CSCI-2275 Programming and Data Structures Recitation 4
Instructor: Rhonda Hoenigman

Learning objectives:

- Pointers
- Referencing and de-referencing
- Pointer to structures
- call by value vs. call by reference

1 Pointers

Every variable we declare in our code grabs a space in the memory. So it has an address in memory where it resides. A pointer is a memory address. By means of pointer variable we can access the address and the value at that address.

2 Address-of operator

The operator `&' is used to get the address of the variable a. Hence it is referencing operator or address-of operator.

3 Dereference operator

Dereferencing means reading in the value at the address

Consider the following code:

```
int a=10;
int *p = &a;
cout<<p;
cout<*p;
```

When you print p it will just give you the value of p. But when you add the operator * before it, it will go to the particular location and print whatever value is stored there. As we can see the *p will print the value of a rather than the address of a

4 Pointer to struct

Declaring a pointer to a struct is similar to declaring a pointer to a normal int. The fun part comes when you have to access the contents of the struct.

Consider the following code:

```
struct Distance{
	int miles;
	int feet;
	int inches;
}

int main{
	Distance d;
	Distance* p=&d;
	d.inches=6;
	p->inches=5;
	cout<<d.inches;
	cout<<p->inches;
	return 0;
}
```

5 Call by Value vs. Call by address

Call by value :When you call a function and pass something as a parameter the function creates a copy of the parameters and so if you change the values there it might not reflect if you do not return it.

Call by reference: Here we try and pass the address of the value we want to change and in this case we are passing the address of a. The function will again create a local copy of the address.However since both of this addresses are same they will refer to to the variable a. Hence changing the value at the pointer will change the value of a and that change will be persisted.

6 new and delete

Non-static and local variable get memory allocated in stack. Dynamically allocated memory is allocated in heap. We will use new operator to dynamically allocate memory. For example-

```
int a  = new int [10] ;
```

will dynamically allocates memory for 10 integers continuously.

Advantage: We can take the size of the array as user input during run time and dynamically create an array.
Cons : Once programmer allocates memory dynamically it is his/her responsibility to deallocate it. Otherwise even if the array is no longer required the space will be occupied. That will cause a memory leak.

To delete a space pointed by a pointer p we need to use-

delete p ;

If p points to an array then to free the space-
delete [ ] p ;

7 Exercise

Write down c++ program that will do as follows-
1. It will read from a text file. Each line of the file contains a number. Provide the file name as a command line argument. Number of line in the file is not known.
2. Create an array dynamically of a capacity (say 10) and store each number as you read from the file.
3. If you exhaust the array but yet to reach end of file dynamically re-size (Double the previous size) the array and keep on
adding.