LECTURE 19

1. Pointers and Arrays:

In C++, there is a close correspondence between array data type and pointers. An array name in C++ is very much like a pointer but there is a difference between them. The pointer is a variable that can appear on the left side of an assignment operator. The array name is a constant and cannot appear on the left side of an assignment operator. In all other respects, both the pointer and the array are the same.

Valid :

```
int value[20];
int *ptr;
ptr=&value[0];   //the address of the zeroth element is assigned to a pointer variable ptr.
ptr++ == value[1];
ptr+6 == &value[6];
*ptr == &value[0]
*ptr == &value[ ]
*(ptr+6) == value[6]
ptr ++ == &value[1]
```

Example:

```
int s[200];
int *sptr;
sptr=s;        →        sptr = &s[0];
a[i]    →    *((a) + (i))        →    *(&(a)[0] + (i))
```

Example 1:

A program to display the content of an array using a pointer arithmetic

```
#include <iostream.h>
 void main(void)
{
        static int a[4]= {11, 12, 13, 14 };
        int i, n, temp;
        n=4;
        cout<<"Contents of the array" << endl;
        for (i=0; i<=n-1; ++i)  {
        temp = *((a) + (i) );     //     or temp= *(&(a)[0] + (i));
        cout<<"value ="<<temp <<endl;
}
}
```

```
Contents of the array
Value =11
Value =12
Value =13
Value =14
```

2. Arrays of Pointers:

The pointers may be arrayed like any other data type. The declaration for an integer pointer array of size 10 is **int *ptr[10];** makes **ptr[0], ptr[1], ptr[2], … , ptr[10];**

Where ptr is an array of pointers that can be used to point the first elements of the arrays a, b, c. The following are valid assignment statements in C++:

Ptr[0] = &a[0];

Ptr[10] = &b[0];

Ptr[20] = &c[0];

Example 2:

A program to display the content of pointers using an array of pointers

```
#include <iostream.h>
 void main(void)
{
Char *ptr[3];
Ptr[0]="Ahmed";
Ptr[1]="Reem";
Ptr[2]="Ali";
Cout<<"contents of pointer 1 ="<<ptr[0]<<endl;
Cout<<"contents of pointer 2 ="<<ptr[1]<<endl;
Cout<<"contents of pointer 3 ="<<ptr[2]<<endl;  }
```

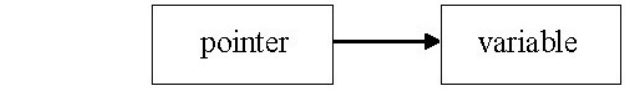
```
contents of pointer 1 = Ahmed
contents of pointer 2 = Reem
contents of pointer 3 = Ali
```

3. Pointers to Pointers:

An array of pointers is the same as pointers to pointers. As an array of pointers is easy to index because the indices themselves convey the meaning of a class of pointers. However, pointers to pointers can be confusing. The pointer to a pointer is a form of multiple of indirections or a class of pointers. In the case of a pointer to a pointer, the first pointer contains the address of the second pointer, which points to the variable that contain the values desired. Multiple indirections can be carried on to whatever extent desired, but there are a few cases where more pointer to a pointer is needed or written.

Int **ptr2;

Where ptr2 is a pointer which holds the address of the another pointer.



Example 3:

A program to declare the pointer to pointer variable and to display the contents of thesepointers

```
#include <iostream.h>
 void main(void)
{
int value;
int *ptr1;
int **ptr2;
value = 120;
cout<< "value ="<<value<<endl;
ptr1=&value;
ptr2=&ptr1;
cout<<"pointer 1="<<*ptr1<<endl;
cout<<"pointer 2="<<**ptr2<<endl;
}
```

```
Value = 120
Pointer 1 = 120
Pointer 2 =120
```