Dr. Neamah E. Kadhim

1.1 A Brief Overview of Computers **Lecture One**

A computer is an electronic device capable of performing commands. The basic commands that a computer performs are input (get data), output (display result), storage, and performance of arithmetic and logical operations. There are two main components of a computer system: hardware and software.

Major **hardware components** include the central processing unit (CPU); main memory (MM), also called random access memory (RAM); input/output devices; and secondary storage. Some examples of input devices are the keyboard, mouse, and secondary storage. Examples of output devices are the screen, printer, and secondary storage. See figure(1).

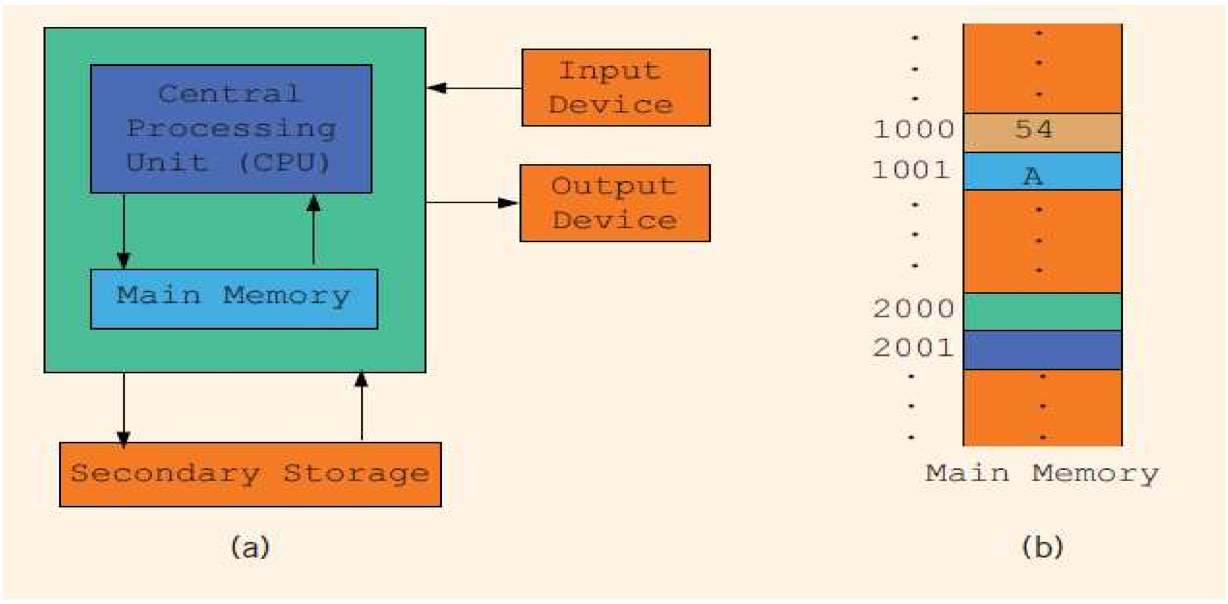


FIGURE 1: (a) Hardware components of a computer and (b)main memory

Software is programs written to perform specific tasks. For example, word processors are programs that you use to write letters, papers, and even books. All software is written in programming languages. There are two types of programs: system programs and application programs.

System programs control the computer. The operating system is the system program that loads first when you turn on your PC. Without an operating system, the computer is useless. **Application programs** perform a specific task. Word processors, spreadsheets, and games are examples of application programs. The operating system is the program that runs application programs.

Remember that a computer is an electronic device. Electrical signals are used inside the computer to process information. There are two types of electrical signals: analog and digital. **Analog signals** are continuous waveforms used to represent such things as sound. Audio tapes, for example, store data in analog signals. **Digital signals** represent information with a sequence of 0s and 1s. A 0 represents a low voltage, and a 1 represents a high voltage. Digital signals are more reliable carriers of information than analog signals and can be copied from one device to another with exact precision. Because digital signals are processed inside a computer, the language of a computer, called machine language, is a sequence of 0s and 1s. The digit 0 or 1 is called a **binary digit, or bit**. Sometimes a sequence of 0s and 1s is referred to as a binary code or a binary number. **Bit**: A binary digit 0 or 1. A sequence of eight bits is called a **byte**. Moreover, 1024 bytes is called a **kilobyte** (KB).

