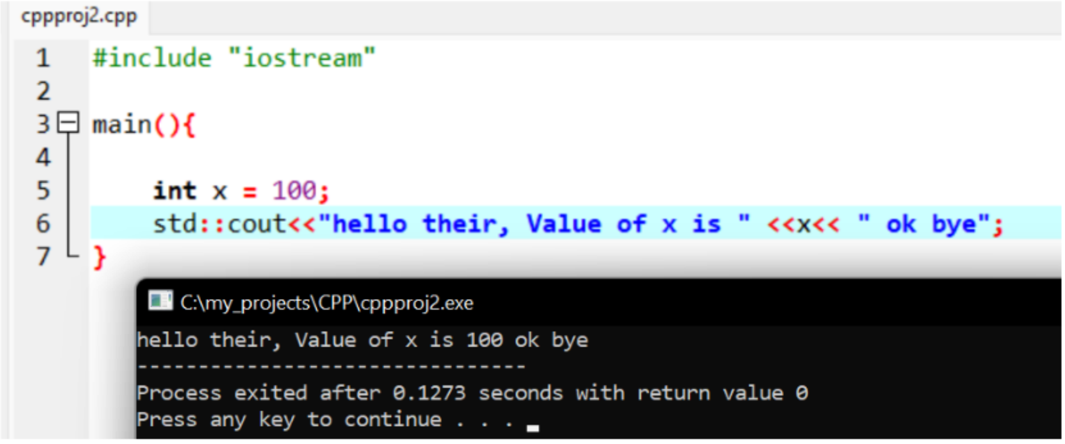
Summary CPP

Sessions No 5(14-10-2022)

- Our compiler is the one, which decides how much space to reserve for any particular datatype.
- ***std::cout<<"hello their, Value of x is " <<x<< " ok bye";***
  Output **- hello their, Value of x is 100 ok bye**



cppproj2.cpp

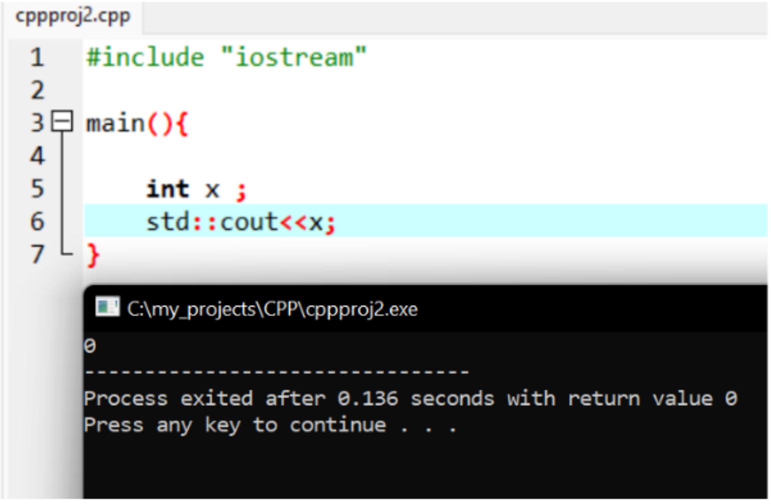
```
#include "iostream"

main(){

    int x = 100;
    std::cout<<"hello their, Value of x is " <<x<< " ok bye";
}
```

C:\my_projects\CPP\cppproj2.exe

```
hello their, Value of x is 100 ok bye
--------------------------------
Process exited after 0.1273 seconds with return value 0
Press any key to continue . . .
```

- In the above example, it looks like everything is coming from a single string, but in reality, The value of “x” comes from RAM, And other strings are hardcoded by us in the program.

