Weeks 14 – CS 101 –J

Pointers and File Handling

Remaining Topics

- Till now, we have studied pointers, then strings
- This week we will study dynamic memory management, followed by file handling
- Let us start with 2 simple examples
- Tell me the output of Example 1 and Example 2 – you may use compilers if you want

Example 1

```
#include <iostream>
using namespace std;
int main()
{
        cout<<"Hello"<<endl;
        cout<<"Hi"<<endl;
        return 0;
}
```

Example 2

```
#include <iostream>
using namespace std;
int main()
{
        cout<<"Hello"<<endl;
        int marks[5000000];
        cout<<"Hi"<<endl;
        return 0;
}
```

Dynamic Memory Management

- There is no output of Example 2
- Why?

- Answer: There is not enough memory available.

- Next Big Question: What is the solution then?

Dynamic Memory Management

- C++ enables programmers to control the allocation and de-allocation of memory in a program for any built-in or user-defined type

- Performed with operators **new** and **delete**

- Through dynamic memory, we can allocate array space and we can free space also (when not required)

Example 3

```
#include <iostream>
using namespace std;
int main()
{
        cout<<"Hello"<<endl;
        int *marks = new int[5000000];
        delete [] marks;
        cout<<"Hi"<<endl;
        return 0;
}
```

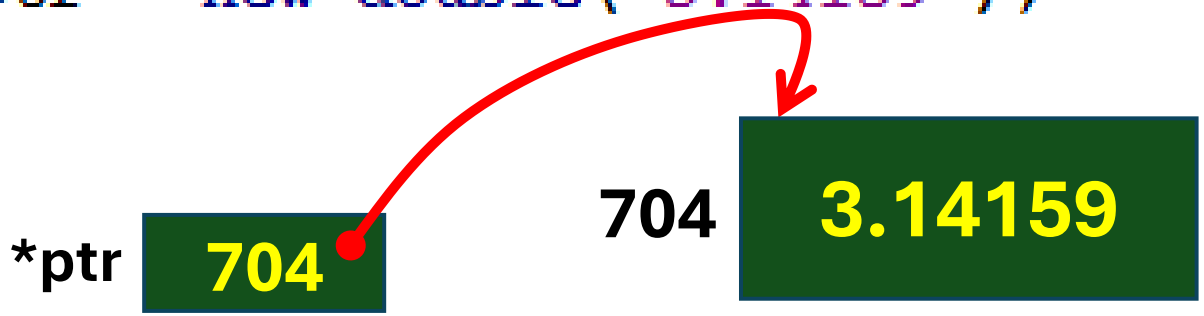
Dynamic Memory

- Dynamically allocating memory in this fashion causes an array to be created in the **free store** (**heap**)
- Heap is a region of memory assigned to each program for storing objects created at execution time
- Once the memory is allocated in the free store, pointer points to the first byte of that allocated memory
- After used, memory can be return to heap by using delete operator

Dynamic memory – other use of new

- C++ allows you to provide an initializer for a newly created fundamental-type variable

```
double *ptr = new double( 3.14159 );
```

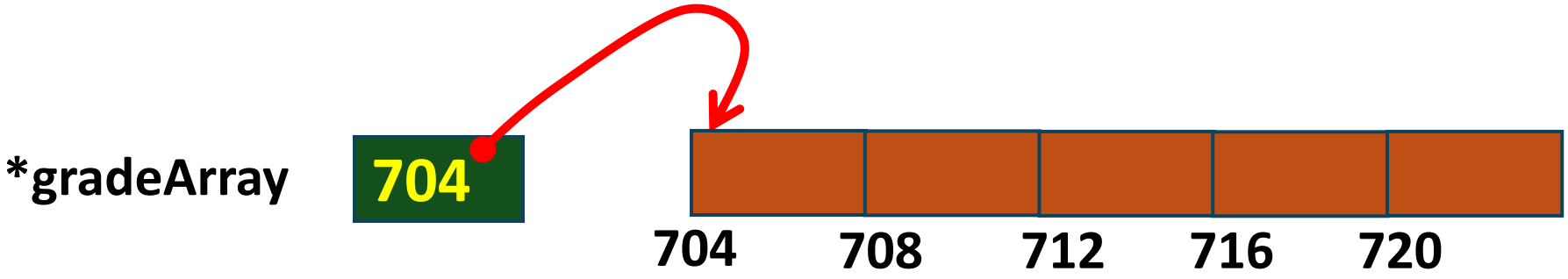



```
delete ptr;
```

Dynamic memory – array

- **new** operator can be used to allocate arrays dynamically

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




```
delete [] gradesArray;
```

File Handling

Why to use Files?

- Storage of Data in variables is temporary
- Files are used for permanent storage of data
- Files are stored on secondary storage devices

Methods to Access Files

- Sequential
- Random

Files and Streams

- When a file is opened, an object is created and a stream is associated with that object.