Every letter, number, or special symbol (such as * or {}) on your keyboard is encoded as a sequence of bits, each having a unique representation. The seven-bit American Standard Code for Information Interchange (ASCII) is the most commonly used encoding scheme on personal computers.

1.2 The Evolution of Programming Languages

The most basic language of a computer, the machine language, provides program instructions in bits. Even though most computers perform the same kinds of operations, computer designers may have chosen different sets of binary codes to perform the operations. Therefore, the machine language of one machine is not necessarily the same as the machine language of another machine. The only consistency among computers is that all data is stored and manipulated as binary codes.

Assembly languages were developed to make the programmer's job easier. In assembly language, instruction is an easy-to-remember. It is much easier to write instructions in assembly language. However, a computer cannot execute assembly language instructions directly. The instructions first have to be translated into machine language. A program called **an assembler** translates the assembly language instructions into machine language.

Moving from machine language to assembly language made programming easier, but a programmer was still forced to think in terms of individual machine instructions. The next step toward making programming easier was to devise **high-level languages** that were closer to natural languages. Basic, FORTRAN, COBOL, Pascal, C, C++, C#, and Java are all high-level languages.

EX: write the weekly wages equation

C++	Assembly	Machine Language

wages = rate * hours;	LOAD rate
	MULT hours
	STOR wages
		100100 010001
		100110 010010
		100010 010011

1.3 Programming Methodologies

Two popular approaches to programming design are the structured approach and the object-oriented approach.

Structured Programming means that dividing a problem into smaller subproblems is called structured design. Each subproblem is then analyzed, and a solution is obtained to solve the subproblem. The solutions to all of the subproblems are then combined to solve the overall problem.

Object-oriented design (OOD) is a widely used programming methodology. In OOD, the first step in the problem-solving process is to identify the components called objects, which form the basis of the solution, and to determine how these objects interact. A programming language that implements OOD is called an **object-oriented programming (OOP)** language.

1.4 Reviewing the basics of a C++ language syntax

In this section, we will briefly review basic C++ syntax. We are assuming that you have a C++ programmer with non-OOP skills.

1. **Variable declarations** and standard data types: Variables may be any length and may consist of letters, digits, and underscores. Variables are case-sensitive and must begin with a letter or an underscore. Standard data types in C++ include:

   • int: To store whole numbers

   • float: To store floating-point values

   • double: To store double-precision floating-point values

   • char: To store a single character

   • bool: For Boolean values of true or false

EX: Here are a few straightforward examples using the standard data types described above:

`int x = 5;` `int a = x;`	`float y = 9.87;` `float y2 = 10.76f; // optional 'f' suffix on float literal` `float b = y;`	`double yy = 12345.78;` `double c = yy;`
`char z= 'Z';` `char d = z;`	`bool test = true;` `bool e = test;`	

	`bool f = !test;`	

2. **Comment styles**: Two styles of comments are available in C++:

• The /* */ style provides comments spanning multiple lines of code. This style may not be nested with other comments of this same style.

• The // style of comment provides a simple comment to the end of the current line.

3. **Operators in C++:** An operator is a symbol that tells the compiler to perform specific mathematical or logical manipulations. C++ is rich in built-in operators and provide the following types of operators:

- **Arithmetic Operators**

Assume variable A holds 10 and variable B holds 20:

Operator	+	-	*	/	%	++	--
Expression	**A+B**	**B-A**	**A*B**	**B/A**	**B%3**	**B++**	**A--**
Output	**30**	**10**	**200**	**2**	**2**	**21**	**9**

- **Relational Operators**

Assume variable A holds 1 and variable B holds 2:

Operator	==	!=	>	<	>=	<=
Expression	**A==B**	**B!=A**	**A>B**	**B<A**	**B>=B**	**B<=A**
Output	**False**	**True**	**False**	**False**	**True**	**False**

- **Logical Operators**

Assume variable A holds 1 and variable B holds 0

Operator	**&&(and)**	**||(or)**	**!(not)**
Expression	**A && B**	**B || A**	**!A**

Output	False	True	False

- **Bitwise Operators**

Assume variable A holds 60(00111100) and variable B holds 13(00001101)

