CPP Summary Session 3 (07-10-22)

- Keyboard is a Standard Input Device. And Console is Standard output device - that's why while printing a value in console we type - **std::cout<<"hii"; (std - means standard)**
- **ASCII Table** - ASCII stands for American Standard Code for Information Interchange. In ASCII Table binary value of almost all the available characters, symbols and numbers is mentioned.
- **8 bits** =1 bytes
- 8 bits are like 8 transistors(torches), every transistor can store on bit information, which means it can be either 0 or 1 (**on or of**)
- The language of the Machine is "**binary**" . It understands only 0 and 1, So it means we will have to convert our high level code into machine language.
- RAM is volatile in nature, because if we cut the electricity supply, Then all the transistors will be set to off mode(0) hence we will lose our data.
- **Compiler** is the program which converts our cpp code into machine language i.e binary language
- Before running a CPP Code, everytime we need to compile it first.
- The binary notation of number 65 and letter "A" is **01000001 ,** Now as both of them have the same binary notation, So at the time of storing this "A" or 65 in RAM we can store it, But while reading this binary data we should know in which form we need to read it (in character or in integer form) and that's why **data type** plays a vital role, data type tell our program to read this binary notation in char or in integer form.
- Compilers are the ones which convert code into machine language, so different compilers reserve different bytes of space for the same data types.
- The compiler we are using here is **mingw**, and this compiler reserves 4 byte of space for int, and 2 bytes of space for char.
- Get the size of any datatype with the help of **sizeof** method
- By default for integer data type "**mingw**" compiler reserves 4 byte of space which is equal to **4*8=32 bits**, And this means a total 2^32 different number combinations can be stored by this data type, if you need more space then use some modifiers like **long** and **double, which will provide 8 bytes of space.**

helloworld.cpp

```cpp
#include "iostream"

main(){

    int x=7;
    float y=7.5;
    char z = 'a' ;

    std::cout<<"int:";
    std::cout<<sizeof(x);
    std::cout<<"\n";
    std::cout<<"float:";
    std::cout<<sizeof(y);
    std::cout<<"\n";
    std::cout<<"char:";
    std::cout<<sizeof(z);

```

```
C:\my_projects\CPP\helloworld.exe
int:4
float:4
char:1
--------------------------------
Process exited after 0.1248 seconds with return value 0
Press any key to continue . . .
```

- Now, as a good developer, we will need to manage out memory in the right way, So that we can save some memory while running our code, For example if we thing that we are going to create a integer variable whose value will be in range of few lakhs, So in this case by using some **modifiers** we can save some byte of extra space in RAM, check the example below, Here i have saved 2 bytes of space by using modifiers(*by default int datatype reserve 4 byte of space, and now with the help of modifier it is just reserving 2 byte of space so 4-2 =2 byte of space is been saved*)

helloworld.cpp

```cpp
#include "iostream"

main(){

    short int x=54545454;
    std::cout<<sizeof(x);


}
```

```
C:\my_projects\CPP\helloworld.exe
2
--------------------------------
Process exited after 0.122 seconds with return value 0
Press any key to continue . . .
```

- Now if we think we are going to assign a big value to our integer variable then we can use modifiers like **double**, which will reserved 8 bytes of space.

helloworld.cpp

```cpp
#include "iostream"

main(){

    double x=7009088978786;
    std::cout<<sizeof(x);


}
```

```
C:\my_projects\CPP\helloworld.exe
8
--------------------------------
Process exited after 0.1287 seconds with return value 0
Press any key to continue . . .
```

Now as shown above we can optimise our RAM space by selecting the right data type.

- In cpp we have some interesting functions which can tell us what maximum and minimum range can be stored in a data type. To use these functions we will have to first of all import a new library named **"limits".** Like in this case "char" data type has range from -128 to 127

helloworld.cpp

```cpp
#include "iostream"
#include "limits"

main(){
    std::cout<<CHAR_MAX;
    std::cout<<"\n";
    std::cout<<CHAR_MIN;


}
```

```
C:\my_projects\CPP\helloworld.exe
127
-128
--------------------------------
Process exited after 0.0889 seconds with return value 0
Press any key to continue . . .
```

helloworld.cpp

```cpp
#include "iostream"
#include "limits"

main(){
    std::cout<<INT_MAX;
    std::cout<<"\n";
    std::cout<<INT_MIN;


}
```

```
C:\my_projects\CPP\helloworld.exe
2147483647
-2147483648
--------------------------------
Process exited after 0.06486 seconds with return value 0
Press any key to continue . . .
```

If you will notice by default for integer data type we have 4bytes of space reserved. Now 4 bytes means - ***4*8=32*** *bits and this means* ***2^32=4294967296*** *combinations of numbers*

Now in integer we can store both negative and positive numbers, so if i divide this combinations with 2(because negative and positive two type of combinations we will have) we will get - ***4294967296/2 =2147483648***

*Now if we match this(****2147483648)*** *with the range of Integer data type(shown in above screenshot) which is (****-2147483648- 2147483647****) matches perfectly fine.*