File Handling – Reading data

- To read some data from a file, we use the following commands

```
ifstream fin ;                    // Create file INPUT stream object
fin.open("my_input.dat") ;        //Open input file
fin>>a>>b ;                       //Read two values from input file
```

File Handling – Writing data

- To write some data from a file, we use the following commands

```
ofstream fout ;                  //Create file OUTPUT stream object
fout.open("my_output.dat");      //Open output file
fout<<c<<endl ;                  //Write result to output file
```

Execute and Understand wk14s14.cpp

```
#include <fstream>
using namespace std;
int main()
{
        int a, b, c ;
        ifstream fin ;                  // Create file INPUT stream object
        ofstream fout ;                 //Create file OUTPUT stream object

        fin.open("my_input.dat") ;      //Open input file
        fin>>a>>b ;                     //Read two values from input file
        c = a + b ;

        fout.open("my_output.dat");             //Open output file
        fout<<c<<endl ;                         //Write result to output file

        fin.close() ;           //Close input file
        fout.close() ;          //Close output file
}
```

File Handling

- You are familiar with cout and cin statements
- cout statement causes the message to be displayed to screen/monitor whereas fout statement causes the message to be written to file
- cin statement is used to input from keyboard and fin statement is used to input from file
- Let us continue with writing 2 programs – one which reads from user **keyboard** and writes it to a file – and another one which reads from the file and displays it

wk14s16.cpp

```
#include <fstream>
#include <iostream>
using namespace std;
int main()
{
        int a;
        cout<<"Please enter a number ";
        cin>>a;

        ofstream fout ;    //Create file OUTPUT stream object
        fout.open("file_out.dat");          //Open output file
        fout<<a<<endl ;  //Write result to output file
        fout.close() ;       //Close output file

        return 0;
}
```

wk14s17.cpp

```
#include <fstream>
#include <iostream>
using namespace std;
int main()
{
        int b;

        ifstream fin ;
        fin.open("file_out.dat");
        fin>>b;
        fin.close() ;

        cout<<"The number read from file is "<<b;

        return 0;

}
```

Explanation of wk14s16.cpp file

- First, we declare an integer **a** , and input a from user using **cin** command

```
int a;
cout<<"Please enter a number ";
cin>>a;
```

- Next, we use ofstream to create fout object

```
ofstream fout ;
```

- Remember: fout is an object name – not a keyword. You can use another object name also (but fout is meaningful name for output stream)

Explanation of wk14s16.cpp file

- Now that we have file output stream object (fout), we need to link it with a file – specify the file name which we want to open

```
fout.open("file_out.dat");      //Open output file
fout<<a<<endl ;
```

- Once everything is written, we need to close the output stream now

```
fout.close() ;   //Close output file
```

wk14s20.cpp

```
#include <fstream>
#include <iostream>
using namespace std;
int main()
{
        int a;
        cout<<"Please enter a number ";
        cin>>a;

        ofstream f ;        //Create file OUTPUT stream object
        f.open("file_out.dat");      //Open output file
        f<<a<<endl ;        //Write result to output file
        f.close() ;         //Close output file
        return 0;
}
```

- Now lets play with the file, and change the output file object to f (or any valid alphabet from A to Z (or a to z))

wk14s20.cpp

- You may change the file object name to b, c, d, or any alphabet (<u>its legal - no compiler errors)</u> and donot be confused in quiz/exam)

BUT

- fout is a meaningful name when we are writing to file
- fin is a meaningful name when we are reading from file

Append mode

- Now, when you run wk14s16.cpp code multiple times, you will notice that the contents of the file file_out.dat are overwritten (previous ones are destroyed)

- Question: How to preserve the previous contents?

- Answer: Use append mode

wk14s23.cpp

```
#include <fstream>
#include <iostream>
using namespace std;
int main()
{
        int a;
        cout<<"Please enter a number ";
        cin>>a;

        ofstream f ;
        f.open("file_out.dat",ios::app);
        f<<a<<endl ;
        f.close() ;
        return 0;
}
```

When we ae using append mode in code, then the contents are written at the end of file (instead of being overwritten OR previous contents being destroyed)

wk14s24.cpp – Difficult code to digest

```
#include <fstream>
#include <iostream>
#include <cstring>
using namespace std;
int main()
{
        string a;
        cout<<"Please enter some text";
        cin>>a;

        ofstream Myfile;
        Myfile.open("file1.txt");
        Myfile<<a<<endl ;
        Myfile.close() ;
        return 0;
}
```

- A more complex code where understanding it is bit difficult
- Question: What is MyFile object?
- Answer: It is associated with output stream
- Observation: The file name here is Myfile.txt (not .dat file)

- Thank You

- Next Week is Revision Week