Operator	Expression	Output
& (And)	A&B	12 (0000 1100
| (Or)	B|A	61 (0011 1101)
^ (Xor)	A^B	49 (0011 0001)
~(Complement)	~A	195 (1100 0011)
<< (Binary Left Shift)	A << 2	240 (1111 0000)
>> (Binary Right Shift)	A>>2	15 (0000 1111)

- **Assignment Operators**

=	+=	-=	*=	/=	%=	>>=	<<=
C=3	C+=3	C-=3	C*=3	C/=3	C%=3	C>>=3	C<<=3
Assign 3 to c	C=C+3	C=C-3	C=C*3	C=C/3	C=C%3	C=C>>3	C=C<<3

- **Casting operators convert one data type to another. For example, int(2.2000) would return 2.**

- **Increment and Decrement Operators**

Pre-increment	Post-increment	Pre-decrement	Post-decrement
++variable	Variable++	--variable	Variable--
X=5 ++x; //x=6	X=5 X++; //x=6	X=5 --x ;//x=4	X=5 x --;//x=4

x = 5; **y = ++x;//x=6,y=6**	**x = 5;** **y = x++;//x=6,y=5**	**x = 5;** **y = --x;//x=4,y=4**	**x = 5;** **y = x--;//x=4,y=5**

1.5 Control Structures

A computer can process a program in one of the following ways: in **sequence**, **selectively**, by making a choice, which is also called a branch; **repetitively**, by executing a statement over and over, using a structure called a loop; or by **calling a function**. Figure 2 illustrates the first three types of program flow.

