

What is Programming?

Computer Programming is defined in wiki books as:

“Computer programming is the craft of writing useful, maintainable, and extensible source code which can be interpreted or compiled by a computing system to perform a meaningful task.”

Computers are very helpful machines, but we need to program them before using. They can't do anything on their own. In order to get a task performed by a computer *“you (or someone else) have to tell it exactly - in excruciating detail - what to do. Such a description of "what to do" is called a program, and programming is the activity of writing and testing such programs.”*

In order to give instructions to the computer, we need to speak to them in a language they understand. Once we are able to do that we can write instructions for computer. Just like any common language computer languages also have some vocabulary and grammar. While using those languages it is essential to comply with the language grammar and use correct vocabulary. Otherwise computer won't understand us; it may end up doing something wrong or may get angry at us and throw errors and warnings.

Details about evolution of computer languages can be seen in the book.

Ungraded Assignment:

- i. Find differences between Programmers and Users
- ii. Create a list of modern programming languages with their preferred targets

Integrated Development Environment (IDE)

In order to write programs using C++ language you need C++ compiler and a text editor. C++ compiler converts the source code to computer understandable form. Text editor is required to type the source code. Often there are computer programs that are combination of compiler and text editor. These programs are referred as Integrated Development Environment (IDE) (they may be free of cost or paid). For this course it is recommend to use an IDE called **Code::Blocks**. It is open source free of cost application targeting Windows, Mac and Linux environment. Further details about code blocks can be obtained from <http://www.codeblocks.org/>

Your First Code

les_01_code_01.cpp

```
1.  #include<iostream>
2.  using namespace std;
3.
4.  int main()
5.  {
6.      cout<<"Hello world!";
7.      return 0;
8.  }
```

Let us traverse the code line by line

Line No	Description
1	Pre Processor Directive to make available facilities from file iostream.h into this program iostream is used when it is required to show some output on monitor, or take input from the keyboard. This directive will essentially be present in all the codes we will develop in this course
2	namespace is an advanced idea, for now just consider it as part of library This line will also be present in all codes
3	A blank space; it has no effect on processing. Used just to make the code look tidy
4	this is the main function definition (main is a name and reserved keyword for C++) Every C++ program must have a function called main to tell it where to start. A function is a named sequence of instructions for computer to execute in order in which they are written. A function has four parts: i. A return type (here it is int) ii. Name (main here) iii. Parameter list (empty here) iv. Function body, enclosed in a set of curly braces { }, which lists the actions called statements that the function is to perform.
5	Opening brace { of the main function.
6	cout (see out) it's the standard output stream (print to console). followed by the symbol << this is put to operator (stream insertion operator) which indicates what ever follows this will be put to standard output (monitor). Then we have "Hello world!" this is string, a fundamental data type in C++ and it is recognized with " " . then we have a semicolon ; this is the end of the statement. Every C++ statement should terminate with a semicolon, this is how compiler recognize that a statement is completed.
7	return 0; this is the return value of main (for time being just consider this is the standard procedure until we discuss functions). A zero (0) returned by main() indicates the program terminated successfully.
8	Closing brace } of the main function.

Understanding endl

les_01_code_02.cpp

```

1.  #include<iostream>
2.  using namespace std;
3.
4.  int main()
5.  {
6.      cout<<"Hello"<<endl<<"world!";
7.      return 0;
8.  }
```

Exercise

Predict output of the following statements

Statement	Output
<code>cout << "Hello" world!" << endl;</code>	
<code>cout << "Hello"; world!" << endl;</code>	
<code>cout << "Hello" ; cout <<" world!" << endl;</code>	

Ungraded Task
















Write a program that prints the following

Hello Your Name!

