GeeksMan

We code therefore
we are

CODE FOUNDATION

LESSON 1

A Brief Introduction to C++:

C++ is a general-purpose programming language created as an extension of the C programming language. It has imperative, object-oriented and generic programming features and is widely used for competitive programming.

Preparing to program:

To solve any problem we need to take various steps. We must first describe the problem, then only we can find it's solution. If the problem is not clear to us, we can't think of it's solution. Once the problem statement is clear, we can start thinking about it's solution. We can apply this approach in the field of programming also.
One should always follow these steps while writing the programs:

1. **Determine the aim of your program:**
   You should be very clear about the purpose of writing your program.

2. **Think about the methods to use in writing the program:**
   Think about any formula you can apply if you are solving a mathematical problem.

3. **Create the program:**
   Choose a programming language like C++, C, Python. Write your program in any programming language.You can use Sapphire Engine to write your program. Sapphire Engine is an online compiler and debugging platform which allows you to compile your program and execute it online.

4. **Run the program to see the results.**
   Simply click on the run button to check the results. If your results are correct, it means your approach towards the solution of the problem is correct.

Hello World program:

In this section we will study about the Hello World program.
The "Hello World" program is the first step towards learning any new programming language. It is one of the simplest programs that you will learn.

The problem statement is : **Print “Hello World” on the screen.**
The links for Sapphire engine and ideone are provided below. You can also refer to these links for the code.

Program Link(Sapphire Engine): **https://sapphireengine.com/@/gwutw0**
Program Link(IDEONE): **https://ideone.com/ckaus/helloworld**

Now, let’s have a look at our first program:

```
#include <iostream>

using namespace std;

int main() {

      //printing Hello World onto the screen
      cout << "Hello World" << endl;
      return 0;
  }
```

Standard Output

```
Hello World
```

Now let’s have a look at all the statements of our program and try to understand them one by one.

1. Comments

In C++, any line starting with `//` or enclosed within `/*...*/` is a comment. Comments are intended for a person reading the program to better understand the functionality of the program. If you want to refer to the code later on, comments help you in understanding your code. Comments are completely ignored by the compilers. There are two ways of writing comments in C++: -

- // a single-line comment
- /*

```
a multi-line comment

*/
```

2. `#include <iostream>`

`#include` is a preprocessor directive which is used to include files in our program. This statement is used in the beginning of the C++ program. The above statement includes the contents of the **iostream** header file in our program.

iostream stands for input output stream.This allows us to use `cout` in our program to print output onto the screen.

For now, just remember that we need to use `#include <iostream>` to use a `cout` statement that allows us to print output on the screen.

3. `int main() {...}`

A valid C++ program must have a `main()` function. The curly braces indicate the start and the end of the function.

The execution of every C++ program begins from the main function.

4. `cout << "Hello World";`

`cout` is used to display the output onto the screen. Here, `cout` displays the content inside the quotation marks on the screen. `cout` must be followed by << followed by anything you want to display.

Note: `;` is used to indicate the end of a statement.

5. `return 0;`

The `return 0;` statement is the "Exit status" of the program. In simple terms, the program ends with this statement.

6. **using namespace std;**

In every scope each name represents a unique entity. For now you can think of scope as everything between a pair of curly braces. In a scope there can't be more than one entity having the same name but can use **namespace** to do so.

A **namespace** is used as additional information to differentiate between entities having the same names. In essence, a namespace defines a scope. So, when we run a program to print something, "**using namespace std**" tells the compiler that if you find something that is not declared in the current scope go and check **std**. Computers need to know the namespace of cout, cin statements, that's why we use "using namespace std;"

Even if your compiler accepts "void main()" avoid it, or risk being considered ignorant by C and C++ programmers.

In C++, main() need not contain an explicit return statement. In that case, the value returned is 0, meaning successful execution.

Variables and Data Types:

In this lesson we will study about variables and various data types.

Variables:

Almost everything in this world has a name. But why do we need names? Having a name helps in identifying and describing a thing. For e.g you call your friends by their names. Similarly we name memory locations to refer to them easily. These named memory locations are called variables.We store our values and results in these memory locations. A variable name consists of alphabets(Upper case and lower case), numbers and underscore sign(_) only. **Note that a variable name must not start with a number.**

Some examples of valid variable names are: age, book1, student_1 etc.