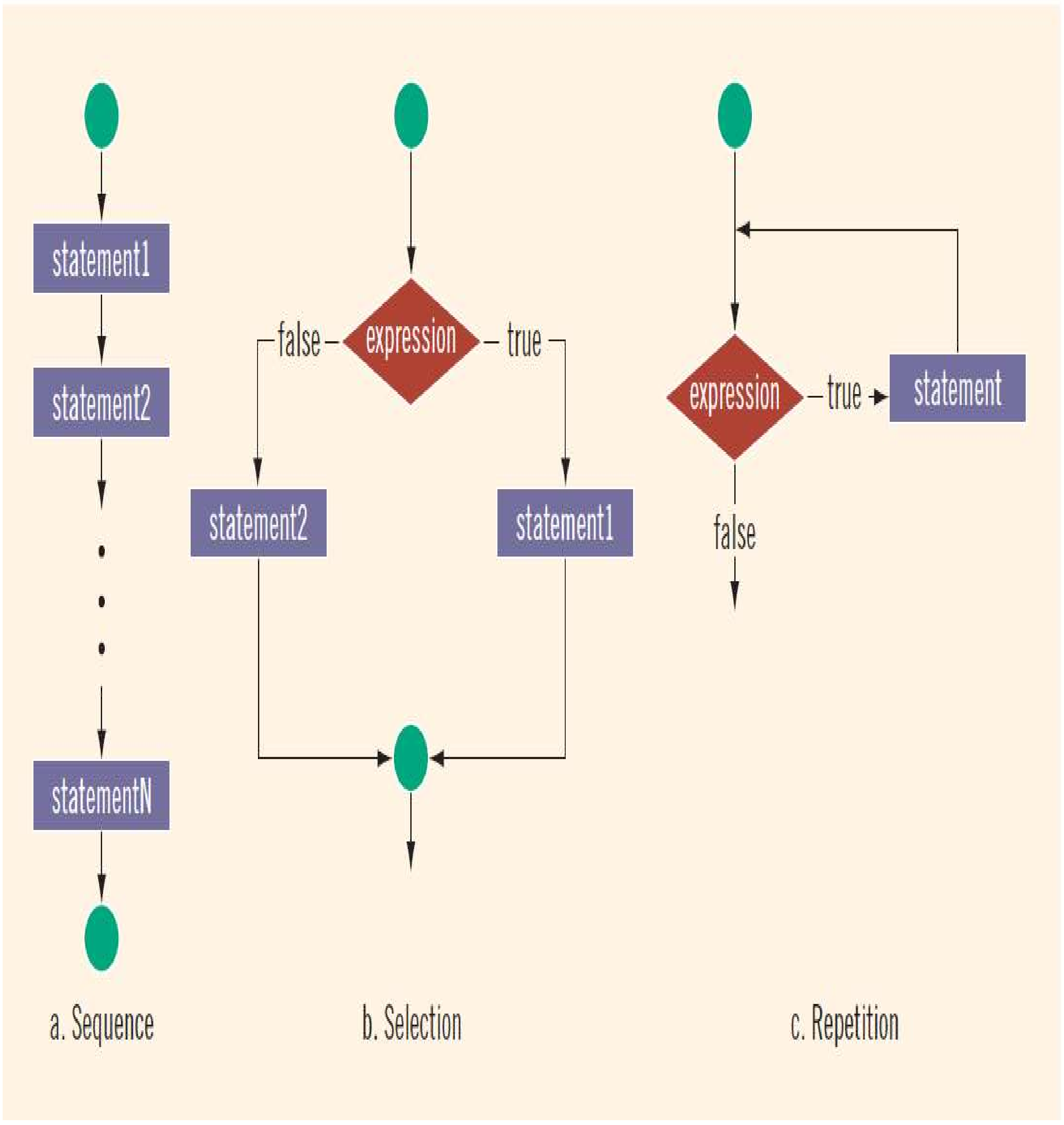


FIGURE 2: Flow of execution

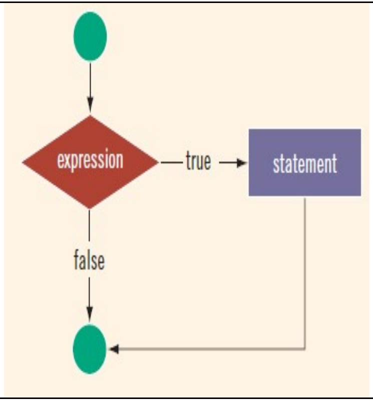
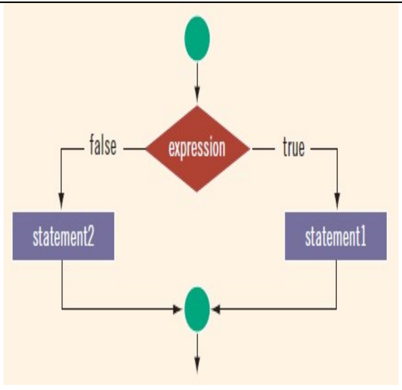
- ## Selection Control Structure

In C++, there are two selections or branch control structures**: if** statements and the **switch** structure.

One-Way Selection	Two-Way Selection	Nested if

if (expression)
  statement
	if (expression)
	  statement1
		if (score >= 90)
		  cout << "The grade is A.";

	else
	statement2
		else if (score >= 80)
		cout << "The grade is B.";
		else if (score >= 70)
		cout << "The grade is C.";
		else if (score >= 60)
		cout << "The grade is D.";
		else
		cout << "The grade is F." ;

if (score >= 60)
grade = 'P';
	if (score >= 60)
	grade = 'P';
	else
	grade = 'P';	

EX: c++ program to test if number is negative or positive, and if number is positive then it is compute its double.

```
#include <iostream>
using namespace std;
int main()
{       int x;
        cout << "Enter an integer: ";
        cin >> x;
        if (x == 0)
                cout << "x is 0" << endl;
        else if (x < 0)
                cout << "x is negative" << endl;
        else        {
                        cout << "x is positive";
                        cout << "and ten times x is: " << x * 10 << endl;}
        return 0;}
```

EX: c++ program to test if character is small letter, capital letter,digits or special character.

```
#include <iostream>
using namespace std;
int main()
```

