IB9N7 C++ for Quantitative Finance
Worksheet 11

Resources

28 January 2016
(Week 17)

Objectives for this lab session

By the end of this session, you should have completed the following:

- Work through some examples of pointers.

Exercises

Exercise 1: Writing addresses to a file

(a) Make a `vector<short>` called `a` which contains the numbers from 1 to 50. For each element write it and its address to a single line of the file `a.txt`. Note that pointers will appear in hexadecimal notation. Check you understand why the addresses of each element differ by the amount they do. What would you see with doubles?

(b) Make a vector `b` which contains the address of each element of `a`.

(c) Change your code so that you write to a file called `:.txt`. Note that your program will not write to this file. This is because in windows it is not permissible to have filenames containing the : character. However your program will run without error. This is because the `ofstream` object enters an error state where it does nothing when it cannot open the file (e.g. because the file is in use or has a bad name). You can check for this error state in several ways, for example by calling `is_open`. We saw similar things early in the course with `cin`.

Exercise 2: References simulated with pointers

Rewrite the following code using pointers in place of all references.

```
#include<iostream>
void assignDouble(int a, int& b){
    b = a+a;
}
int main(){
    int a=100, b=37, c=3;
    int& d = c;
    assignDouble(a,b);
    std::cout<<b<<"\n";
    assignDouble(d,a);
    std::cout<<d<<"␣"<<a<<"\n";
}
```

Exercise 3: const pointers

Just like with references, pointers can be marked const to indicate that they cannot be used to change the pointee. In addition, pointers can be marked const after the * to indicate that they cannot be reassigned. For example:

```
int a = 4;
int aa = 4;
const int b = 9;

const int* c = & b;
c=&a;  //I can change c
//*c = 32; //This would be illegal

int* const d = &a;
*d = 54;  //I can change a through d
//d = &aa; //This would be illegal
c=d;  //This is fine
```

(a) Check all of this makes sense

(b) (Hard) Given

```
int ** a;
const int **b;
```

I cannot make the assignments b=a, *b=*a and *a=*b. Can you understand why all of these need to be banned?

Exercise 4: recursive structure

Consider an investment firm which divides its holdings into portfolios. Each portfolio may contain its own financial assets and other portfolios. We might represent the value of a portfolio with the following object structure:

```
#include<memory>
#include<vector>
#include<string>

using std::string;
class Portfolio{
    std::vector<std::unique_ptr<Portfolio>> m_subportfolios;
    double m_valueExcludingSubportfolios;
    string m_name;
public:
    double getValueExcludingSubportfolios() const {return m_valueExcludingSubportfolios
     ;}
    string getName() const {return m_name;}
    Portfolio(string name, double valueExcludingSubportfolios);
};

Portfolio::Portfolio(string name, double valueExcludingSubportfolios)
    :m_name(name), m_valueExcludingSubportfolios(valueExcludingSubportfolios){}

int main(){
    Portfolio allAssets("total",0);
    //....
}
```

(a) Add a member function to add a subportfolio to a portfolio. Be careful with ownership.

(b) Add a member function to get a reference to the portfolio's nth subportfolio.

(c) Add a member function to get the total value of the portfolio.

(d) In fact, the company has portfolios A and B. Portfolio A has assets with value 4 and also contains a subportfolio called C. Portfolio B has subportfolios D and E. The value in portfolio C, D and E is 47, 2, and 343 respectively. Set all this in the main function.

(e) Write a function which prints for a portfolio the value of each of its subportfolios recursively.