Write your first c++ Program.

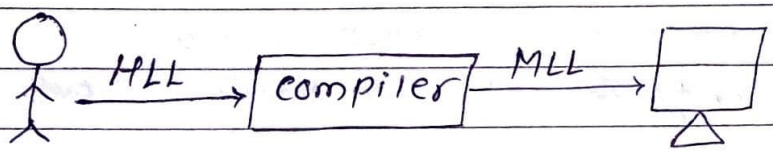
Date: ...
Page: ...

Programming Languages

A language using which we can instruct the computer to carry out real life tasks & computations. It acts as a language in which we could easily express our thoughts to the machine.

It has a fixed set of rules. These programs are then converted into a language, which machines can understand. These converting task is carried out by compiler.

Every language has its own compiler / interpreter.



Compiler act as a translator, which convert (translate) High level language to machine level language.

First c++ code

c++ code start with int main () function

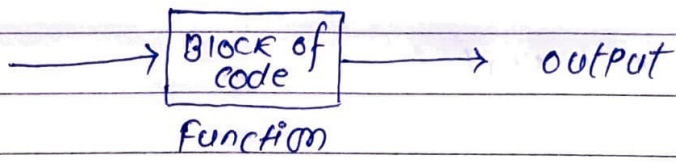
Function is a block of code, which perform a specific task.

```
int main () {
   // Body of function.
}
```

int = Return type of function, it return integer

main () = Name of function

{ } = curly braces define the scope of main function starting & end point of function.



Date: .....
Page: .....

→ How compiler work ?

Source code → compiler → Machine code → output

→ How interpreter work ?

Source code → interpreter → output

→ Let's write down the first code.

i) code execution always start from int main() function.
ii) int main () is an inbuild function / method
iii) function is an entity in which we provides inputs & we may get output.

```
#include <iostream>
using namespace std;
int main () {
    cout << " Hello Duniya " << endl;
    return 0;
}
```

i) # include - preprocessor directive used to include files.
ii) <Postream> - Header file for input & output operations.
iii) int main () - Entry point of code
iv) cout - Identifier to print something on display.
v) << - Insertion operator
vi) ; - Terminator (semi-colon)

Endl - End of Line / next line
'\n' - New line character

Date: .....
Page: .....

→ #include<iostream>

#include - It is a preprocessor directive that tells the C++ compiler to include the content of a specific file in your program before it's compiled. This is commonly used for include standard library.

<iostream> - It's a standard C++ header file that provides the functionality of input & output operations.

→ using namespace std;

namespace - It's a mechanism for organizing code to prevent naming conflicts. It allows you to group related classes, functions & variables under a common name.

std - It is the short form for the "standard" namespace in C++. It contains various components of the C++ standard library, including i/p & o/p streams containers, algorithms & more.

By using the "std" namespace, we can avoid naming conflicts with code or other libraries that might define similar names.

→ cout - It is stands for character output. It's an output stream object provided by the C++ standard library. It is used for printing or displaying text / data to the console or standard output.

→ return 0; - It is used to signify that the program executed successfully without any errors. This return value is typically inspected by the operating system or calling process to determine if the program ran without issues.

Date: .....
Page: .....

→ Variables & Data Types.

Variables - It is a container that is used to store the data values. Variable is a name given to a memory location. The value stored in a variable can be changed during program execution. All the operation done on the variables effects that memory location.

Data Types - The type of value whether int, char, float, double etc. stored in the variable is defined by the Data-types.

Data Types in c++

i) Primitive	ii) Derived	iii) user-defined
↳ integer	↳ function	↳ class
↳ character	↳ Array	↳ structure
↳ Boolean	↳ pointer	↳ union
↳ Floating point	↳ Reference	↳ Enum
↳ Double		↳ Typedef
↳ void		
↳ wide character		

```
int marks = 90 ;
Data Type   variable name   Assignment operator   value   Terminator
```

{ 1 byte = 8 bits
1 bit = 0 or 1 }

```
int num = 5;        char ch = 'S';        float f = 10.2;
```

5	S	10.2
num	ch	f

Date: .....
Page: .....