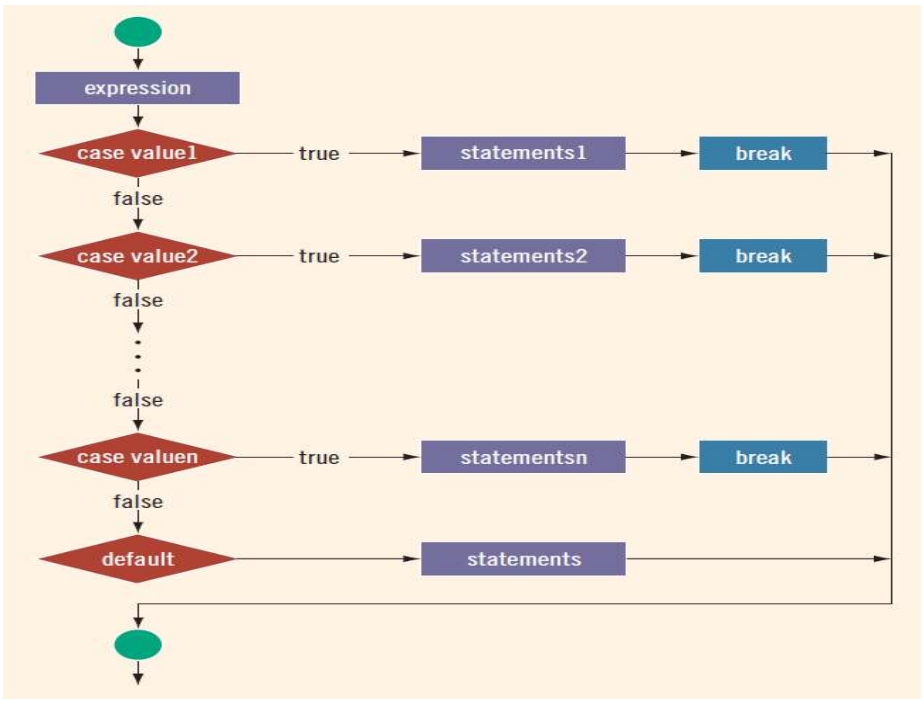
```
{       char x;
        cout << "Enter a character: ";
        cin >> x;
        if (x >= 'A' && x<='Z')
          cout << "The character is CL.";
        else if (x >= 'a' && x<='z')
          cout << "The character is SL.";
        else if (x >= '0' && x<='9')
          cout << "The character is D.";
        else
          cout << "it is special character." ;
        return 0;
        }
```

EX:c++ program to compute the maximum number between three numbers.

```
#include <iostream>
using namespace std;
int main()
{
        int x, y, z, max;
         cout<<"Enter any three numbers: ";
        cin>>x>>y>>z;
        max = x;
        if(y>max)
                max = y;
        if(z>max)
                 max = z;
        cout<<"\n"<<"The largest of "<<x<<", "<<y<<" and "<<z<<" is "<<max;
        return 0;
}
```

C++'s switch structure gives the computer the power to choose from among many alternatives. A general syntax of the **switch** statement is:

```
switch (expression)
{
case value1:
       statements1
       break;
case value2:
        statements2
       break;
.
.
.
case valuen:
        statementsn
        break;
default:
        statements
}
```



EX: C++ program to evaluate the grade of students.

```
#include <iostream>
using namespace std;
int main()
{        int score;
char grade;
cout << "Enter degree of student: ";
cin >> score;
switch (score / 10)
{
          case 0:
          case 1:
          case 2:
          case 3:
          case 4:
          case 5:
                    grade = 'F';
          break;

          case 6:
                    grade = 'D';
          break;
          case 7:
                    grade = 'C';
          break;
          case 8:
                    grade = 'B';
          break;
          case 9:
          case 10:
                    grade = 'A';
          break;
          default:
          cout << "Invalid test score." <<
endl;
}
          cout<<"grade="<<grade;
          return 0;
}
```

EX: C++ program that shows the appropriate age to vote.