Welcome to C++ Programming

Bonus Material

IEEE Spectrum Top Ten Programming languages

Language Rank	Types	Spectrum Ranking
1. Python	 	100.0
2. C	  	99.7
3. Java	  	99.5
4. C++	  	97.1
5. C#	  	87.7
6. R		87.7
7. JavaScript	 	85.6
8. PHP		81.2
9. Go	 	75.1
10. Swift	 	73.7

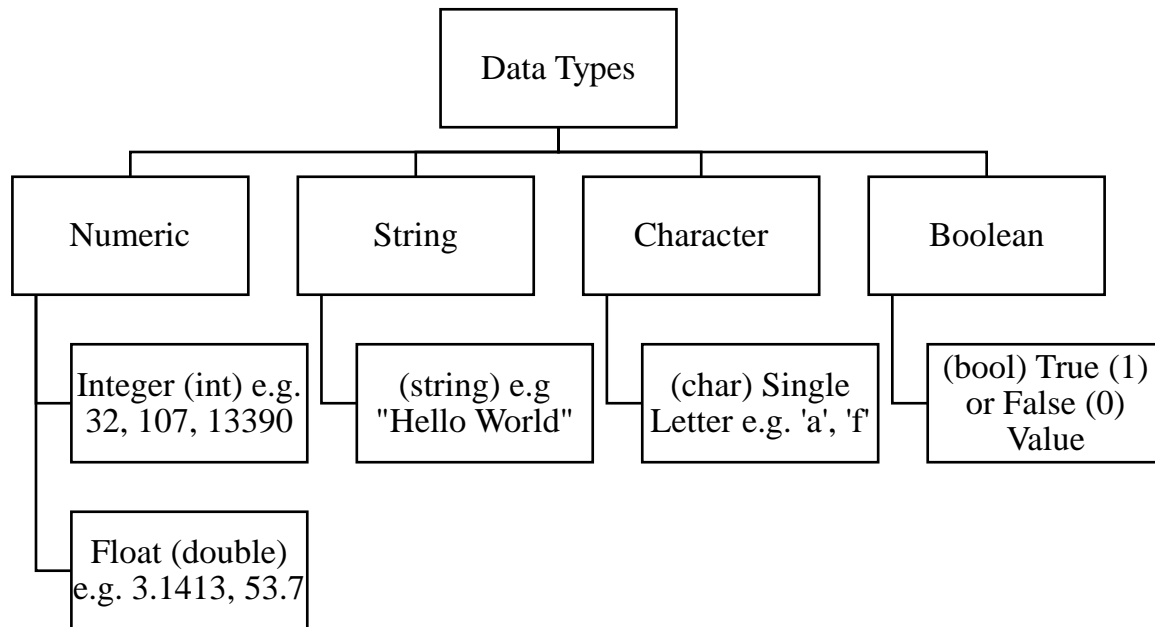
<https://spectrum.ieee.org/computing/software/the-2017-top-programming-languages>

Understanding Data Types

Data types, defines the proper use of an identifier and expression. An expression is a sequence of operators and their operands that specifies a computation. Various data items with symbolic names in C++ is called as identifiers. Following data items are identifier in C++

- Names of functions
- Names of arrays
- Names of variables
- Names of classes

Data types are important to understand because certain operations can only be performed on certain data types. Following chart shows the broad categories of data types available in C++



Numeric Data Types

Whole Numbers or Integers

- `int` can store number hold upto 4 bytes can store number from -2,000,000,000 to +2,000,000,000 (approximately)
- `unsigned int` can hold up to 4 bytes (only positive numbers) but that doubles the positive range, ranges between 0 – 4,000,000,000

Data Types for floating point numbers

- `double` Size is 8 bytes +-1.7 e +- 308;
- `float` 4 bytes

Data Types for Alpha Numeric

`string`

a string is zero or more characters surrounded by double quotes. E.g. “Hello World!”, “100”,

`char`

Only a single character can be stored in a variable of type char, char values are stored as an integer, based on ASCII value of character e.g. 'a'

char was important data type when systems had less memory, now a days memory in the systems is ample so char is not much useful now.

Boolean Data

`bool`

bool is another data type, it holds true (1) or false (0) values.