→ Data Type Tell us two things.
i) The type of data stored in variable.
ii) size of data (size of memory it will take)

NOTE : To find the size of a specific data type, we use the sizeof() operator in c++.

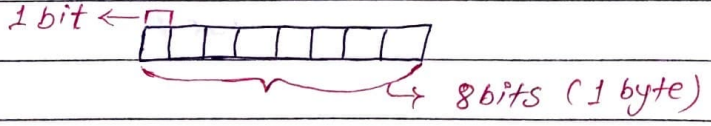
```
int a = 5;
cout << sizeof(a) << endl;  // output = 4 byte
```

→ The actual size of data types may vary on different systems & compiler.

→ Data Types & size

Data Type	size	Description
int	2 / 4 bytes	Store whole number, without decimals
char	1 byte	store a single character / letter / number or ASCII values.
float	4 bytes	Stores fractional no. containing one or more decimal. sufficient for 6-7 decimal
double	8 bytes	Some as float. sufficient for 15 " digits.
boolean	1 byte	stores true or false
short	2 bytes	
long	4 "	
byte	1 byte	stores from -128 to 127

→ size of memory measures in terms of 'bytes'.



Date: .....
Page: .....

Smallest addressable memory size must be atleast 1 byte.

→ Declaration of variable

Syntax : Data-type variable_name ;

int num ; char ch ;

→ Initialization of variable

Syntax : Data-type variable_name = value ;

int num = 50 ; char ch = 'S' ;

→ In C++, if we declare an integer variable without initialize it with a value, the value will contain that is known as an "indetermine" or "Garbage" value. The actual value of the uninitialized variable will be unpredictable & depends on the current state of the memory at the time the program runs.

Eg :-
```
# include <iostream>
using name-space std;
int main () {
    int num ;
    cout << num << endl;
    return 0;
}
```

Output is unpredictable.

Decimal	Binary	Decimal	Binary
1	1	6	110
2	10	7	111
3	11	8	1000
4	100	9	1001
5	101	10	1010

Date: .....
Page: .....

→ How Positive & Negative data stored?

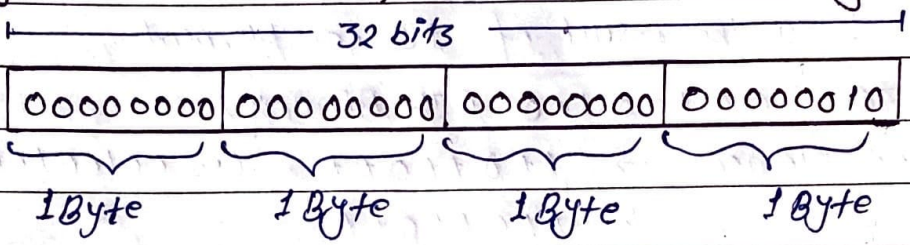
Positive data (integer) typically stored in memory using a binary representation.

First data convert into binary representation & then stored in memory according to data type size.

Eg:- int num = 2;

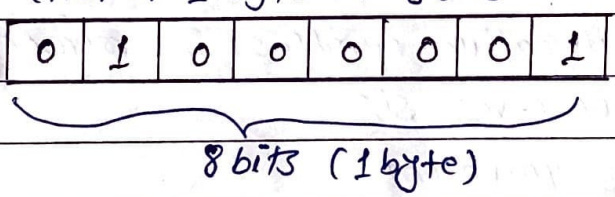
Binary representation = 10

Here, data type is 'int', which size is 4 bytes (32 bits). So the binary representation of '2' stored in memory using 32 bits



char ch = 'A'; (ASCII value of A = 65)

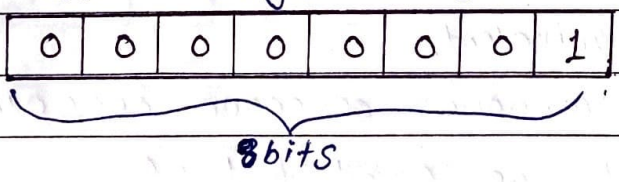
char → 1 Byte → 8 bits



Binary equivalent of ASCII value of 'A' is stored in the memory

bool flag = True; (True = 1 & False = 0)

bool → 1 Byte → 8Bits



Binary equivalent of true (i.e 1) is stored in the memory.