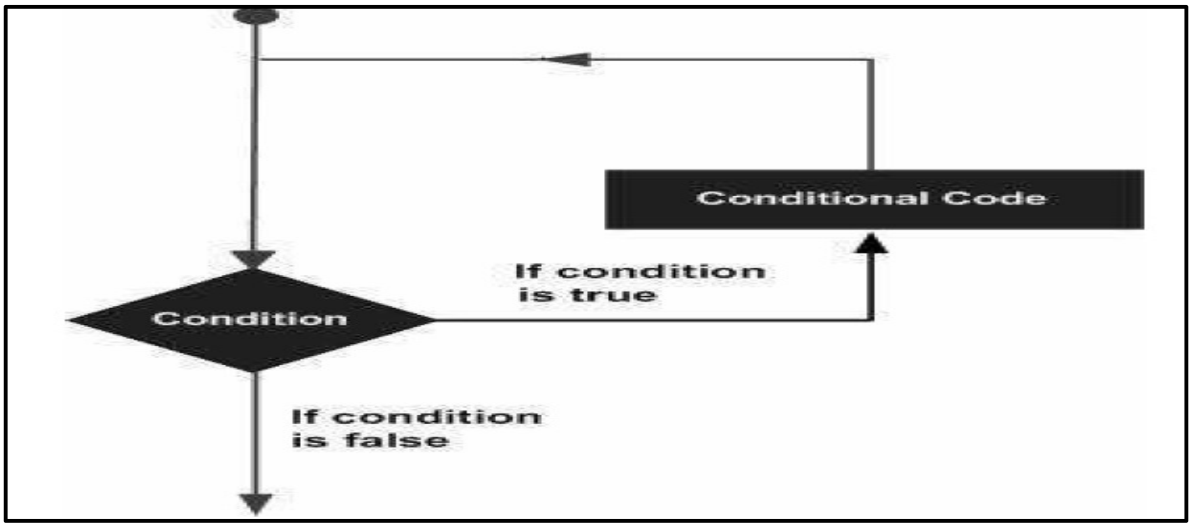
```
#include <iostream>
```

```
using namespace std;
int main()
{       int age;
        cout << "Enter the age: ";
        cin >> age;
        switch (age >= 18)
                {
                        case 1:
                                cout << "Old enough to be drafted." << endl;
                                cout << "Old enough to vote." << endl;
                        break;
                        case 0:
                                cout << "Not old enough to be drafted." << endl;
                                cout << "Not old enough to vote." << endl;
                }

return 0;
}
```

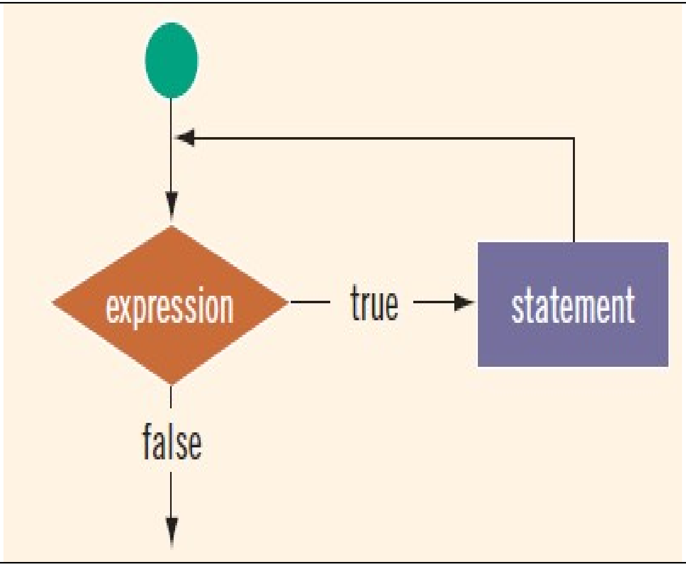
- **Lopping Control Structure**

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

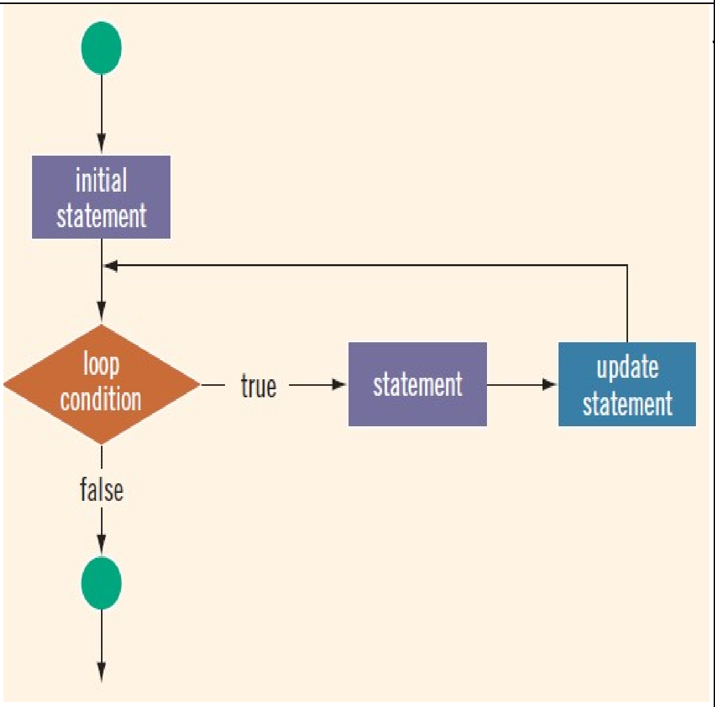


C++ programming language provides the following type of loops to handle looping requirements.