Primitive Built-in Types

C++ offer the programmer a rich assortment of built-in as well as user defined data types. Following table lists down seven basic C++ data types:

Type	Keyword
Boolean	bool
Character	char
Integer	int
Floating point	float
Double floating point	double
Valueless	void
Wide character	wchar_t

Several of the basic types can be modified using one or more of these type modifiers:

- signed
- unsigned
- short
- long

The following table shows the variable type, how much memory it takes to store the value in memory, and what is maximum and minimum value which can be stored in such type of variables.

Type	Typical Bit Width	Typical Range
char	1byte	-127 to 127 or 0 to 255

unsigned char	1byte	0 to 255
signed char	1byte	-127 to 127
int	4bytes	-2147483648 to 2147483647
unsigned int	4bytes	0 to 4294967295
signed int	4bytes	-2147483648 to 2147483647
short int	2bytes	-32768 to 32767
unsigned short int	Range	0 to 65,535
signed short int	Range	-32768 to 32767
long int	4bytes	-2,147,483,647 to 2,147,483,647
signed long int	4bytes	same as long int
unsigned long int	4bytes	0 to 4,294,967,295
float	4bytes	+/- 3.4e +/- 38 (~7 digits)
double	8bytes	+/- 1.7e +/- 308 (~15 digits)
long double	8bytes	+/- 1.7e +/- 308 (~15 digits)
wchar_t	2 or 4 bytes	1 wide character

Variables

Variables are named objects with a specific type. (*Object is a region of memory with a type.*) Variables can be used to store data of a certain type which can later be used in the program. A variable must be declared using a legal keyword (listed in table above) and a legal name. Following are the rules to assign variable names:

- Variable name must start with a letter or _ (underscore)
- May contains letter, numbers and the underscore character
- Examples: salary, firstName, name3, first_name, _newVal
- Illegal : 3name, first name, new, class, struct, while, for, my@val
- Uppercase and lower case are distinct

Programs to Understand Variables and Data Types

les_01_code_03.cpp

```
1.    #include<iostream>
```

```
2.
3.     using namespace std;
4.
5.     int main()
6.     {
7.         int num1;
8.         num1 = 2;
9.
10.        char sym;
11.        sym = '@';
12.
13.        string name;
14.        name = "Leo Tolstoy";
15.
16.        cout<<"You inserted "<<num1<<" as num1"<<endl;
17.        cout<<"You inserted "<<sym<<" as sym"<<endl;
18.        cout<<"You inserted "<<name<<" as name"<<endl;
19.
20.        return 0;
21.    }
```

Output

```
You inserted 2 as num1
You inserted @ as sym
You inserted Leo Tolstoy as name
```

les_01_code_04.cpp

```
1. #include<iostream>
2.
3. using namespace std;
4.
5. int main()
6. {
7.     int num2 = 2.5; // Initialization statement
8.     cout<<num2;
9.
10.        return 0;
11.    }
```

Output

Declaring Multiple Variables of same type

les_01_code_05.cpp

```
1. #include<iostream>
2.
3. using namespace std;
4. int main ()
5. {
6.     int num1, num2, num3;
7.     int num4, num5 = 10, num6;
8.     int num7 = 20, num8 = 30, num9 = 40;
9.
10.     return 0;
11. }
```

Common Mistakes

les_01_code_06.cpp

```
1. #include<iostream>
2.
3. using namespace std;
4.
5. int main()
6. {
7.     int num1 = 7;
8.     cout<<Num1;
9.
10.     cout<<num2;
11.     int num2;
12.
13.     strings name;
14.
15.     return 0;
16. }
```


Interactive Computing

Often it is required in a program to take input from the user in run time. This idea of interacting with the user is called interactive computing. This can be done in C++ through standard input stream object **cin** (of namespace **std**) and the stream extraction operator **>>** (also called get from operator). The following code will help you to understand the idea.

les_01_code_07.cpp

```
1.  #include<iostream>
2.
3.  using namespace std;
4.
5.  int main()
6.  {
7.      int num1;
8.      cout<<"Please enter the value of num1 : ";
9.      cin>>num1;
10.     cout<<"You Entered : "<<num1;
11.
12.     return 0;
13. }
```

Output

```
Please enter the value of num1 : 100
You Entered : 100
Process returned 0 (0x0)   execution time : 3.482 s
Press any key to continue.
```