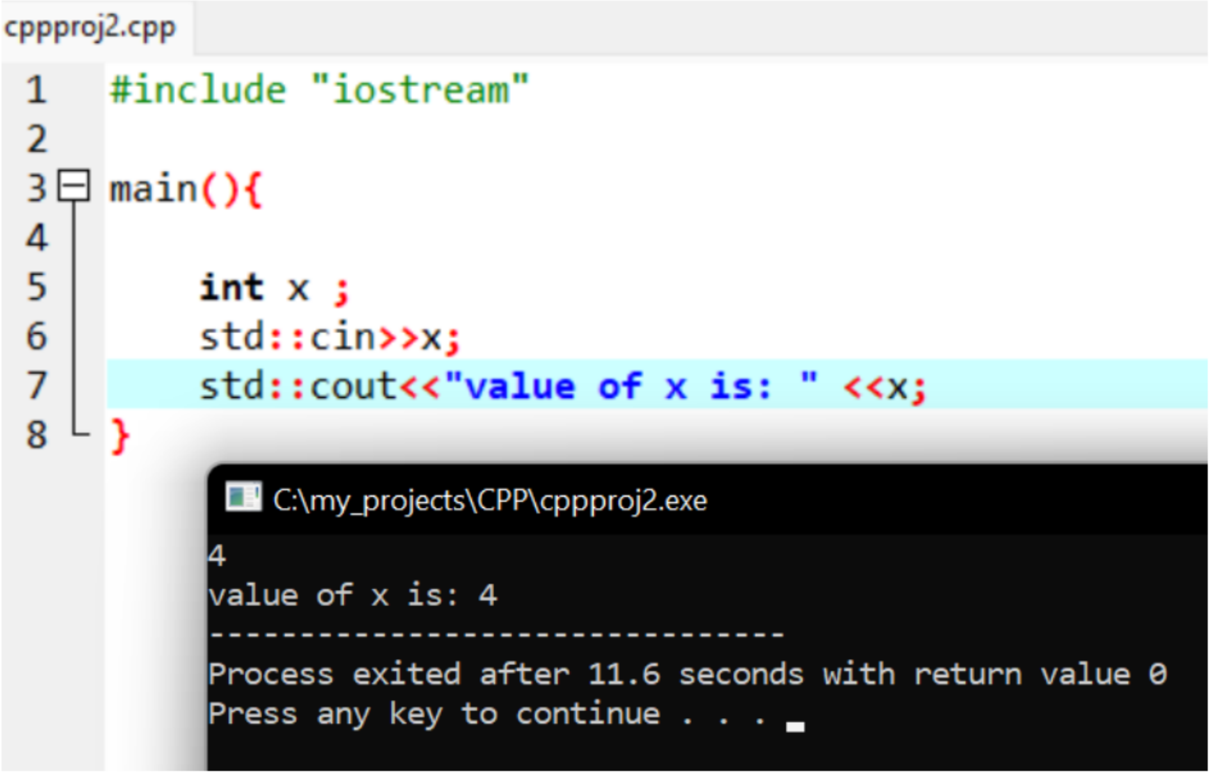
- If we declare a variable in CPP but didn’t assign a value to it, Then it will reserve some space in the RAM, but it would be filled with some garbage value, Now as we will assign some value from our side to this variable, then that garbage value will be removed, But until we are not assigning a value to this variable, it will show some garbage value, now MinGW is one of the latest compilers, so it always shows the garbage value as 0.



```cpp
#include "iostream"

main(){

    int x ;
    std::cout<<x;
}
```

```
0
--------------------------------
Process exited after 0.136 seconds with return value 0
Press any key to continue . . .
```