Date: .....
Page: .....

NOTE: Bool data type store 1 (true) or 0 (false). It can be done or stored using only 1 bit. But it is not possible because smallest addressable space in the memory is 1 Byte.

→ 1's complement

change binary digits 1 to 0 & 0 to 1

Eg:- 100110, 1'S complement = 011001

→ 2's complement

1) Find 1's complement (flip 1 & 0)

2) Add 1 to the result of 1's complement.

Eg:- 36 = 00100100 (In Binary)

1'S complement → 11011011 (∵ 00100100)

2'S " → 11011011

+1

11011100 (-36)

→ Steps to store Negative numbers in memory.

i) Ignore negative (-ve) sign.

ii) Find Binary Equivalent

iii) Take 2'S complement

Eg:- int a = -5;

1st, ignore -ve sign. i.e. int a = 5;

2nd, find binary equivalent,

00000000 00000000 00000000 00000101

3rd, 1'S complement, i.e. 0→1 & 1→0

11111111 11111111 11111111 11111010

4th. 2'S complement,

i.e. Add 1 to 1'S complement.

Date: .....
Page: .....

```
11111111 11111111 11111111 11111010
                                +1
-------------------------------------
11111111 11111111 11111111 11111011  → -5 stored in memory
```

+ How To calculate the Range of Data Types.

1) calculate data type size in bits.
   Eg:- int = 4 bytes = 32 bits.

2) Find the total combinations.
   Total combination = $2^{\text{bits size}}$ = 2^{32}

3) Find the range of data type.
   $2^{\text{bits size}} - 1$ = $2^{32} - 1$
   Range = 0 → $2^{32} - 1$ (Range of Integer)

Formula to calculate the range & total combination of any types of data (for unsigned data type)

Data type size = n bits
combination = 2^n
Range : 0 → $2^n - 1$

Eg:- char = 1 byte → 8 bits
Total combination = 2^8
Range : 0 → 255
or, 0 → $2^8 - 1$

{ As there're 8 blocks & each block can hold two values, i.e. 0 & 1

long → 8 bytes → 64 bits
Total combination = 2^{64}
Range : 0 → $2^{64} - 1$

Date: .....
Page: .....

→ Signed & unsigned Data

Signed Data - It can hold positive, negative numbers & zero. i.e. all integer values.

unsigned Data - It can hold only positive numbers & zero

→ Ranges of Data Types

→ To find whether the stored number in memory is positive or negative.

Most significant Bit (Left most bit)

The left most bit or most significant bit tells whether the stored number is positive or negative.

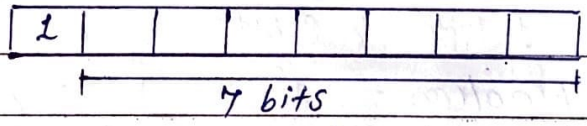
if (sign bit = 0), Then positive number

if (sign bit = 1), Then negative "

1Y char → 1 byte → 8 bits

Positive = 

Total combination = 2^7 (unsigned)

Negative = 

Total combination = 2^7 (signed)

∴ Positive Range = $0 \rightarrow 2^7 - 1$

= $0 \rightarrow 127$

Negative Range = $-1 \rightarrow -2^7$

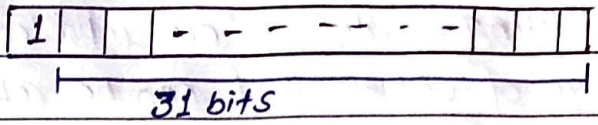
= $-1 \rightarrow -128$

} 0 is not considered as already considered in Positive Range

Combined Range = $-128 \rightarrow 127$

Date: .....
Page: .....

