	**return**	**signed**
if	**sizeof**	virtual	**default**	inline	static	**void**
delete	**int**	static_cast	volatile	**do**	**long**	struct
wchar_t	**double**	mutable	**switch**	dynamic_cast	namespace	template

Keywords:

In C++ are explicitly reserved words that have a strict meaning and may not be used in any other way. They include words used for type declarations, such as int, char, and float; words used for statement syntax, such as do, for, and if; and words used for access control, such as public, protected, and private.

Identifier Names:

The names that are used to reference variables, functions, labels, and various other user-defined objects are called *identifiers.* Identifiers can vary from one to several characters in length. An identifier in C++ is a sequence of letters, digits, and underscores. An identifier cannot begin with a digit. Uppercase and lowercase letters are treated as distinct. It is good practice to choose meaningful names as identifiers. One- or two-letter identifiers can be used when it is obvious what the name is being used for. Avoid using identifiers that are distinguished only by case differences. In principle, identifiers can be arbitrarily long, but many systems distinguish only up to the first 31 characters.

Example: determine the types of the words in C++ language:

Float, float, 1float, test23, hit!it, Beta, βeta, 5q,

Ans:

Word	Float	float	1float	Test32	Hit!it	Beta	βeta	5q
Type	identify	keyword	wrong	identify	wrong	identify	wrong	wrong

Question: are there the same identifiers :Count, COUNT, and count?

The Form of a C++ Program:

A program in C++ is a collection of functions and declarations.

Global Declarations

main()

{

statement sequence

}

The global declarations is refer to library which is the file containing the standard functions that can be used by your program. These functions include all I/O operations as well as other useful routines.

The keyword main with the parentheses () represented beginning of program execute.

The brace symbols {} is the block of main command to be execute.

The statements sequence is the instructions used to perform in the program.

Example: write a program in C++ to print on the screen of computer the integer number 5.
Ans:

First program is:

```
#include<iostream.h>
main()
{
int x=5;
cout<<x;
}
```

Computer screen appearing 5

Second program is:

```
#include<iostream.h>
main()
{
int x=5;
cout<<"x= "<<x;
}
```

Computer screen appearing x= 5

Third program is:

```
#include<iostream.h>
main()
{
cout<<5;
}
```

Computer screen appearing 5

iostream is library file consist of all instructions of input and output.

Question: write a program in C++ to print on the screen of computer your fill name.

Data Types:

There are five atomic data types in the C++: character, integer, floating-point, double floating-point, and valueless. Values of type **char** are used to hold ASCII characters or any 8-bit quantity. Variables of type **int** are used to hold integer quantities. Variables of type **float** and **double** hold real numbers. (Real numbers have both an integer and a fractional component.) The **void** type has three uses. The first is to declare explicitly a function as returning no value; the second is to declare explicitly a function as having no parameters; the third is to create generic pointers.

Type Modifiers:

Except type **void**, the basic data types may have various *modifiers* preceding them. A type modifier is used to alter the meaning of the base type to fit the needs of various situations more precisely. The list of modifiers is shown here:

signed

unsigned

long

short

The modifiers **signed**, **unsigned**, **long**, and **short** can be applied to integer base types. The character base type can be modified by **unsigned** and **signed**. You can also apply **long** to **double**. Access Modifiers that is used to control the ways in which variables may be accessed or modified. This modifier is called **const** variable of type **const** may not be changed during execution by your program.

Data Type	*Keyword*	*Byte width*	*Range*	*modified*	*Range*
Character	**char**	1	0~255	**signed**	-127~127
				unsigned	0~255
Short Integer	**short int**	2	$0\sim(2^{16}-1)$	**signed**	$-2^{15}\sim(2^{15}-1)$
				unsigned	$0\sim(2^{16}-1)$
Integer Long integer	**int** **long int**	4	$0\sim(2^{32}-1)$	**signed**	$-2^{31}\sim(2^{31}-1)$
				unsigned	$0\sim(2^{32}-1)$
Float	**float**	4	$0\sim(2^{32}-1)$	6 digits of precision 1.18E–38 to 3.40E+38	
Double	**double**	8	$0\sim(2^{64}-1)$	10 digits of precision 2.23E–308 to 1.79E+308	
Long double	**long double**	10	$0\sim(2^{80}-1)$	10 digits of precision 3.37E–4932 to 1.18E+4932	