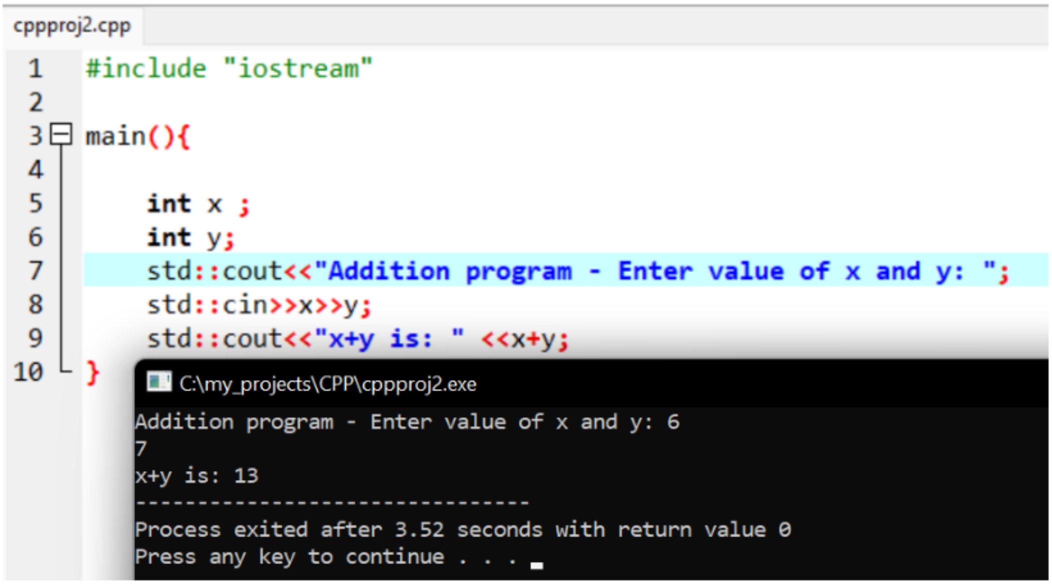
- To take input from the user in CPP, we use “**cin**” function. Here in the given example, we have taken an integer value from the user and then printed it back to the screen. (keyboard is standard input device)



```cpp
#include "iostream"

main(){

    int x ;
    std::cin>>x;
    std::cout<<"value of x is: " <<x;
}
```

```
4
value of x is: 4
--------------------------------
Process exited after 11.6 seconds with return value 0
Press any key to continue . . .
```

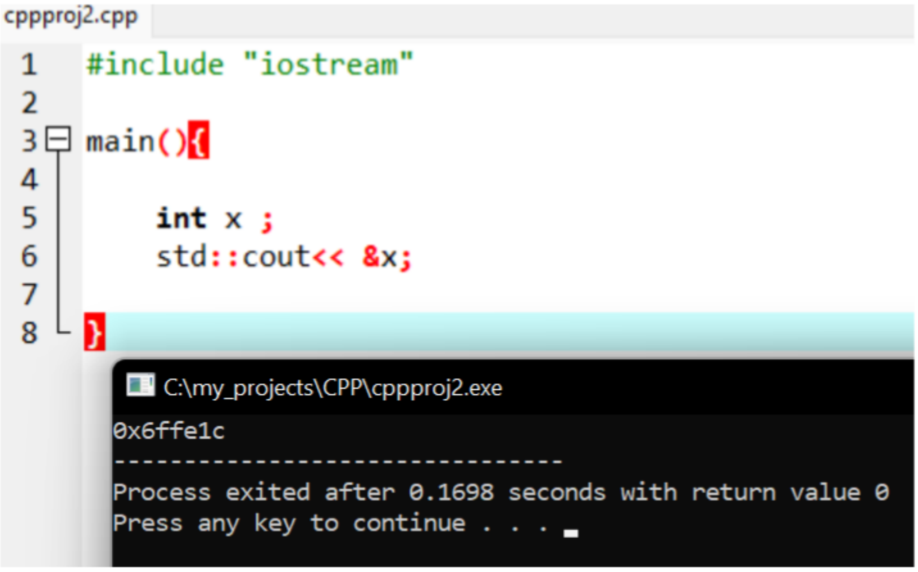
- **CPP Adition Program** -



```cpp
#include "iostream"

main(){

    int x ;
    int y;
    std::cout<<"Addition program - Enter value of x and y: ";
    std::cin>>x>>y;
    std::cout<<"x+y is: " <<x+y;
}
```

```
Addition program - Enter value of x and y: 6
7
x+y is: 13
--------------------------------
Process exited after 3.52 seconds with return value 0
Press any key to continue . . .
```

