Chapter 1

First steps in C++

Introduction

The general form to write a program in C++ language its:

1 Complier definition needed in programs,
2 define the variable which we (needed) used in the program,
3 program body.
And the constant form in each program is

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



	system("pause");


}
```

I=input , o=output , that's mean in this program there is input and output variable

Start main program and the end of program

Using this statement to show run windows to input and output the result of programs

For example

```
#include <iostream>
int main()
{
	using namespace std;
cout <<" Hello my dear student "
	system("pause");
}
```

Data types
We can manipulate in C++ programs two different categories of types:
• built-in types which are defined in the C++ compiler itself;
• class type which are defined in C++ source code from the built-in types.

We will focuse on four different built-in types:
• bool is a boolean value (true / false);
• int is an integer value (−2147483648 to 2147483647);
• double is a floating-point value (precision goes from 2.2250738585072014× 10−308 to 1.7976931348623157× 10308) ;
• char is a character (letter, digit, punctuation, etc.)
Beside the four built-in types presented above, other ones exist. The main idea behind this large number of different types is to allow the programmer to control precisely the efficiency in term of memory usage and speed of its programs considering his needs.
For instance, the unsigned int encode an integer value between 0 and 4294967295.
Both **int** and unsigned **int** use four bytes of memory on a x86 CPU. If we need to store a large number of smaller integer values, we can use the short type (or unsigned short), which takes only two bytes.
For the floating point values, the float type is less precise but takes less memory.
Also, computation goes faster with this type.

A simple example of variable manipulation

```
int main()
{
int i, j;
i = 4;
j = 12 * i + 5;
}
```

int main( ) is by convention the declaration of the part of the program which is run when the program starts, we will come back to this syntax later ;
int i, j; declares two variables of type int, called respectively i and j. It reserves two areas in the memory, and name them so that we can refere to them later in the program. The name of variables are called their identifiers ; i = 4; copies the value 4 in the area of the memory called i ; j = 12 * i + 5; reads the value in thfsdae area called i, multiplies it by 12, adds 5, and copies the result to the area of memory called j .
We have here made arithmetic operations between variables (i and j) and lit-eral constants (12 and 5). Variable types are defined in their declarations; constant types are defined by the syntax itself. Basically, an int constant is a number with no dot; a double constant is a number with a dot, a bool con-stant is either true or false, and a char constant is a character between ‘‘(for example “char c = ’z’;”).

For floating point constant, we can use the e notation for powers of ten. For
example x = 1.34e-3 makes 0.00134 in x.

Variable naming conventions

A variable identifier can be composed of letters, digits, and underscore character
'_'. It must begin with either a letter or an underscore.
Usually, one concatenate words to build long identifier either using the under-score character '-' as a space, or by using upper caps for first letter of each word (except the first letter) :
int numberOfCars;
double first_derivative;
We will reserve identifiers starting with an upper caps for our class names.

Streams, include files

Beside the built-in types, the C++ compiler is often accompanied with lot of files containing predefined types. The main one we will use in our example programs is the stream type.
To use it, we need to indicate to the compiler to include in our source file another file called iostream (where this class type is defined).

```
#include <iostream>
int main()
{
        using namespace std;
int k;
k = 14;

k = k + 4;
k = k * 2;
cout << k << '\n';
        system("pause");
}
```

The variable **cout** is defined in the included file and is of type ostream. The only thing we need to know for now is that we can display a variable of a built-in type by using the << operator.
We can also read values with **cin** and the >> operator. The following program gets two float values from the user and prints the ratio.

```
#include <iostream>
int main()
{
        using namespace std;
double x, y;
cin >> x >> y;
cout << x / y << '\n';
system("pause")
}
```

List of operators

The standard mathematic symbols can be used to write arithmetic expressions :

Symbol	**Function**
+	addition
-	subtraction
*	multiplication
/	division
%	reminder

The % computes the reminder of an integer; \ division (for instance 17 % 5 has the value 2) ; the result is well defined only if both operands are positive.
All those operators but % can be used with either integer or floating point operands.

Operators depend on the types of the operands

Internally, each symbol corresponds to different operations, depending with the type of the operands:

```
#include <iostream>
int main() {
using namespace std;
cout << 15 / 4 << ' ' << 15.0 / 4.0 << '\n';
system("pause");
}
displays 3   3.75.
```

Implicit conversions

Basically operators are defined for two operands of the same type. The compiler can automatically convert a numerical type into another one so that the operator

```
#include <iostream>
int main()
{
        using namespace std;
cout << 3 + 4.3 << '\n';
        system("pause");
}
result is 7.3
```

The if statement

The **if** statement executes a part of a program only **if** a given condition is true.

if(condition)
<statement to execute if the condition is true>
or
if(condition)
<statement to execute if the condition is true>
else
<statement to execute if the condition is false>
A statement here is either a part of a program enclosed in { }, or a single line terminated with a ’;’. For example:

```
#include <iostream>
int main()
{
using namespace std;
int n;
cin >> n;
if(n < 0) n = 0;
else
n = 2 * n;
n = n - 1;
cout << n << ’\n’;
system("pause");
}
```

Example) write a program to read 3 integer No., then find the Maximum No.