1_student, book-3, int are some examples of invalid variable names.

- 1_student : A variable name cannot start with a number.
- book-3: - is an invalid character.
- student ID: spaces are not allowed in variable names.
- int : int is a keyword. Keywords convey a special meaning to the C++ compiler.

Note: **C++ is case sensitive. It means that vnum and vNUM are considered as two different variable names.**

What values can a variable store? Let's see this in the next section.

Data Types:

Data types define the ***type of data a variable* can hold.** For example, an ***integer*** variable can hold integer data, a ***character*** type variable can hold character data, etc. Let's have a look at data types in C++.

1. int: Integer variables can only take integer values.

E.g You want to store the number of apples in a variable named num. You will define the variable as : **int num;**

Note: Here int is the datatype and num is the variable name.

Then suppose you want to state that the number of apples is 4. You will will write :
num=4;

2. <u>character:</u> Character variables store a single character like ‘a’ , ‘z’, ‘D’. Characters are always enclosed inside single quotes ‘ ‘.

Eg.: **char var = ‘a’;**

Note: Here *char* is the data type and *var* is the variable name. We have assigned character ‘a’ to the variable *var*.

3. <u>float and double:</u> Float and double variables store decimal and exponential values.

Eg.: **float salary = 9999.89;**

Eg.: **double volume = 314.86782;**

Note: Here *salary* and *volume* are the variable names and their corresponding data types are *float* and *double,* respectively.

If both float and double can store decimal values then what is the need for two separate data types?

The answer is that the size of *float* is 4 bytes and size of *double* is 8 bytes. *float* has 7 decimal digits of precision whereas *double* has 15 decimal digits of precision. Thus, *double* has two times the precision of *float*.

4. <u>bool:</u> Bool data type can contain only 2 values - true or false.

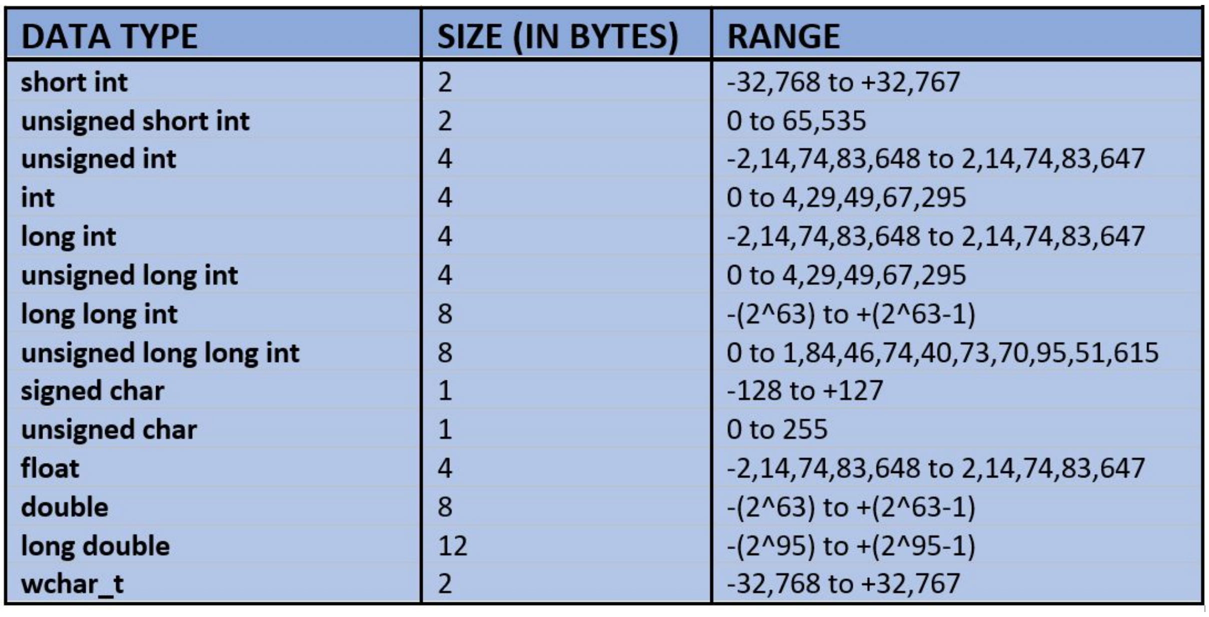
Eg.: bool cond = false;

Note: Here *cond* is a variable of *bool* type.