2) int → 4 byte → 32 bits

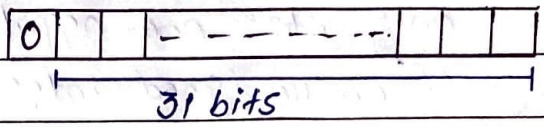
signed Integer → 

combination = 2^{31}

minimum value
$= -2^{(\text{bits size} - 1)}$
$= -2^{32-1} = -2^{31}$

maximum value $= -2^{(\text{bits size}-1)} - 1$
$= -2^{32-1} - 1 = -2^{31} - 1$

Ranged of signed integer : $-2^{31} \backsim -2^{31}-1$

unsigned Integer → 

combination = 2^{31}

minimum value = 0
Maximum " $= 2^{\text{bitsize}-1} - 1$
$= 2^{32-1} - 1 = 2^{31} - 1$

Range of unsigned integer : $0 \backsim 2^{31} - 1$.

∴ combined Range : $-2^{31} \backsim 2^{31} - 1$.

→ If we total 'n' bits of data size, Then combined range of that type is : $-2^{n-1} \backsim 2^{n-1} - 1$.
→ signed & unsigned combinations : 2^{n-1}
→ signed Range : $-1 \backsim -2^{n-1}$
→ unsigned " : $0 \backsim 2^{n-1} - 1$

Imp

NOTE: It's important to note that, when working with signed integers, one bits is used for the sign (+ve, -ve), so the range of positive value is one less than the range of unsigned integers with same no. of bits.

Date: .....
Page: .....

+ More Detail about Range of signed & unsigned Integer

In computing, integers are represented using fixed no. of bits. The range of values that can be represented depends on whether the integer is signed or unsigned as well as the no. of bits used.

The range of values for signed & unsigned integer can calculated using the formula 2^n. (n = no. of bits)

1) Unsigned Integers.

- An unsigned integer uses all its bits to represent non-negative values. It only store non-negative values.
- The range of an unsigned integer with 'n' bits is:
  0 to $2^n - 1$.

Eg:- 32 bits unsigned integer = 0 to $2^{31} - 1$
= 0 to 4294967295

16 bits unsigned integer = 0 to $2^{16} - 1$
= 0 to 65535

2) Signed Integers.

- A signed integer allocates one bit for the sign (+ve, -ve) & uses the remaining bits for the magnitude. It store both +ve & -ve numbers.
- The range of a signed integer with 'n' bits is:
  -2^{n-1} to $2^{n-1} - 1$.

Eg:- 32-bits signed integer = -2^{n-1} to $2^{n-1} - 1$
= -2^{31} to $2^{31} - 1$

16-bits signed integer = -2^{n-1} to $2^{n-1} - 1$
= -2^{15} to $2^{15} - 1$.

Date: .....
Page: .....

Operators

```
int sum = a + b;
```

`+` → operator
`a`, `b` → operands

1) Arithmetic operator.

i) + (addition : x+y)
ii) - (subtract : x-y)
iv) / (division : x/y → division returns quotient)
v) % (modulus : x%y → Modulus " remainder)
iii) * (multiplication = x*y)

Eg:- 10/5 return 2. & 10 % 5 return 0.

2) Assignment operator

i) = (a=b)
ii) += (a+=5 → a = a+5)
iii) -= (a-=5 → a = a-5)
iv) *= (a*=5) → a = a*5)
v) /= (a/=5 → a = a/5)
vi) %= (a %=5 → a = a%5)

3) Relational (comparison) operator.

i) == (is equal to → 3==5 return false)
ii) != (not " " → 3!=5 " true)
iii) > (Greater than → 3>5 return false)
iv) < (Less than → 3<5 " true)
v) >= (Greater than or equal to → 3>=5 return false)
vi) <= (Less " " " → 3<=5 " true)

4) Logical operator.

i) && (Logical AND → Return true if both statements are true)
ii) || (" OR → " " " one of the statement is true)
iii) ! (" NOT → " the result, returns false if the result is true, return true is the result is false)