Declaration of Variables:

As you probably know, a *variable* is a named location in memory that is used to hold a value that can be modified by the program. All variables must be declared before they are used. The general form of a declaration is shown here:

type variable list;

Variable list may consist of one or more identifier names with comma separators. Some declarations are shown here:

```
int i, j, l;
char h = 'a';
short int si;
unsigned int ahmed;
double balance, loss;
```

Question: write the instruction to declaring about:

i- integer x and y

ii- floating point -1.233432345

iii- character 5,h

There are three basic places where variables can be declared: inside functions, in the definition of function parameters, or outside all functions. These variables are called local variables, formal parameters, and global variables, respectively.

Constants:

Constants refer to fixed values that may not be altered by the program. They can be of any data type as:

```
const float pi=3.141;
```

Hexadecimal and Octal Constants

It is sometimes easier to use a number system based on 8 or 16 instead of 10. The number system based on 8 is called ***octal*** and uses the digits 0 through 7. The base 16 number system is called ***hexadecimal*** and uses the digits 0 through 9 plus the letters A through F,

which stand for 10, 11, 12, 13, 14, and 15. Because of the frequency with which these two number systems are used, C++ allows you to specify integer constants in hexadecimal or octal instead of decimal if you prefer. A hexadecimal constant must begin with a **0x** (a zero followed by an x) or **0X**, followed by the constant in hexadecimal form. An octal constant begins with a **zero**. Here are two examples:

int hex = 0x80; // 128 in decimal

int oct = 012; // 10 in decimal

Example: what is the result of program that appearing on the screen of computer.

```
#include<iostream.h>
        main()
        {
        int d=0x12, s=0x11;
        cout<<(d+s);
        }
```

Ans: The result appearing on the screen is 35

Question: what is the result if:

I: int d=12, s=0x11;

II: int d=012, s=011;

III: int d=0x12, s=011;

Example: what is the result of program that appearing on the screen of computer.

```
        #include<iostream.h>
        main()
        {
        char d='2', f='8', w;
        w=d+f;
        cout<<w;
        }
```

The result is j

2 in ASCII code ≡ 50

8 in ASCII code ≡ 56

j in ASCII code ≡ 106

Question: what is the result if:

```
char d='2', f='8'; int w;
```

C++ Operators:

An operator is a symbol that tells the compiler to perform specific mathematical or logical manipulations. There are four general classes of operators in C++: **arithmetic**, **relational and logical**, and **bitwise**. In addition, there are some special operators for particular tasks

1- . Arithmetic Operators:

The table lists C++ arithmetic operators.

Operator	***Action***
–	Subtraction, also unary minus
+	Addition
*	Multiplication
/	Division
%	Modulus
– –	Decrement
++	Increment
Precedence	***Operators***
Highest	++ – – – (unary minus)
	* / %
Lowest	+ –

The operators +, –, *, and / all work the same way in C++ as they do in most other computer languages. They can be applied to almost any built-in data type allowed by C++. When / is applied to an integer or character, any remainder is truncated; for example, **10 / 3** equals 3 in integer division. The modulus division operator (%) also works in C++ the way it does in other languages. Remember that the modulus division operation yields the remainder of an integer division. However, as such, **%** cannot be used on type **float** or **double**. The reason the last line prints a 0 and 1 is because **1 / 2** in integer division is 0 with a remainder of 1. **1 % 2** yields the remainder 1. The unary minus, in effect, multiplies its single operand by –1. That is, any number preceded by a minus sign switches its

sign.C++ allows two very useful operators not generally found in other computer languages. These are the increment and decrement operators, ++ and —. The operation ++ adds 1 to its operand, and — subtracts 1.Therefore, the following are equivalent operations:

x = x + 1;

is the same as

++x; Or x++;

Also,

x = x - 1;

is the same as

--x; Or x--

However, there is a difference when they are used in an expression. When an increment or decrement operator precedes its operand, C++ performs the increment or decrement operation prior to obtaining the operand's value.

