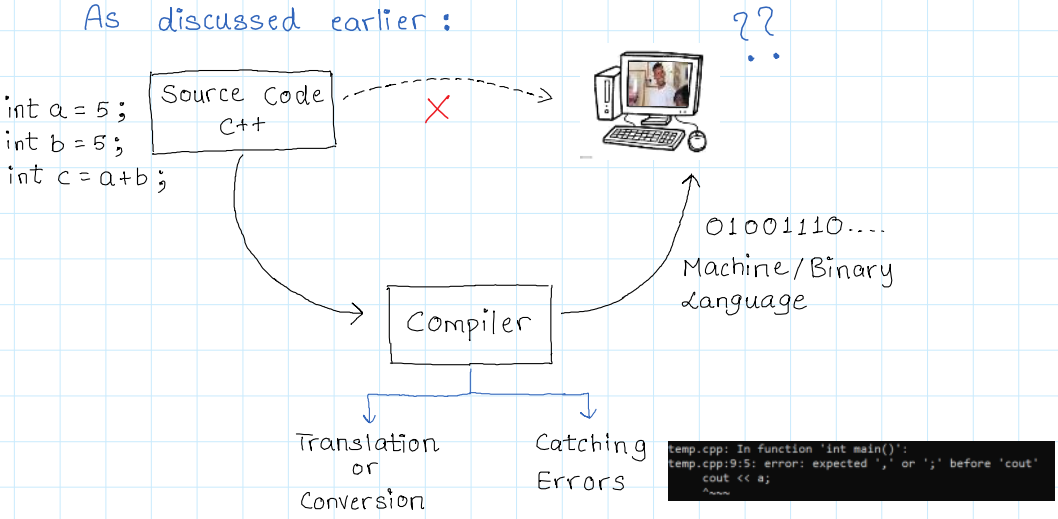
First Program in C++

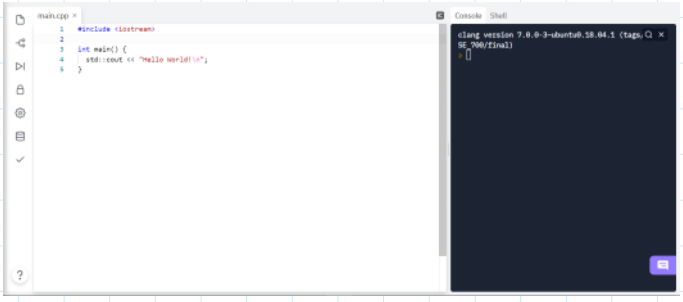
04 January 2022 20:53

As discussed earlier :



IDEs (Integrated Development Environment) :

An environment that helps you write, run and even debug (in some cases) code in a programming language.
Eg: VS CODE, Code Studio, Eclipse, NetBeans, etc.



First Program in C++ : (Namaste Duniya)

① Our program will find `int main(){ }` and start executing from there.

② `int main(){ }` These brackets show the 'scope' of the main function i.e. the code which belongs to / is defined within int main() function.

More about scope of a function

③ In C++, we use 'cout' to print something.
Eg: `cout << "Namaste Duniya";`

④ This cout is already defined in a file (header file)

which must be included before using cout.

⇒ # include < file_name > is a preprocessor directive which runs before the program is compiled and includes the file to be used later in the source code.

A file called iostream has cout defined in it so:

include <iostream>

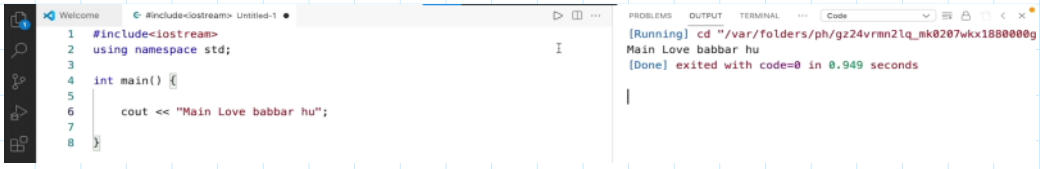
Hint: i/o means input/output.

