Summary CPP

Sessions No 6(18-10-2022)

- **Garbage Value** - When we turn on our System, There is an electric flow all over the system, even in RAM, As we know in the terms of electricity it would be either on or off, which in machines' terms we can say is 0,1. Now there is a lot of random 0,1 in RAM and if we try to read them then we will not get any significant data, And this we called garbage value.

- **Hexadecimal** is the name of the numbering system that is base 16. This system, therefore, has numerals 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, and 15. That means that two-digit decimal numbers 10, 11, 12, 13, 14, and 15 must be represented by a single numeral to exist in this numbering system. To address the two-digit decimal values, the alphabetic characters A, B, C, D, E, and F are used to represent these values in hexadecimal and are treated as valid numerals.

  We, humans, understand decimal number systems and computers understand binary number systems, But as humans, we can't guess the value of binary very easily(we will have to covert decimal to binary with some formulas, which take time and calculation), So here we have created one more number system that is Hexadecimal, which is some like our decimal number system only but here binary to hexadecimal conversion have patterns which we can understand very easily. That's why we use a hexadecimal number system, and even when our system has to show us any values for example the physical address of our data, it will show us in hexadecimal form.

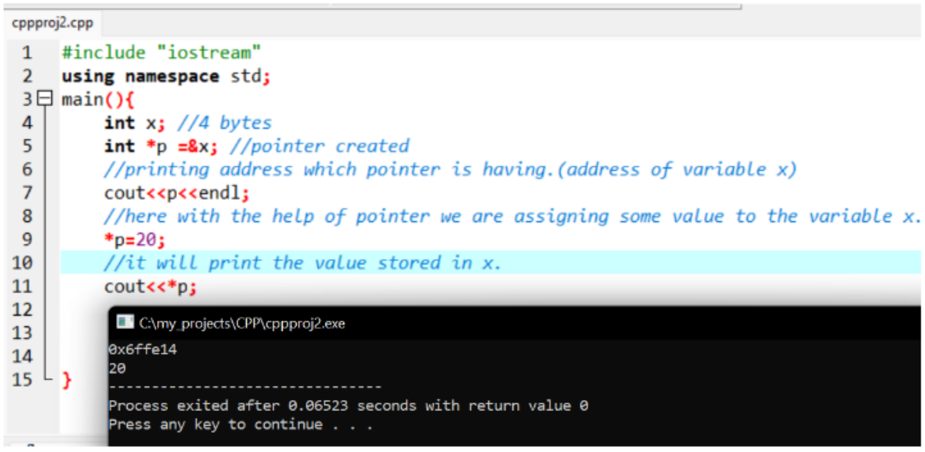
- **Pointers** - Let's see how to declare a pointer, for example -

  int x;

  int *p=&x;

  Now here our pointer **p** is going to remember the address of variable **x**, And we know the pointer is not like a normal variable it is a special variable that remembers the physical address, Hence we define it here as ***p**. Pointer only remembers the first-bit address of the variable, which means if there is a variable that takes **4 bytes** of space in RAM, then the pointer will just remember the first-bit address, Now while further using this pointer we should know till which bit I should keep on reading, and here this **"int"** help pointer that you have stored address of an integer variable in **RAM**, so first go that first bit and then from their read next 4 bytes.