5. <u>void:</u> void is a keyword which means “no data”.

Note: We don’t declare variables of void type.

Data types in C++ are categorised into three groups: built-in, user-defined and derived types. Integer, character, boolean, floating-point, double floating-point are examples of built in data-types.



DATA TYPE	SIZE (IN BYTES)	RANGE
short int	2	-32,768 to +32,767
unsigned short int	2	0 to 65,535
unsigned int	4	-2,14,74,83,648 to 2,14,74,83,647
int	4	0 to 4,29,49,67,295
long int	4	-2,14,74,83,648 to 2,14,74,83,647
unsigned long int	4	0 to 4,29,49,67,295
long long int	8	-(2^63) to +(2^63-1)
unsigned long long int	8	0 to 1,84,46,74,40,73,70,95,51,615
signed char	1	-128 to +127
unsigned char	1	0 to 255
float	4	-2,14,74,83,648 to 2,14,74,83,647
double	8	-(2^63) to +(2^63-1)
long double	12	-(2^95) to +(2^95-1)
wchar_t	2	-32,768 to +32,767

NOTE: Above values may vary from compiler to compiler. Here, we have considered GCC 64-bit compiler.

Declaring More than one Variable:

If declaring more than one variable of the same type, they can all be declared in a single statement by separating their identifiers with commas.

```
int a, b, c;        //more than one variable declaration.
```

This declares three variables (a, b and c), all of them of type int, and has exactly the same meaning as:

```
int a;  //integer variable declaration.
int b;  //integer variable declaration.
int c;  //integer variable declaration.
```

Initialization of variables:

When a **variable** is defined, you can also provide an **initial value** for the **variable** at the same time. This is called **initialization**. Much like copying assignment, this copies the **value** on the right-hand side of the equals to the **variable** being created on the left-hand side.

Method 1 (Declaring the variable and then initializing it):

```
int a;
a = 5;
```

Method 2 (Declaring and Initializing the variable together):

```
int a = 5;
```

Method 3 (Declaring multiple variables simultaneously and then initializing them separately):

```
int a, b;
a = 5;
b = 10;
```

Method 4 (Declaring multiple variables simultaneously and then initializing them simultaneously):

```
int a, b;
a = b = 10;

int a, b = 10, c = 20;
```

Constant Keyword :

const' keyword stands for **constant**. In **C++** it is used to make some values **constant** throughout the program. If we make an artifact of a **C++** program as **constant** then its value cannot be changed during the program execution.

Syntax :-

```
                const    datatype    variableName = value;
```

const : Fixed Keyword.
datatype : int , float , char double etc.
variableName : user defined name of variable.

Example :-

```
        int main()
        {
                const int i = 10;
                const int j = i + 10;     // works fine
                 i++;                     // this leads to Compile time error
        }
```

Data Types Program:

In this section, we will see a basic program on data types and variables.

Program Link(Sapphire Engine): **https://sapphireengine.com/@/8zij1n**
Program Link(IDEONE): **https://ideone.com/ckaus/datatypes**

Here's our data types program:
This program takes an integer input from user and then displays it on the screen.

```
#include <iostream>
using namespace std;

int main() {

   // declaring a variable of type "int"
   int value;
   cout << "Enter an integer: " << endl;

     // taking user-input
     cin >> value;

     /* displaying the integer
         value entered by user */
     cout << "You entered: " << value << endl;

     // displaying the size of data type "int" in bytes
     cout << "Size of int(in bytes): " << sizeof(int) << endl;

     return 0;
 }
```

Output:

```
Enter an integer:
```

```
10

You entered: 10

Size of int (in bytes): 4
```

Understanding the program:

We have previously studied the use of #include<iostream>, using namespace std etc. Here we will study about how to take input from a user and then display it on the screen.