⑤ Namespaces : Stack Overflow Question

using namespace std ;

⑥ Using cout :

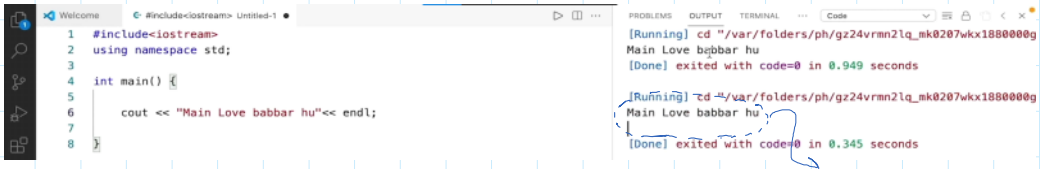
We use '<<' after 'cout' to display something to Standard Output (your screen) within std namespace.



⑦ endl : Used to enter new line. Just like ENTER.

endl is like "\n" which is a new line escape sequence character used in various languages including C++.

cout << " Namaste Duniya " << endl ;



New line

⑧ endl; ; is used to terminate statements.

DATA TYPES :

Different types of data to be stored in memory. Eg- integer, float, character, double, etc.

Eg- int : Stores integers like -5, 0, 8, etc.
char : Single character like 'a', '+', '$', '7', etc.
float : Floating point values like -2.014, 1.0000, 6.7800

Different data types use different amounts of memory. Amount of memory used also depends on the architecture of your CPU.

Data Type	Meaning	Size (in Bytes)
`int`	Integer	2 or 4
`float`	Floating-point	4
`double`	Double Floating-point	8
`char`	Character	1
`wchar_t`	Wide Character	2
`bool`	Boolean	1
`void`	Empty	0

Source: Programiz

Character : A 1-byte (= 8 bits) data type that takes 1 character.

```
char ch = 'a';
```

Boolean : True / False. Take 1 bit and 1 : True
0 : False

```
bool isGood = 1;
bool isBad = false;
```

Float / Double : Float takes 4 / 8 bytes
Double takes 8 bytes.

```
float num1 = 1.2;
double num2 = 2.4;
```

Know more about C++ Data Types

Variable Naming / Nomenclature :

1. Can contain alphabets, numbers and underscores.
2. Cannot start with a number.
3. Cannot be keywords like int, cout, double, bool, etc.
4. Case sensitive
5. Cannot contain special symbols like %, $, !, #, etc.

WARNING : Don't use same variable name multiple times.

```
int a = 123;

cout<< a << endl;

char a = 'v';

cout<< a <<endl;
```

```
ning] cd "/var/folders/ph/gz24vrmn2lq_mk0207wkx1880000gn/T/"
CodeRunnerFile.cpp:10:10: error: redefinition of 'a' with a
char a = 'v';
         ^
CodeRunnerFile.cpp:6:9: note: previous definition is here
int a = 123;
    ^
```

Using different data types in code :

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

int main() {

    int a = 123;

    cout<< a << endl;

    char b = 'v';

    cout<< b <<endl;

    bool bl = true;
    cout<< bl << endl;

    float f = 1.2;
    cout<< f << endl;

    double d = 1.23;
    cout<< d << endl;