```
#include <iostream>
int main()
{
using namespace std;
int x, y, z, max;
cin >> x>>y>>z;
max=x
if ( max <y ) max=y;
if (max<z ) max=z;
cout << “max=”<<max;
        system("pause");
}
```

Example) write a program to solve the equation $\mathbf{ax^2+bx+c=0}$.

```
#include <iostream>
int main();
{
using namespace std;
double a,b,c,x1,x2,r;
cin >> a>>b>>c;
r=b*b-4*a*c;
if ( R<0 )
cout<<" The equation has no roots"
else
{
x1=(-b+Sqrt(r))/(2*a);
x2=(-b-Sqrt(r))/(2*a);
cout<<x1<<x2;
}
        system("pause");

}
```

The (for) statement

The (**for**) statement repeats the execution of a part of a program.
for (initialization; condition; increment)
<statement to repeat>
For example, to display all positive integers strictly smaller than a value given by the user:

```
#include <iostream>
int main() {
using namespace std;
int n, k;
cin >> n;
for(k = 0; k < n; k++) cout << k << '\n';
        system("pause");
}
```

Note that we have declared two variables of type int on the same line. The k++ notation means in that context simply k = k+1.
The (for) can be used to make more complex loops:

Array (Matrix)

First: One dimensional (array)s:

int array name[size of array(integer)];
double array name[size of array(integer)];
char array name[size of array(integer)];

Ex.1) write a program to read an array of size n then print it.
Solu.)

```
#include <iostream>
int main() {
using namespace std;
int n,k, x[100];
cin << n;
for(k = 1; k < n+1; k++)
cin<<x[k];
for(k = 1; k < n+1; k++)
cout>>x[k];
system("pause");
}
```

Ex.2) write a program to read an array of size n for integers then find the maximum no.
Solu.)

```
#include <iostream>
int main() {
using namespace std;
int n,k, max,x[100];
cin <<n;
for(k = 1; k < n+1; k++)
cin<<x[k];
max=x[1];
for(k = 2; k < n+1; k++)
if(max<x[k]) max=x[k];
cout>>the max.=”>>max;
system("pause");
}
```

Ex.3) Write a program to read an array of size n for integers then find the location of minimum no.
Solu.)

```
#include <iostream>
int main()
 {
using namespace std;
int n,k, min,l,x[100];
cin >> n;
for(k = 1; k < n+1; k++)
cin>>x[k];
min=x[1];l=1;
for(k = 2; k < n+1; k++)
if(min>x[k])
{ min=x[k];
l=k;}
cout>>the location of min.=”>>l;
system("pause");
}
```

Ex.4) write a program to read an array of size n for integers then sort it in ascending order.
Solu.)

```
#include <iostream>
int main()
 {
using namespace std;
int n,k,i, x[100];
cin <<n;
for(k = 1; k < n+1; k++)
cin<<x[k];
for(i=1;i<n;i++)
for(k = i+1; k < n+1; k++)
if(x[i]>x[k])
{ t=x[i];
x[i]=x[k];
x[k]=t;}
for(k = 1; k < n+1; k++)
cout>>x[k];
system("pause");
}
```

Ex.5) write a program to read an array of size n for integers then find the no. of odds no. in it.
Solu.)

```
#include <iostream>
int main()
 {
using namespace std;
int n, k, oddn, x[100];
cin <<n;
for(k = 1; k < n+1; k++)
cin<<x[k]; oddn=0;
for(i=1;i<n;i++)
if(x[i]%2==1) oddn= oddn+1;
cout>>" the no. of odd no. in matrix X=">>oddn;
system("pause");
}
```

H.W
Ex.1) write a program to read an array of size n for integers then find the no. of prime no. in it.
Ex.2) write a program to read an integer no. then convert it octal system no.

Second: Two dimensional arrays

int array name[no. raw(integer)][no. of colum(integer)] ;
double array name[no. raw(integer)][no. of colum(integer)] ;
char array name[no. raw(integer)][no. of colum(integer)] ;

Ex.1) write aprogram to read an array of size n by m then print it.
Solu.)

```
#include <iostream>
int main() {
using namespace std;
int n,m,i,j, x[100][100];
cin >> n>>m;
for(i = 1; i < n+1; i++)
for(j = 1; j< m+1; j++)
cin<<x[i][j];
for(i = 1; i < n+1; i++)
{
for(j = 1; j< m+1; j++)
cout>>x[i][j];cout<<endl;
}
system("pause");
}
```

H.W

Ex.1) write a program to read an array of size n by m for integers then find the location of maximum no.

Ex.2) write a program to read an array of size n by m for integers then change the location of max. with min. no.

Ex.3) write a program to read an array of size n by m for integers then sort each Colum in ascending order.