Edit & Run

CL-217 OBJECT ORIENTED PROGRAMMING LAB

LAB MANUAL 2

INSTRUCTOR: MUHAMMAD HAMZA

SEMESTER SPRING 2020

Pointers And Functions

As we can pass normal variables to functions, same as we can pass pointers to functions as parameters. We know that normal variables can be passed to function either by value or by reference. Same the pointers can also be passed as **By Reference** or **By Value.**

Pass By Value:

In Pass By Value the formal parameter is assigned a separate space in memory, hence it has a separate address. The formal parameter points to the same address to which the actual parameter is pointing to.

```
1. #include <iostream>
2. using namespace std;
3.
4. void fun(int * p){
5.
6. cout<<"This is function"<<endl;
7.
8. cout<<"Address of pointer p: "<<&p<<endl;
9.
10. }
11.
12. int main(){
13.
14. cout<<"This is main"<<endl;
15.
16. int * x = new int;
17.
18. cout<<"Address of pointer x: "<<&x<<endl;
19. fun(x);
20.
21. return 0;
22. }
```

Pass By Reference:

In Pass By Reference the formal parameter is not assigned a separate space in memory whereas it becomes an **alias** of the actual parameter. The formal parameter and the actual parameter have same addresses and they are hence pointing to same locatons.

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

void fun(int * &p){

cout<<"This is function"<<endl;

cout<<"Address of pointer p: "<<&p<<endl;

}

int main(){

cout<<"This is main"<<endl;

int * x = new int;

cout<<"Address of pointer x: "<<&x<<endl;
fun(x);

return 0;
}
```

Pointers as Return Type:

As we have seen in last chapter how C++ allows to return an array from a function, similar way C++ allows you to return a pointer from a function. To do so, you would have to declare a function returning a pointer as in the following example −

```
int * myFunction() {
   .
   .
   .
}
```

Second point to remember is that, it is not good idea to return the address of a local variable to outside of the function, so you would have to define the local variable as **static** variable.

Now, consider the following function, which will generate 10 random numbers and return them using an array name which represents a pointer i.e., address of first array element.

Live Demo

```
#include <iostream>
#include <ctime>

using namespace std;

// function to generate and retrun random numbers.
int * getRandom( ) {
   static int  r[10];

   // set the seed
   srand( (unsigned)time( NULL ) );

   for (int i = 0; i < 10; ++i) {
      r[i] = rand();
      cout << r[i] << endl;
   }

   return r;
}

// main function to call above defined function.
int main () {
   // a pointer to an int.
   int *p;

   p = getRandom();
   for ( int i = 0; i < 10; i++ ) {
      cout << "*(p + " << i << ") : ";
      cout << *(p + i) << endl;
   }

   return 0;
}
```

When the above code is compiled together and executed, it produces result something as follows −

```
624723190
1468735695
```

```
807113585
976495677
613357504
1377296355
1530315259
1778906708
1820354158
667126415
*(p + 0) : 624723190
*(p + 1) : 1468735695
*(p + 2) : 807113585
*(p + 3) : 976495677
*(p + 4) : 613357504
*(p + 5) : 1377296355
*(p + 6) : 1530315259
*(p + 7) : 1778906708
*(p + 8) : 1820354158
*(p + 9) : 667126415
```