- Here I have printed the address of the “**x**” variable in hexadecimal form in the RAM

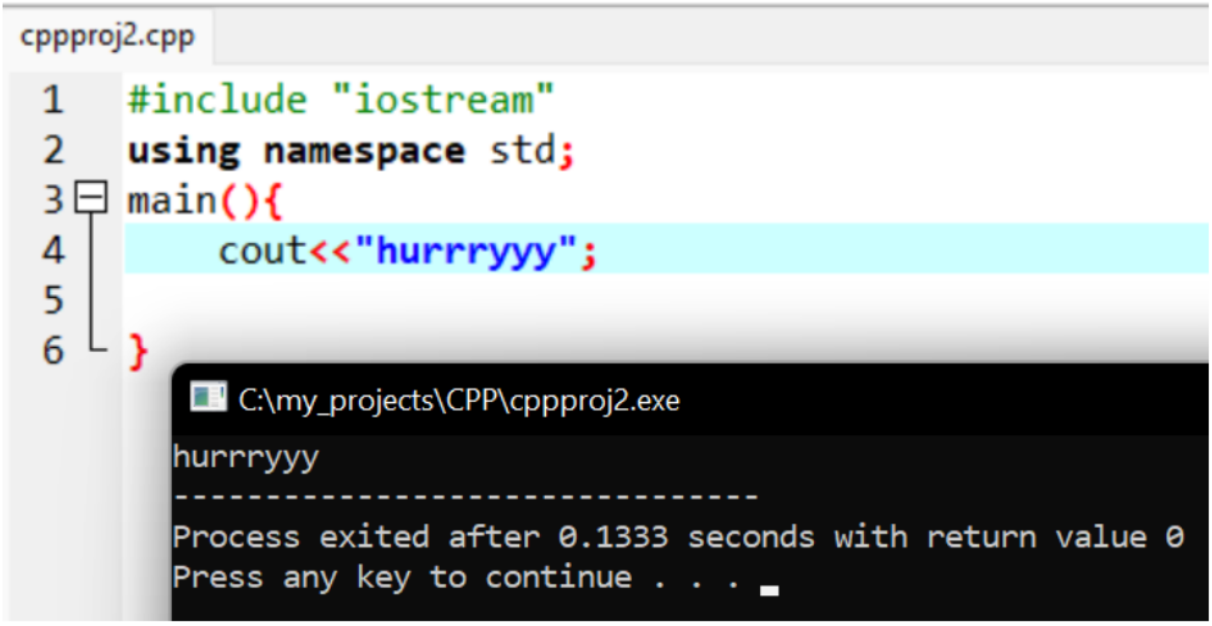


```cpp
#include "iostream"

main(){

    int x ;
    std::cout<< &x;

}
```

```
0x6ffe1c
--------------------------------
Process exited after 0.1698 seconds with return value 0
Press any key to continue . . .
```

- **Pointers** - Now, when we declare a variable and then assign some value to it, Then behind the scene it first goes and finds the address of that variable in RAM, then it went to that place, and then finally store the value. Now this is quite a long process, which makes our program slow, Now to make it fast, we can use pointers, now onward pointers will remind the address of the variable, And we will use this pointer to directly get the address and perform the action.
- Now to create a pointer -
  int x ;
  int *p= &x; *//here p is the pointer(pointer is not a normal variable) it will store the address of x.*
- ***p=10** *// Here we are assigning some value in the “x” variable using the pointer, And this operation would be very fast in comparison to if we do here (x=10).*
- **std::cout<<*p;** *- it will print the value we have in x i.e 10.*

- Now, if we don't want to use **"std::"** always while using **cout** and **cin** function, Then above the "main" function we can write this statement - **using namespace std;** And now onward we can use **cout** and **cin** function directly.



```
#include "iostream"
using namespace std;
main(){
	cout<<"hurrryyy";

}
```

```
hurrryyy
--------------------------------
Process exited after 0.1333 seconds with return value 0
Press any key to continue . . .
```