}
```

```
[Running] cd "/var/folders/ph/gz24vrmn2lq_mk0207wkx1880000g
123
v
1

[Done] exited with code=0 in 0.595 seconds

[Running] cd "/var/folders/ph/gz24vrmn2lq_mk0207wkx1880000g
123
v
1
1.2

[Done] exited with code=0 in 0.551 seconds

[Running] cd "/var/folders/ph/gz24vrmn2lq_mk0207wkx1880000g
123
v
1
1.2
1.23

[Done] exited with code=0 in 0.537 seconds
```

Check the size of different Data Types for your system using sizeof(variable_name);

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

int main() {
    int a = 4;
    double b = 1.90;
    char c = '+';
    int sizeInteger = sizeof(a), sizeDouble = sizeof(b), sizeChar = sizeof(c);
    cout << "Size of an integer is " << sizeInteger << " bytes" << endl;
    cout << "Size of a double is " << sizeDouble  << " bytes" << endl;
    cout << "Size of a char is " << sizeChar  << " bytes" << endl;
}
```

```
> clang++-7 -pthread -std=c++17 -o main mai
> ./main
Size of an integer is 4 bytes
Size of a double is 8 bytes
Size of a char is 1 bytes
>
```

HOW IS DATA STORED IN MEMORY ?

Eg: int a = 8; // int takes 4 bytes = 32 bits

In binary, 8 = 1000 (4 bits needed)

∴ int a → 00000000 00000000 00000000 00001000 } 32 bits

Eg: int b = 5;

b

| 5 |

4 bytes.

address = 100 (assume)

100, 101, 102, 103

4 bytes are consumed

Eg: char c = 'a';

Characters are mapped to the standard ASCII values

'a' → 97		'A' → 65	
'b' → 98		'B' → 66	
'c' → 99		'C' → 67	
⋮	⋮	⋮	⋮
'z' → 122		'Z' → 90	

ASCII Table

(Homework)

char c = 'a';

$$\xrightarrow{\text{ASCII}} 97 \xrightarrow{\text{Binary}} 1100001 \xrightarrow{\text{Store}} \underbrace{0\ 1\ 1\ 0\ 0\ 0\ 1}_{1\ \text{byte}}$$

Type Casting and Storing Variables

05 January 2022 17:58

Conversion of one data type to another (if it is valid) is called Type Casting.

Example :

int a = 'a';

variable a will store the ASCII value of 'a' = 97.

Example :

char ch = 98;

98 will automatically get typecasted to its corresponding character.

This automatic typecasting is called implicit typecasting.

```
int a = 'a';
cout<< a << endl;

char ch = 98;
cout << ch <<endl;
```

→

```
[Running] cd "/var/folders/ph/gz24vrmn2lq_mk0207wkx1880000g
97
b
```

What if we type cast from int to char but the value is too large to be stored in char ?

Soln : A warning is thrown and the last byte from the original value is given to the character.

```
char ch1 = 123456;
cout<< ch1 <<endl;
```

→

```
[Running] cd "/var/folders/ph/gz24vrmn2lq_mk0207wkx1880000g
tempCodeRunnerFile.cpp:33:16: warning: implicit conversion
    char ch1 = 123456;
        ~~~   ^~~~~~
1 warning generated.
```

How are negative numbers stored ?

Soln : The first bit tells us whether the number is positive or negative.

First Bit → 0 means Positive
→ 1 means Negative

Steps to store -5 in binary format :

1. Ignore the -ve sign. (5)
2. Write the binary representation of 5.
3. Take its 2's complement and store it.

Example : a = -5.

1. -5 → 5 (Ignore the -ve sign)
2. 5 → 0101 → $\underbrace{00\ldots\ldots0}_{29\text{ Zeros}}101$ (Binary)
3. 2's complement = 1's Complement + 1.

5 → 00.....0101

1's compl. → 11.....1010 (Flip the bits)

\+ 1

2's Compl. → 11.....1011

$$+ \quad 1$$

2's Compl. $\longrightarrow$ $\underbrace{11\ldots\ldots}_{29\text{ Ones}}1011$

Displaying Negative Number :

① Take 2's complement of the stored number

Stored : 111 1011

This shows -ve

	11.... 1011
1's comp.	00....0100
2's comp.	00...0101 = 5

-5 print ho gaya !

Why 2's Complement ?

If we stored numbers as it is without using 2's complement, then

1	0	0	-----	0	0

and

0	0	0	-----	0	0

will be equal & thus waste space.