Taking Multiple Inputs

les_01_code_08.cpp

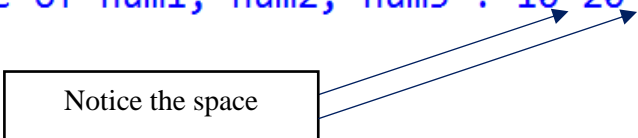
```
1.  #include<iostream>
2.
3.  using namespace std;
4.
5.  int main()
6.  {
7.      int num1, num2, num3;
8.      cout<<"Please enter the value of num1, num2, num3 : ";
9.      cin>>num1>>num2>>num3;
10.     cout<<"You Entered : num1 "<<num1<<endl;
11.     cout<<"You Entered : num2 "<<num2<<endl;
12.     cout<<"You Entered : num3 "<<num3<<endl;
13.     return 0;
14. }
```

Possible Outputs

- i. Entering values separated by space (whitespace is delimiter for cin)

Please enter the value of num1, num2, num3 : 10 20 30

Notice the space



Please enter the value of num1, num2, num3 : 10 20 30

You Entered : num1 10

You Entered : num2 20

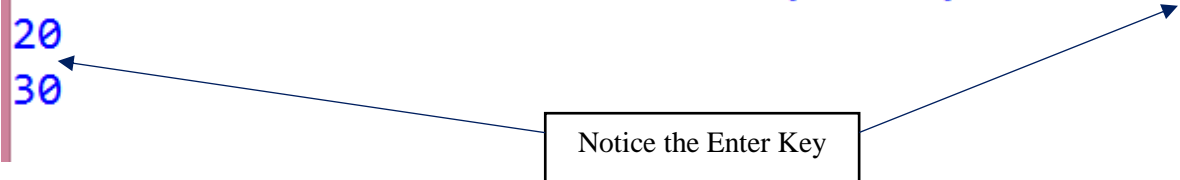
You Entered : num3 30

Process returned 0 (0x0) execution time : 131.052 s

Press any key to continue.

- ii. Entering Values separated by Return Key

Please enter the value of num1, num2, num3 : 10
20
30



Notice the Enter Key

Please enter the value of num1, num2, num3 : 10

20

30

You Entered : num1 10

You Entered : num2 20

You Entered : num3 30

Process returned 0 (0x0) execution time : 126.293 s

Press any key to continue.

Careful with Strings

les_01_code_09.cpp

```
1.    #include<iostream>
2.
3.    using namespace std;
4.
5.    int main()
6.    {
7.        string name;
8.        cout<<"Please enter the name : ";
9.        cin>>name;
10.       cout<<"Name you entered is : "<<name;
11.       return 0;
12.    }
```

Output

```
Please enter the name : Leo Tolstoy
Please enter the name : Leo Tolstoy
Name you entered is : Leo
Process returned 0 (0x0)   execution time : 43.960 s
Press any key to continue.
```

Note that whitespace is delimiter for the `cin` so everything you typed after `Leo` is lost

Remedy

Use `getline()` function to input strings

les_01_code_10.cpp

```
1.    #include<iostream>
2.
3.    using namespace std;
4.
5.    int main()
6.    {
7.        string name;
8.        cout<<"Please enter the name : ";
9.        getline(cin,name);
10.       cout<<"Name you entered is : "<<name;
11.
12.       return 0;
13.    }
```

Output

Run 1

```
Please enter the name : Leo Tolstoy
Name you entered is : Leo Tolstoy
Process returned 0 (0x0)   execution time : 7.403 s
Press any key to continue.
```

Run 2

```
Please enter the name : Leo           Tolstoy
Name you entered is : Leo           Tolstoy
Process returned 0 (0x0)   execution time : 20.486 s
Press any key to continue.
```

Beware that getline can only be used to input strings

Exercise

Write a program that stores speed of light, Planck's constant, charge on electron, Avagadro's number, your age, your full name, your department, your cgpa in appropriate variables and print them on console.

Re implement the same program, this time making it interactive by involving user input.