Example: writ a program in C|++ language to test the operation of arithmetic operators with printing the result appearing on the screen of computer.

Ans:

```
#include<iostream.h>
//test arithmetic operators:
main()
{
int a=13, b=5;
cout<<a<<"+"<<b<<"="<<(a+b)<<endl;
cout<<"-b="<<(-b)<<endl;
cout<<a<<"*"<<b<<"="<<(a*b)<<endl;
cout<<a<<"/"<<b<<"="<<(a/b)<<endl;
cout<<a<<"%"<<b<<"="<<(a%b)<<endl;
cout<<"a++="<<a++<<endl;
cout<<"++a="<<++a<<endl;
cout<<"--a="<<--a<<endl;
cout<<"a--="<<a--<<endl;
}
```

The result appearing on the screen of computer is:
13+5=18
-b=-5
13*5=65
13/5=2
13%5=3
a++=13
++a=15
--a=14
a--=14

Question: What is the difference between ++a and a++.

Question: find the result of the following if int n=7, m=24;

i: 37/(5%3)

ii: m-8-n

iii: m%n++

iv: ++m-n—

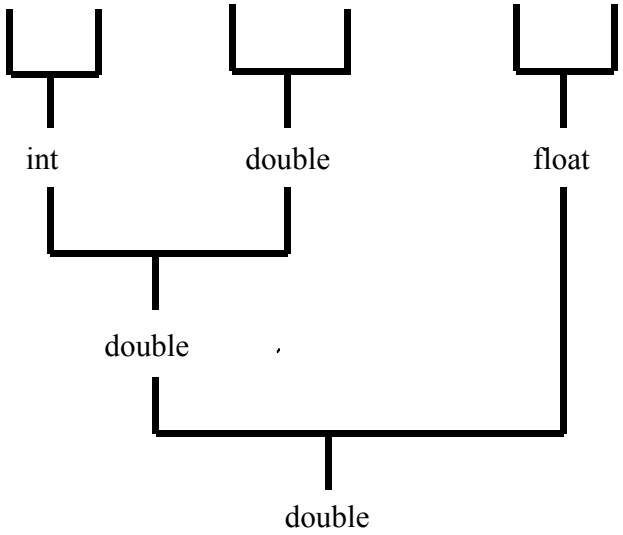
v : m*=n++

vi: m+=--n

Type Conversion:

Let data types as: Character **char**; Integer **int**; Floating point **float**; Double float point **double.**

Result = (char / int) + (float * double) - (float + int);



Question: what is the type of output if int x, int y, float z, float d:

i- z=x/y; **ii- x=z/y;** **iii- x=z/d;**

2- Relational and Logical Operators:

In the term *relational operator* the word *relational* refers to the relationships values can have with one another. In the term *logical operator* the word *logical* refers to the ways these relationships can be connected together using the rules of formal logic

The key to the concepts of relational and logical operators is the idea of **true** and **false**. In C++, *true* is any value other than 0. *False* is 0. Expressions that use relational or logical operators will return **0** for false and **1** for true.

Operator	***Action (*** **Relational Operators**)
>	Greater than
>=	Greater than or equal
<	Less than
<=	Less than or equal
= =	Equal
!=	Not equal
Operator	***Action (*** Relational Operators)
&&	AND
||	OR
!	NOT
Precedence	***Operators***
Highest	!
	> >= < <=
	= = !=
	&&
Lowest	||

Example: writ a program in C|++ language to test the operation of Relational and Logical Operators with printing the result appearing on the screen of computer.

