7COM1025
Programming for Software Engineers
Lecture 6

Dr. Renato C. de Amorim

POINTERS AND ARRAYS

Pointer arithmetic can sometimes be faster than array indexing.

Particularly when accessing an array in strictly sequential order

```
#include<iostream>
#include<cctype>
using namespace std;
int main()
	{
	char *p;
	char str[80] = "This is a Test";
	cout<<"Original string: "<<str<<endl;
	p=str; //an array with no index returns a pointer to the 1st element
	while(*p)   {
		if(isupper(*p))
			*p=tolower(*p);
		else if (islower(*p))
			*p=toupper(*p);
		p++;
		}
	cout<<"Inverted-case string: "<<str;
	return 0;
	}
```

INDEXING A POINTER

```
#include <iostream>
#include<cctype>
using namespace std;
int main()
        {
        char *p;
        int i;
        char str[80]="This is a Test";
        cout<<"Original string: "<<str<<endl;
        p=str;
        for (i=0; p[i]; i++){
                if(isupper(p[i]))
                        p[i]=tolower(p[i]);
                else if (islower(p[i]))
                        p[i]=toupper(p[i]);
                }
        cout<<"Inverted-case string: "<<str<<endl;
        return 0;
}
```

The statement p[i] is functionally identical to *(p+i)

PROBLEM 5.1

Regarding your encryption program, we need such programs to run as fast as possible, so re-write it using pointers.

ARRAYS OF POINTERS

```
#include<iostream>
#include<cstring>
using namespace std;
int main(){
        char *dictionary[][2]={
        "pencil", "a writing instrument",
        "keyboard", "an input device",
        "", ""};
        char word[80];
        int i;
        cout<<"Enter word: ";
        cin>>word;
        for (i=0; *dictionary[i][0]; i++)
                if (!strcmp(dictionary[i][0], word)){
                        cout<<dictionary[i][1]<<endl;
                        break;
                        }
        if (!*dictionary[i][0])
                cout<<word<< " not found. "<<endl;
        return 0;
}
```

NULL POINTER CONVENTION

If you try to use a pointer that has not been initialized you may crash the computer.

These bugs can be difficult to find, a solution is to always initialize your pointers.

```
float *p=0; //this is now a null pointer
To check for a null pointer
If(p) //succeeds if p is not null
If(!p) //succeeds if p is null
Now you can easily find if a pointer has been initialized.
```

MULTIPLE INDIRECTION

```
#include<iostream>
using namespace std;
int main()
{
    int x, *p, **q;
    x=10;
    p=&x;
    q=&p;
    cout<<**q<<endl;
    return 0;
}
```

BASIC: FUNCTIONS

```
Return-type name(parameter-list)
{
	//body of function;
}
#include<iostream>
using namespace std;
void myfunc(); //prototype: allows the compiler to know return and parameters before its use.
int main()
{
	cout<<"In main()"<<endl;
	myfunc();
	cout <<"Back in the main()"<<endl;
	return 0;
}
void myfunc()
{
	cout<<"Inside myfunc()"<<endl;
}
```

BASIC: FUNCTIONS WITH PARAMETERS

```
#include<iostream>
using namespace std;
int box(int length, int width, int height);
int main()
{
    cout<<"The volume is: "<<box(7,20,4);
    return 0;
}
int box(int length, int width, int height)
{
    return length*width*height;
    //if you write a command after return, it won’t be executed.
}
```

PROBLEM 5.2

1) Write a function whose interface accepts two integers and returns the first to the power of the second.

Show that your function works for any two integers entered by the user.

2) Update your encryption program so that the encryption happens in a function. This function should receive a string and a key, and return a string. The decryption should be a different function with a similar list of parameters.

BASIC: VARIABLE SCOPE

Local variables
Declared in a block, valid inside that same block.
Not know outside their own block

```
#include<iostream>
using namespace std;
void f1();
int main(){
	int val=10;
	cout <<"Val in main "<<val<<endl;
	f1();
	cout <<"Val in main "<<val<<endl;
	return 0;
}
void f1(){
	int val=98;
	cout<<"Val in f1 "<<val<<endl;
}
```

BASIC: VARIABLE SCOPE

What would be the output of:

```
#include<iostream>
using namespace std;
void f();
int main()
{
    for (int i=0; i<3; i++) f();
    return 0;
}
void f()
{
    int num=99;
    cout<<num<<endl;
    num++;
}
```