- Pointers are very fast and make our program very fast. 👇



cppproj2.cpp

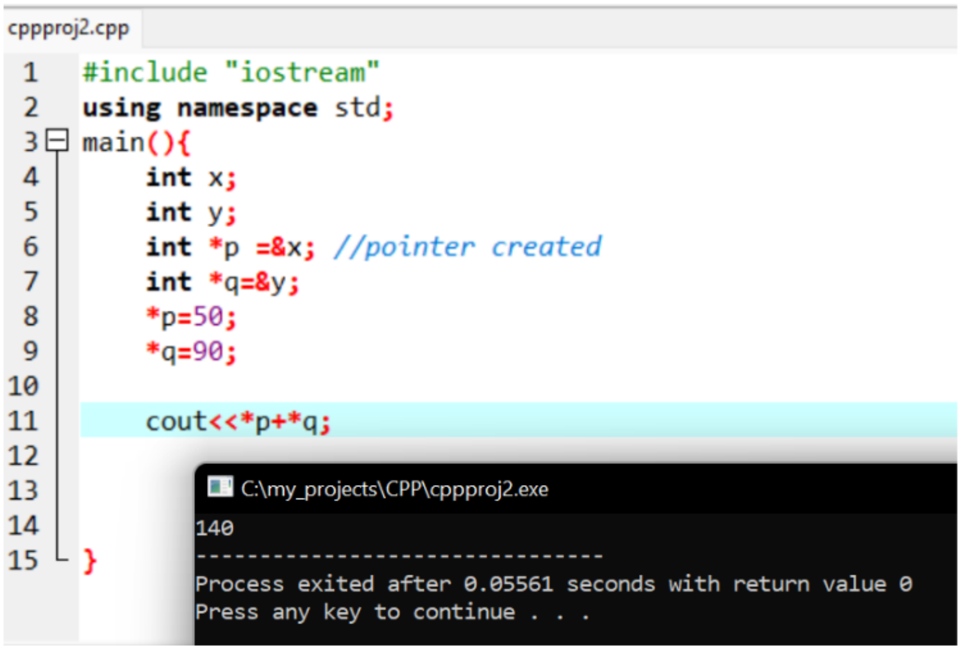
```
#include "iostream"
using namespace std;
main(){
    int x; //4 bytes
    int *p =&x; //pointer created
    //printing address which pointer is having.(address of variable x)
    cout<<p<<endl;
    //here with the help of pointer we are assigning some value to the variable x.
    *p=20;
    //it will print the value stored in x.
    cout<<*p;



}
```

C:\my_projects\CPP\cppproj2.exe

```
0x6ffe14
20
--------------------------------
Process exited after 0.06523 seconds with return value 0
Press any key to continue . . .
```

-

- Now with the above information, we can create an **addition program** using pointers, which would be very fast in comparison to the addition program we created in session 5.



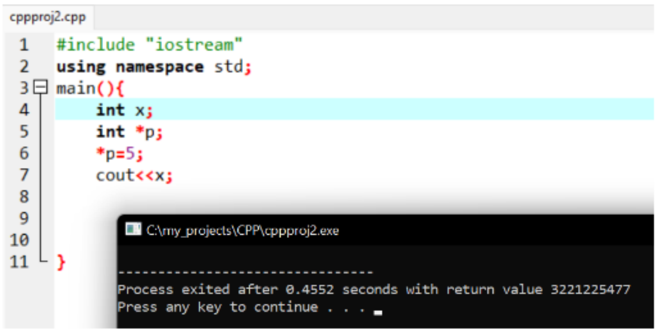
cppproj2.cpp

```
#include "iostream"
using namespace std;
main(){
    int x;
    int y;
    int *p =&x; //pointer created
    int *q=&y;
    *p=50;
    *q=90;

    cout<<*p+*q;



}
```

C:\my_projects\CPP\cppproj2.exe

```
140
--------------------------------
Process exited after 0.05561 seconds with return value 0
Press any key to continue . . .
```

- Now remember while using pointers, It is obviously very important to tell the pointer which physical address you have to remember. For example, in the below code, we haven't defined whose address to remember, and still, we try to change some value via a pointer(*p=5), So it will fail. Add **p=&x**, and then it would work fine.



cppproj2.cpp

```
#include "iostream"
using namespace std;
main(){
    int x;
    int *p;
    *p=5;
    cout<<x;



}
```

C:\my_projects\CPP\cppproj2.exe

```
--------------------------------
Process exited after 0.4552 seconds with return value 3221225477
Press any key to continue . . .
```