Ans:

```
    #include<iostream.h>
// Program to test Relational and Logical Operators
    main()
    {
    int A=57,B=57;
    char C='9';
    cout<<"(A<57)="<<(A<57)<<endl;
    cout<<"(A<90)="<<(A<90)<<endl;
    cout<<"(A<30)="<<(A<30)<<endl;
    cout<<"(A<=57)="<<(A<=57)<<endl;
    cout<<"(A>B)="<<(A>B)<<endl;
    cout<<"(A>=B)="<<(A>=B)<<endl;
    cout<<"(A==B)="<<(A==B)<<endl;
    cout<<"(A!=B)="<<(A!=B)<<endl;
    cout<<"(A==C)="<<(A==C)<<endl;
    cout<<"(A&&0)="<<(A&&0)<<endl;
    cout<<"(A||20)="<<(A||20)<<endl;
    cout<<"(A=!C)="<<(A=!C)<<endl;
    cout<<"A="<<A;
    }
```

```
(A<57)=0   (A<90)=1  (A<30)=0  (A<=57)=1
(A>B)=0  (A>=B)=1  (A==B)=1  (A!=B)=0
(A==C)=1  (A&&0)=0  (A||20)=1  (A=!C)=0
A=0
```

Question: what is the result of following if: a=3, b=3

i- a<3 ii- a<=3 iii- a>b

iv- a>=b v- a==b vi- a!=b

Question: What is the difference between: a==b and a=b

Question: Write expression to obtain:

i- $-5 \leq x \leq 15$ ii- $-5 \geq x \geq 15$

iii- $x < -5$ and $x = 13$ iv- Accept all months number

Question: what is the result of (10>5 && !(10<9) || 3<=4).

3- Bitwise Operators:

Bitwise *operations* are the testing, setting, or shifting of the actual bits in a byte or word, which correspond to the standard **char** and **int** data types and variants. Bitwise operators cannot be used on type **float**, **double**, **long double**, **void**, or other more complex types. The bitwise AND, OR, and NOT (one's complement) are governed by the same truth table as were their logical equivalents except that they work on a bit-by-bit level.

Operator	***Action***
&	AND
|	OR
^	Exclusive OR (XOR)
~	One's Complement
>>	Shift Right
<<	Shift Left
Precedence	***Operators***
Highest	~
	>> <<
	&
	^
Lowest	|

The general form of the shift right statement is:

variable >> number of bit positions;

and the shift left statement is

variable << number of bit positions;

Example: writ a program in C|++ language to test the operation of Bitwise Operators with printing the result appearing on the screen of computer.

Ans:

#include<iostream.h>// Program to test Bitwise Operators

```
main()
{
int Ali=0x5c,Basim=0x3B;
cout<<"Ali ="<<Ali<<",  Basim ="<<Basim<<endl;
cout<<" Ali & Basim = "<<(Ali&Basim)<<endl;
cout<<" Ali | Basim = "<<(Ali|Basim)<<endl;
cout<<" Ali ^ Basim = "<<(Ali^Basim)<<endl;
cout<<" Ali >> 2 = "<<(Ali>>2)<<endl;
cout<<" Basim >> 4 = "<<(Basim>>4)<<endl;
cout<<" Ali << 3 = "<<(Ali<<3)<<endl;
cout<<" Basim << 1 = "<<(Basim<<1)<<endl;
cout<<" ~Ali  = "<<(~Ali)<<endl;
cout<<" ~Basim  ="<<(~Basim)<<endl;
}
```

The result appearing on the screen of computer is:

Basim =59 ,Ali =92 Basim >> 4 = 3

Ali & Basim = 24 Ali << 3 = 736

Ali | Basim = 127 BAsim << 1 = 118

Ali ^ Basim = 103 ~Ali = -93

Ali >> 2 = 23 ~Basim =-60

Question: what is the result of following if: int x=13, w=013:

i- x&w ii- x|w iii- x<<1

iv- w>>1 v- x^w vi- x^x

4- Remainder Operators:

- ***The Comma Operator:*** The comma operator strings together several expressions. The left side of the comma operator is always evaluated as **void**. This means that the expression. For example,

```
x = (y=3, y+1);
```

First assigns **y** the value 3 and then assigns **x** the value of 4.