1. int value;

   In this statement we are declaring an integer type variable named value. We want to take an integer variable from the user, so we will declare an integer variable. Always declare a variable before using it to store a value.

2. cout<< “Enter an integer: “<<endl;

   You must be familiar with the cout statement. It helps us display output on the screen. Here we are displaying “Enter an integer” on the screen. Notice we have ended this statement using “endl”. When we need to insert a new line character or we want the cursor on the screen to be on the next line, we use endl. We can also use ‘\n’ instead of endl.
   E.g cout<<”Enter an integer:”<<”\n”;

3. cin>>value;

   cin is used to accept input from standard input devices like a keyboard. The above statement takes input from the user and stores that input into the variable named value.
   If we want to take multiple inputs from the user we can write as:
   Suppose I want to store the multiple inputs into 2 variables a and b.
   cin>>a>>b;

4. cout<<”You entered: ”<<value<<endl;

   We know that cout is used to display output on screen. We can also use it to display the contents of a variable. In this statement it will first

display “You entered: “ and then it will display the contents of variable value.
Eg Suppose the user input is 5. So 5 is stored in the value variable.
Now the above statement will give output as:
You entered: 5

5. cout<<”Size of int(in bytes) : “<<sizeof(int)<<endl;
The above statement displays the size of an integer. The sizeof is a keyword used to get the size of any data-type.

