PH5720: Numerical Methods and Programming

Week - 03

How to program

A few words on C++ and STL

Prabhat R. Pujahari

Indian Institute of Technology Madras

❖ Function:

```
float AreaRectangle(float len, float wid)
{
    return len*wid;
}
```

- For normal function, the compiler creates a set of instruction. If you are calling a function, it jumps to that place and executes it and comes back to the original place.
- That means only one set of copy exists.
- Some overhead because of jumping back and forth.

❖ Analysis

❖ Function

```
float AreaRectangle(float len, float wid)
{
    return len*wid;
}
```

- Here arguments are passed by values. The calling function creates a new variable of same type as the argument and copies the argument values to the variables. The function can not access the original variables in the calling program.

- Passing arguments by value is useful when the function does not need to modify the original value in the calling program.

- If in the function, the argument value is getting changed, then it needs to be passed by reference.

❖ Reference

Reference is nothing but the address of the memory location

❖ Program

➢ Reference declaration:
& var_name

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

int main()
{
    int length = 10;

    cout << “Address of length = “ << &length << endl;

    return 0;
}
```

❖ Example: Area of rectangle – arguments passed through reference

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

void AreaRectangle(float len, float  wid, float & area)
{
   area = len * wid;
}

int main()
{
      float length = 10.;
      float wid     = 5.;
      float area   = 0.;
      AreaRectangle(len, wid, area);

      cout << “ Area of Rectangle = “ << area <<  endl;

      return 0;
}
```

- **Pointers:** These are the variables that hold address values

- **Syntax:** int *ptr;

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

int main()
{
      int length = 10;
      int *ptr;
      ptr = & length;

      *ptr = 20;

      cout << "Value of length = " << length <<  endl;
      cout << "Value of length = " << *ptr     <<  endl;

      return 0;
}
```

There are two asterisk used.

1. int *ptr: It's a pointer to integer variable

2. *ptr = 20: This is a dereference operator.

   This means that the value of the pointer pointed to by the pointer "ptr".

❖ Pointers: Example

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

int main()
{
      int length = 10;

      int *ptr;

      *ptr = 20;

      cout << “Value of length = “ << *ptr     <<  endl;

      return 0;
}
```

❖ Analysis

During compilation, no error or no warning. But while running the program, it will give “segmentation fault”.

Because the pointer defined does not contain any address value.

Note: *First the pointer should be assigned the address value.*

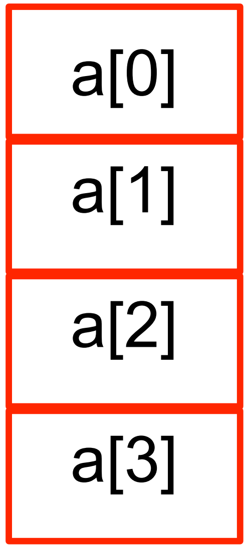
❖ Example: Area of rectangle – arguments passed through pointer

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

void AreaRectangle(float len, float  wid, float * area)
{
   *area = len * wid;
}

int main()
{
      float len     = 10.;
      float wid     = 5.;
      float area    = 0.;
      AreaRectangle(len, wid, &area);

      cout << " Area of Rectangle = " << area <<  endl;

      return 0;
}
```

A few words on C++ and STL : Pointers and Arrays

- Arrays: Syntax – int a[4];
- The name of the array is its address.
- One can use dereference operator to fetch the value stored in the memory.

How is it stored in the memory?



4th element:
a[3], or *(a+3)

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

int main()
{
    int a[5] = {1, 2, 3, 4};

    for (int i = 0; i < 4; i++)
        {
            cout << *(a+i) << endl;
        }
      return 0;
}
```

A few words on C++ and STL : new ………… delete

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

int main()
{
    int irow = 5;
    int *aa;
    aa = new int [irow];
    for (int ii = 0; ii < 5; ii++)
       {
         aa[ii] = 10;
       }
    // Delete from the memory
    delete [] aa;
    return 0;
}
```

- Example: 1-dimensional array using new ….. delete

- Syntax :

```
int *a = new int [5];

statement;

delete [] a;
```

A few words on C++ and STL : new .... delete

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

int main()
{
    int irow = 5, icol = 64;
    int **aa;
    aa = new int * [irow];
    for (int ii = 0; ii < 5; ii++) aa[ii] = new int [icol]; // aa[5][64]
    for (int ii = 0; ii < irow; ii++)
       {
         for (int jj = 0; jj < icol; jj++)
            {
               aa[ii][jj] = 10;
            }
       }
    // Delete from the memory
    for (int ii = 0; ii < irow; ii++) delete [] aa[ii];
    delete [] aa;
    return 0;
}
```

❖ Example: 2-d array using new ..... delete