Question: what is the result of x if:

```
Int x, y = 10;
x = (y=y-5, 25/y);
```

- ***Assignment Operator:*** In C++, the assignment operator is the single equal sign. When assigning a common value to several values, you can "string together" several assignments. For example,

  int a =10, b =10, c = 10; can be assigns as: int a=b=c=10;

  Another "shorthand" take the general form:

  Variable operation = expression;

  For example, these two assignments

  x = x+10;

  y = y/z;

  can be recoded as shown:

  x += 10;

  y /= z;

  Question: using assignments for the expressions:

  i- x=-x+3 ii- y=3/y iii- t=2(t+2)

- ***The Cast Operator:*** A *cast* is a special operator that forces one data type to be converted into another. The C++ support the form of cast shown here:

  (type) expression;

  where *type* is the desired data type.

  For example, the following cast causes the outcome of the specified integer division to be of type **double**:

  double d;

  d = (double) 10/3;

  The result is 3.3333333333

  Question: what is the result of:

  float d;

  d=(double) 10/3;

- ***The I/O Operators:*** In C++, the << and the >> are overloaded to perform I/O operations. When used in an expression in which the left operand is a stream, the >> is an input operator and the << is an output operator. In the language of C++, the >> is called an *extractor* because it extracts data from the input stream. The << is called an *inserter* because it inserts data into the output stream. The general forms of these operators are shown here:

  cin>> variable 1>> variable 2>>……>> variable N;

  cout<< variable 1<< variable 2<<……<< variable N;

  For example, the following fragment inputs two integer variables:

```
int i, j;
cin >> i >> j;
```

  The following statement displays "This is a test 10 20":

```
cout << "This is a test " << 10 << << ' ' << 4*5;
```

  Example: writ a result of the programming that appearing on the screen of computer:

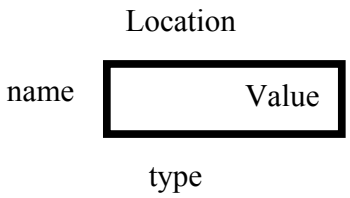
```
#include<iostream.h>
main()
{
int I=10, j=15, k=27;
cout<<I<<'\t'<<j<<'\t'<<k<<endl;
cout<<oct<<I<<'\t'<<j<<'\t'<<k<<endl;
cout<<hex<<I<<'\t'<<j<<'\t'<<k<<endl;
cout<<"Enter 3 integer, e,g. 11 11a  12"<endl;
cin>>I>>hex>>j>>k;
cout<<dec<<I<<'\t'<<j<<'\t'<<k<<endl;
}
```

  The result appearing on the screen of computer is:

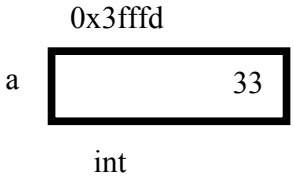
```
10              15              27
12              17              33
a               f               1b
Enter 3 integer, e,g 11  11a  12
    11          282             c
```

- ***The & and * Pointer Operators:*** A *pointer* is the memory address of a variable. A *pointer variable* is a variable that is specifically declared to hold a pointer to a value of its specified type.

We can connect the value of data with the location in form: Let int a=33;



Therefore we can represent the declaring of int a=33 in location for example 0x3fffd as:



The first pointer operator is **&**. It is a unary operator that returns the memory address of its operand. Remember that a unary operator only requires one operand. For example,

```
m = &count;
```

places into **m** the memory address of the variable **count**.

The second operator, *****, is the complement of the **&**. It is a unary operator that returns the *value of the object located at the address that follows.* For example, if **m** contains the memory address of the variable **count**, then

```
q = *m;
```

places the value of **count** into **q**.

In another meaning: n=*&n and p=&*p

Example: write a program in C++ language to find the location of int a=33 in your computer, then store it in new location.

Ans:

```
#include<iostream.h>
main()
{
int a=33;
int *p=&a;
cout<<"a="<<a<<"   ,&a="<<&a<<endl;
cout<<"p="<<p<<"   ,&p="<<&p<<endl;
}
```

The result is:

```
a=33   ,&a=0x0012ff88
```

p=0x0012ff88 ,&p=0x0012ff84

Question: Rewrite the program to repeat the value of a from address pointer.