Lab 16
Pointers

Dr. John P. Abraham, Professor

Until now we have talked about variables that hold values of differing data types. For example:

int x =20; The memory location referenced by x has a value of 20.

float y=21.22; The memory location referenced by y has a value of 21.22.

In early chapters we discussed that a variable name is an alias for a memory address. What if we wanted to know the address of the memory location where the inter variable x that has a value of 20 is stored? We can use the address operator &x to do that. For example, cout << &x.

In this chapter we will study of a different kind of variable, the pointer variable. **A pointer only can hold the address of a memory location.** Suppose you want to look up a definition of a term. You take a book, go to the index page and find the term. The definition of the term is not given there, but a page number. Now, you turn to that page to find the definition. Suppose you have a variable called x where an integer is stored. A pointer variable of x, let's call it pointerx, will contain the address of the memory location where the value is actually stored. The **pointer variable** pointerx **indirectly references** the value whereas the variable x references the value directly. For example:

int *pointerx; pointerx may contain a memory address which references a memory location where an integer is located; however, currently it is empty.

int x; memory location referenced by x can hold an integer.

pointerx = &x; pointerx now contains the address of the memory location where value of variable x is located. Pointerx points to x.

cout << *pointerx; Prints the value of x. The * is pronounced **dereference**.

Program 16-1

```
/********************
Program Pointer1
```

```
Objective: Pointer basics
by Dr. John Abraham
**********************/



#include <iostream>
using namespace std;

int main()
{
      int x;            //creates a variable called x to hold an
integer
      int *pointerx;    //creates a pointer variable to hold an address
where an integer can be stored
      cout << "The value for inter x will be stored in memory location:
"<<&x;
      cout << "\nEnter a value for x: "; cin >> x;
      pointerx = &x;    //pointerx gets memory location of  variable x
      cout << "You entered " << *pointerx <<endl;    //prints contents
of location pointed by pointerx.
      cout << "The address where " << x << " is stored: " <<pointerx
<<endl;
      return 0;
}
```

Program Run 16-1

```
The value for inter x will be stored in memory location: 0012FF40
Enter a value for x: 1381
You entered 1381
The address where 1381 is stored: 0012FF40
```

At this point it would be beneficial for me to talk about passing a pointer to a function. I wrote a function called findSquare. I passed the value where the pointer nPointer is pointing. Essentially what the function does is: multiply the content of memory location referenced by nPtr and multiply by the same and place the result in the memory location referenced by nPtr. You may not see the advantage of using pointers at this juncture; but it will become evident to you later.

Program 16-2

```
/********************
Objective: Pointer basics 2
by Dr. John Abraham
**********************/


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

```
void findSquare( int *);
int main()
{
      int number=5;
      cout << "The original value of number is " <<number;
      findSquare (&number);
      cout <<"\nThe new value of number is " << number << endl;
      return 0;
}
void findSquare (int *nPtr)
{
       *nPtr = *nPtr * *nPtr;
}
```

Program Run 16-2

```
The original value of number is 5
The new value of number is 25
```

The next program I want to discuss with you is one where you can have two pointers pointing to the same memory location. Here both pointerx and pointery is pointing to the same memory with the same value. This is to show you that you can assign one pointer variable to another.

Program 16-3

```
/*******************
Program Pointer1
Objective: Pointer basics
by Dr. John Abraham
**********************/



#include <iostream>
using namespace std;

int main()
{
      int x;            //creates a variable called x to hold an
integer
      int *pointerx, *pointery;     //creates a pointer variable to
hold an address where an integer can be stored
      cout << "The value for inter x will be stored in memory location:
"<<&x;
      cout << "\nEnter a value for x: "; cin >> x;
      pointerx = &x;    //pointerx gets memory location of  variable x
      pointery = pointerx;
      cout << "\nYou entered " << *pointerx <<endl;    //prints
contents of location pointed by pointerx.
```

```
	cout << "The address where " << x << " is stored: " <<pointerx
<<endl;
	cout << "\npointery is also pointing to location: " <<pointery;
	cout << "\nThe value stored in that location is: " <<*pointery;
	return 0;
}
```

Program Run 16-3

```
The value for inter x will be stored in memory location: 0012FF40
Enter a value for x: 345

You entered 345
The address where 345 is stored: 0012FF40

pointery is also pointing to location: 0012FF40
The value stored in that location is: 345
```

We can look at what is stored in each word of the memory. In the following program (Program 16-4), I incremented the memory to the next word (my machine is a 64 bit machine, therefore, 4 bytes are incremented) and printed its contents. Obviously, it should print garbage since we did not put anything there.

Program 16-4

```
/********************
Program Pointer1
Objective: Pointer basics
by Dr. John Abraham
**********************/



#include <iostream>
using namespace std;

int main()
{
	int x;			//creates a variable called x to hold an
integer
	int *pointerx, *pointery;	//creates a pointer variable to
hold an address where an integer can be stored
	cout << "The value for inter x will be stored in memory location:
"<<&x;
	cout << "\nEnter a value for x: "; cin >> x;
	pointerx = &x;	//pointerx gets memory location of  variable x
	pointery = pointerx;
```

```
      cout << "\nYou entered " << *pointerx <<endl;     //prints
contents of location pointed by pointerx.
      cout << "The address where " << x << " is stored: " <<pointerx
<<endl;
      cout << "\npointery is also pointing to location: " <<pointery;
      cout << "\nThe value stored in that location is: " <<*pointery;
      pointery++;
      cout << "\nThe value stored in the next word address location  "
<<pointery << "is: " <<*pointery;
      return 0;
}
```

Program Run 16-4

```
The value for inter x will be stored in memory location: 0012FF40
Enter a value for x: 759

You entered 759
The address where 759 is stored: 0012FF40

pointery is also pointing to location: 0012FF40
The value stored in that location is: 759
The value stored in the next word address location  0012FF44is: -858993460
```