while loop	while (expression)
	    statement
		i = 0;
		while (i <= 20)
		{
		  cout << i << " ";
		  i = i + 1;
		}

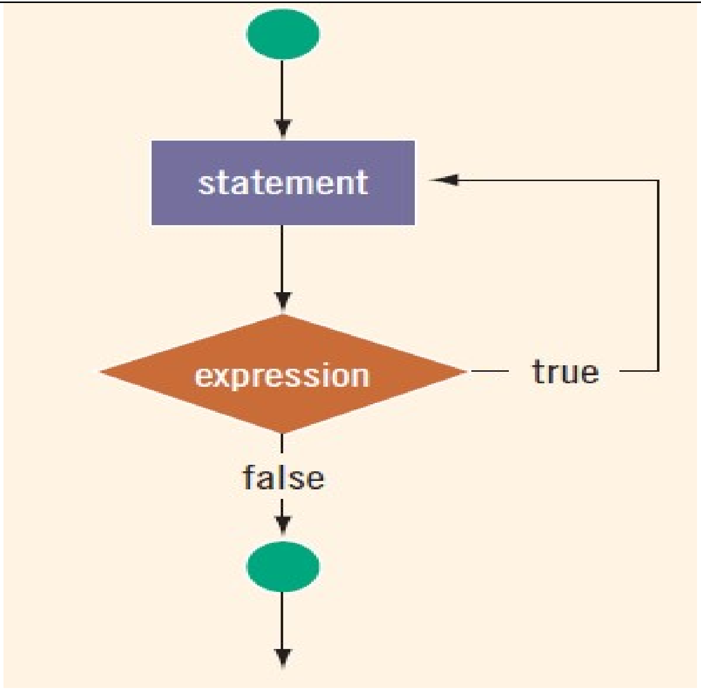
		i = 0;
		while (i <= 20)
		{
		  cout << i << " ";
		  i = i + 2;
		}

For loop	for (initial statement; loop condition; update statement)
	    statement
		for (i = 0; i < 20; i++)
		   cout << i << " ";

	 initial statement
	loop condition
	true
	statement
	update statement
	false
		for (i = 0; i < 20; i+=2)
		   cout << i << " ";

	do	i = 0;

Do while	statement
	while (expression);
		do
		{
		cout << i << " ";
		i = i + 1;
		}
		while (i <= 20);

		i = 0;
		do
		{
		cout << i << " ";
		i = i + 2;
		}
		while (i <= 20);

EX: C++ program to Calculate $1^2 + 2^2 + 3^2 + 4^2 + 5^2 + 6^2 \ldots + n^2$ series

```
#include <iostream>
using namespace std;
int main()
{       int i,a,n,sum=0;
        cout<<"enter the limit of series";
        cin>>n;
        cout<<endl;
        for(i=1;i<=n;i++)
            sum+=i*I;
        cout<<"Sum: "<<sum;
        return 0;                }
```

EX: C++ program to Calculate $1^2 / 2^2 + 2^2 / 3^2 + 3^2 / 4^2 + \ldots + n^2 / (n+1)^2$ series.

```
#include <iostream>
using namespace std;
int main()
```

```
{       int i,a,n,sum=0;
   cout<<"enter the limit of series";
        cin>>n;
        cout<<endl;
   for(i=1;i<=n;i++)
      sum+=(i*i)/(i+1)*(i+1);
   cout<<"Sum: "<<sum;
return 0; }
```

EX: C++ program to Calculate (1) + (1+2) + (1+2+3) + (1+2+3+4) + ... + (1+2+3+4+...+n) series

```
#include <iostream>
using namespace std;
int main()
{  int i,j,n,sum=0;
   cout<<"enter the limit of series";
        cin>>n;
   for(i=1;i<=n;i++)
        for( j=1;j<=i;j++)
             sum+=j;
   cout<<"Sum: "<<sum;
return 0;}
```