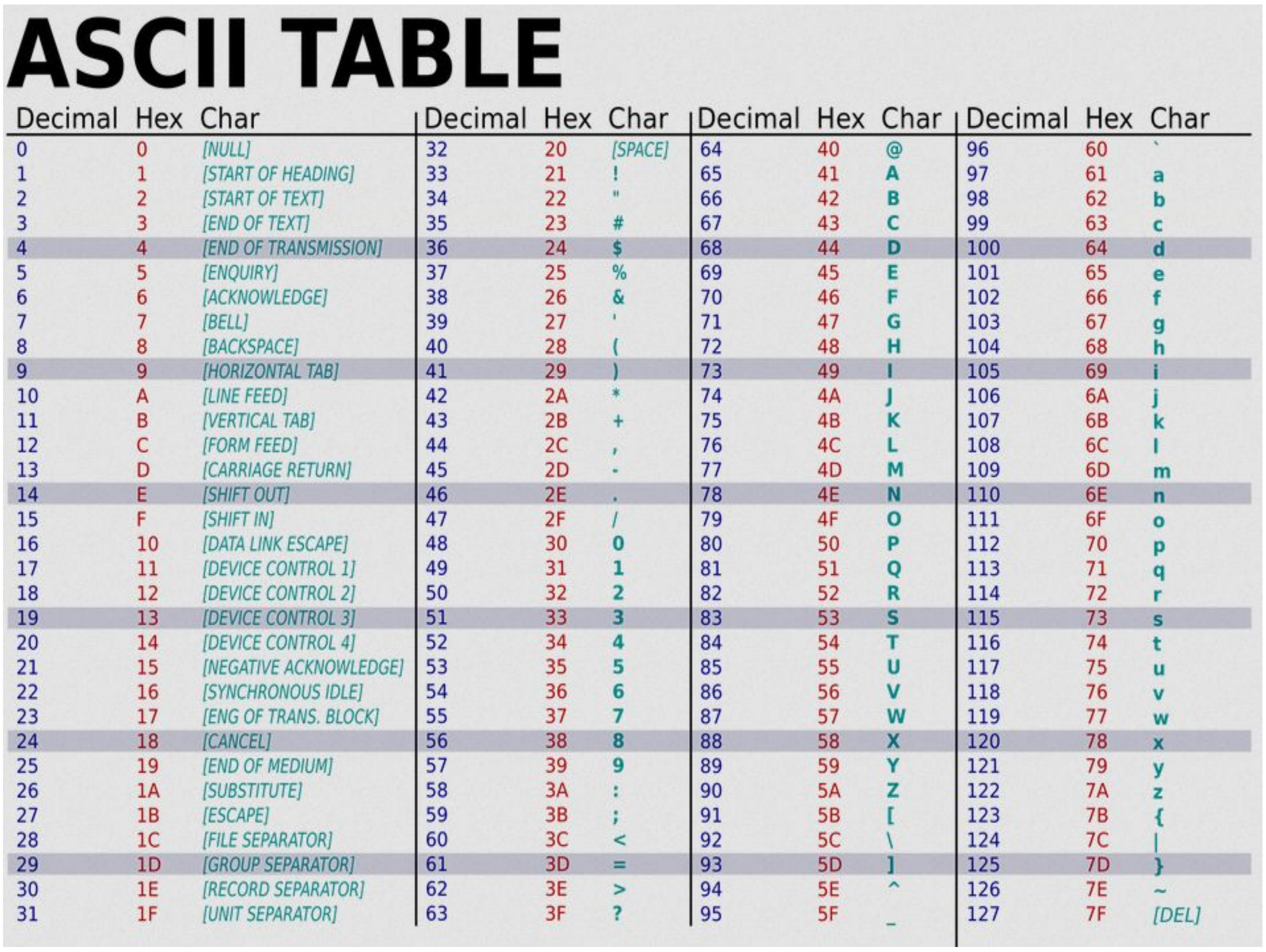
Brain-Teaser

We all know that computers understand only 0’s and 1’s, then how do they understand characters and decimal numbers?
Since computers only understand 0’s and 1’s everything we give as input is to be converted into a form that is easily understood by a computer, i.e, into a binary system.
Decimal numbers can be converted into binary numbers, but how to convert characters?
For characters we have ASCII codes. ASCII stands for American Standard Code for Information Interchange. ASCII is standard used to represent characters in computers or any other electronic devices. ASCII codes consist of 128 characters and their encodings. Each character corresponds to a 7 digit sequence of 0’s and 1’s which can be represented easily by a decimal number.
ASCII characters are divided into 4 groups:

1. (0-31): Non printable characters
2. (32-64): Numeric digits and punctuations
3. (65-95): Capital Letters
4. (96-127): Lower case letters

E.g ‘A’ corresponds to `1000001` i.e 65 in the decimal number system.

ASCII table is given below for your reference:



ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Question 1

What will be the output?

char c=74;

cout<<c<<endl;

The output will be ‘J’. In this code snippet we are assigning integer value to character data type so this integer is treated as ASCII value of corresponding character. ‘J’ is printed because 74 is the ASCII value of ‘J’.

Question 2

What will be the output?

cout<<’b’+1;

The output will be 99 because when we add a character and an integer, it adds the ASCII value of the character and the integer. The ASCII value of ‘b’ is 98. We add 1 to it, therefore the output is 99.

Statements and Expressions

Statements:

Fragments of a C++ program which are executed in a sequence are called **statements**. A statement is a complete instruction used to indicate the computer to carry out a task.

E.g

1) int n=2;   is a statement
   Here we are indicating the computer to declare an integer variable n and initialise it with 2.
2) cout<<”n= “<<n<<endl;   is another statement.
   Here we are indicating the computer to display the value of variable n

Note: Statements always end with a semicolon ‘;’

Compound statements: When we want to execute multiple statements in sequence we enclose them in brackets { }

E.g

```
{
int n=3;
cout<<” value of n: “<<n<<endl;
}
```

Note : The above 2 statements will be executed in a sequence one after the other.

Expressions:

C++ expression consists of variables, constants and operators. There can be 0, 1 or more than one operator in an **expression**. We can also say that an expression is a sequence of operators and operands which evaluates to a value.

E.g 1) i+1 is an expression

2) 3<2 is an expression

3) (a<b || a<c) is also an expression

4) c   This is one of the simplest expressions.

Note: A single variable or a constant is also an expression.

5) i=a+1; is an expression and a statement..

In this example we are indicating the computer to store or assign the value of a+1 into variable i. Hence it is a statement.

This is also a valid expression. Note that this expression evaluates to value a+1.