Store only Positive Integers

The default signed representation allows us to store both positive & negative values.
To store only positive integers, we use unsigned..

Eg: unsigned int a = 10 ;

What if we store a negative value in an unsigned number ?

Example : unsigned int a = -112 ;
cout << a << endl ;

Output :
4294967184 ??

Explaination :
We tried to store -112.

-112 = 2's Complement of 112.

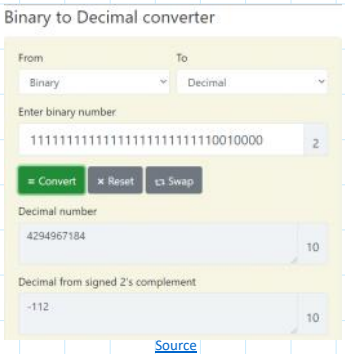
112 = $\underbrace{00\ldots\ldots0}_{25\text{ Zeros}}1110000$.

1's Compl. = 11 10001111

$+ \quad 1$

2's Compl. = 11.....10010000

Unsigned int uses all 32 bits to store the value and the MSB (=1) will make the value.
An unsigned int does not use the 2's complement again to display the number.
Thus, $\underbrace{11\ldots\ldots1}_{25\text{ ones}}$10010000. gets printed as it is in decimal.



Source

Therefore,

```
unsigned int a = -112;

cout<< a << endl;
```

→

```
[Running] cd "/var/folders/ph/gz24vrmn2lq_mk0207wkx1880000g
4294967184
```

Operators

05 January 2022 19:05

Basic Arithmetic Operators:

+ , - , * , / , %

Caution ⚠

① int / int = int (Floor value of answer)

Examples: 5/2 = 1.

3/5 = 0

9/2 = 4

② int / float } float
float / int

double / int } double
int / double

cout << 5.0/2 << endl;

Output:

2.5

```
float a = 2.0/5;

cout << a <<endl;

cout<< 2.0/5 <<endl;
```

```
[Running] cd "/var/folders/ph/gz24vrmn2lq_mk0207wkx1880000g
0.4
0.4

[Done] exited with code=0 in 0.591 seconds
```

Relational Operators:

== , >= , <= , > , < , !=

① Is a = b ?

a==b —Yes→ 1
a==b —No→ 0

② Is a greater than or equal to b ?

a >= b —Yes→ 1
a >= b —No→ 0

```
int a = 2;
int b = 3;

bool first = (a==b);
cout<< first <<endl;

bool second = (a>b);
cout<< second <<endl;

bool third = (a<b);
cout<< third <<endl;

bool fourth = (a>=b);
cout<< fourth <<endl;

bool fifth = (a<=b);
cout<< fifth <<endl;

bool sixth = (a!=b);
cout<< sixth <<endl;
```

```
[Running] cd "/var/folders/ph/gz24vrmn2lq_mk0207wkx1880000g
0

[Done] exited with code=0 in 0.851 seconds

[Running] cd "/var/folders/ph/gz24vrmn2lq_mk0207wkx1880000g
0
0
1
0
1
1

[Done] exited with code=0 in 1.124 seconds
```

Logical Operators:

&& , || , !

(AND) (OR) (NOT)

① Logical AND:

All conditions must be true for the output to be true.

Example: int a = 5, b = 10, c = 15;

```
cout <<((a>0) && (b!=0) && (c<=15));
```

Output :

1

② Logical OR :

At least 1 condition must be true for the output to be true.

Example :

```
int a=5, b=10, c=15;
cout <<((a>5) || (b<10) || (c>=15));
```

Output :

1

③ Logical NOT :

Inverts the logic. True ⇌ False

Non-zero ⇌ Zero

Example :

```
int a=10, b=0;
cout << (!a) << endl;
cout << (!b) << endl;
```

Output :

0

1

Point out what was not covered in Typecasting and try to code yourself.