EX: C++ program finds the sum of positive numbers the user enters. The loop ends if the user enters a negative number.

```
#include <iostream>
using namespace std;
int main() {
   int num;
   int sum = 0;
   cout << "Enter a number: ";
   cin >> num;
   while (num >= 0) {
      sum += num;
      cout << "Enter a number: ";
      cin >> num;    }

   cout << "\nThe sum is " << sum << endl;
   return 0; }
```

```
do {
       sum += num;
       cout << "Enter a number: ";
       cin >> num;     }
   while (num >= 0);
```

EX: C++ program to Calculate $1^0 + 2^1 + 3^2 + 4^3 + 5^4 + 6^5 ... + n^{n-1}$ series

```
#include <iostream>
using namespace std;
int main()
{       int i,j,n,sum=1,f;
        cout<<"enter the limit of series";
        cin>>n;
        cout<<endl;
        for(i=2;i<=n;i++)
           {f=1;
           for(j=1;j<=i-1;j++)
              f*=i;
           sum+=f;}
        cout<<"Sum: "<<sum;
        return 0;               }
```

EX: C++ program to show the following pattern:

```
1
2  2
3  3  3
4  4  4  4
```

```
#include <iostream>
using namespace std;
int main()
{       int i,j,n;
        cout<<"enter the limit of pattern";
        cin>>n;
        cout<<endl;
        for(i=1;i<=n;i++)
           {   for(j=1;j<=i;j++)
                   cout<<i<<" ";
               cout<<endl;      }

        return 0;               }
```

EX: C++ program to show the following pattern:

```
1  1  1  1
2  2  1
3  3
4
```

```
#include <iostream>
using namespace std;
int main()
{       int i,j,n;
        cout<<"enter the limit of pattern";
        cin>>n;
        cout<<endl;
        for(i=1;i<=n;i++)
           {
                for(j=1;j<=n-i+1;j++)
                   cout<<i<<" ";
           cout<<endl;
           }

        return 0;                }
```

EX: C++ program to show the following pattern:

```
1   2  3  4
5   6  7
8   9
10
```

```
#include <iostream>
using namespace std;
int main()
{       int i,j,n,a=1;
        cout<<"enter the limit of pattern";
        cin>>n;
        cout<<endl;
        for(i=1;i<=n;i++)
           {
                for(j=1;j<=n-i+1;j++)
                   cout<<a++<<" ";
           cout<<endl;
           }

        return 0;                }
```

EX: C++ program to show the following pattern:

```
a  b  c  d
e  f  g
h  i
j
```

```
#include <iostream>
using namespace std;
int main()
{       int i,j,n;
   char a='a';
        cout<<"enter the limit of pattern";
        cin>>n;
        cout<<endl;
        for(i=1;i<=n;i++)
           {
                for(j=1;j<=n-i+1;j++)
                   cout<<a++<<" ";
           cout<<endl;
           }
        return 0;                }
```

EX: C++ program to show the following pattern:

```
    a
  b b b
c c c c c
```

```
#include <iostream>
using namespace std;
int main()
{       int i,j,n,a=-1;
        char l='a';
        cout<<"enter the limit of pattern";
        cin>>n;
        cout<<endl;
        for(i=1;i<=n;i++)
           {     for(j=n-i;j>=1;j--)
                        cout<<"  ";
                 a=a+2;
                for(j=1;j<=a;j++)
                       cout<<l<<" ";
                 cout<<endl;
                l++;             }

        return 0;                }
```

EX: C++ program to show the following pattern:

```
a a a a a
  b b b
    c
```

```
#include <iostream>
using namespace std;
int main()
{       int i,j,n,a;
        char l='a';
        cout<<"enter the limit of pattern";
        cin>>n;
        a=n;
        cout<<endl;

for(i=1;i<=n;i++)
           {    for(j=1;j<=i;j++)
                        cout<<"  ";
                for(j=1;j<=a;j++)
                                cout<<l<<" ";
                a-=2;
                cout<<endl;
                l++;            }
        return 0;                }
```

EX: C++ program to show the following pattern:

```
         1
       2  2
     3  3  3
   4  4  4  4
```

```
#include <iostream>
using namespace std;
int main()
{       int i,j,n,a;
        cout<<"enter the limit of pattern";
        cin>>n;
        a=n-1;
        cout<<endl;
        for(i=1;i<=n;i++)
           {
                for(j=1;j<=a;j++)
                   cout<<"  ";
           a--;
           for(j=1;j<=i;j++)
                   cout<<i<<" ";
           cout<<endl;}
        return 0;                }
```