03 Template Dynamic Array

jjcao

Mostly Based on the assignment of Professor Ligang Liu

Problem

- 现在Efficient的动态数组已经比较好了，用户使用起来很方便。但是，新的问题又出来了。
- 上述的动态数组只能存储double类型的数据。想象一下，另一个用户也想用这个动态数组，但是他处理的数据是int类型的，该如何办？
- 当然，你可以再写一个基于int类型的动态数组类，你发现，大部分的代码都不要改动，只要将类型double变成int即可。
- 如果另一个用户需要处理的是float类型，还有用户处理的是char类型的….该怎么办？

Solution

- 这样做的缺点很显然：
  1. 这些类的代码大同小异，代码冗余；
  2. 当对这个类进行修改时，改动的地方太多，不利于代码维护；

- 有无办法解决这个问题？即，如果能把变量的“类型”也当作一个“变量”，不就可以了吗？这个在C中是不能解决的，但是在C++中，有一种新的机制template，就可以解决上述问题。

- Template的学习是“泛型编程”的基础，务必要掌握！

New Interface

```
template<class T>
class TemplateArray
{
private:
    T* m_data; // the pointer to the array memory
    int       m_size; // the size of the array
    int       m_capacity; // the max memory of the array

public:
    TemplateArray():m_data(0),m_capacity(0),m_size(0){} // default constructor
    TemplateArray(int size, T value = 0); // other constructor, set an array with default values.
    TemplateArray(const TemplateArray& ba) { ... }
    template< class T1> TemplateArray(const TemplateArray<T1>& ba) { ... }
    TemplateArray& operator = (const TemplateArray& array) { ... }
    template< class T1> TemplateArray& operator = (const TemplateArray<T1>& array) { ... }
    ~TemplateArray(){ delete[] m_data;} // deconstructor
```

New Interface

```
public:
    int size() const { return m_size;}        // get the size of the array
    int capcity() const {return m_capacity;}

    T  at(int ind);             // get an element at an index
    T  operator[] (int ind) const;  // overload "[]" operator, get an element, such as T tmp = a[k]
    T& operator[] (int ind) ; // overload "[]" operator, set value of specified position, such as a[k]=3.14;

    int push_back(T elem);      // add a new element at the end of the array, return the size of the array.
    int insert(int ind, T value); // insert a new element at some index, return the size of the array
    void erase(int ind);        // delete an element at specified index,

    void print(); // print all elements
```

More Templates

```
jj::TemplateArray<int> a(1);
typedef jj::TemplateArray<double> DBArray;
DBArray b = a;  //call copy constructor
```

```
template< class T1> TemplateArray(const TemplateArray<T1>& ba)// copy constructor
{
   m_size = ba.size();
   m_capacity = ba.capcity();
   m_data = new T[m_capacity];

   for (int i = 0; i < m_capacity; ++i)
   {
      m_data[i] = ba[i];
   }
}
```

A template project with requirements

1. 实现hw03_TemplateDynamicArray_Template要求的高效动态数组
   - 实现Bubble Sort算法

a. 完成满足上述接口的动态数组(Dynamic array)的程序，递交工程文件(*.vcxproj, *.vcxproj.user, (旧的版本*.dsp)) 和源文件(*.h, *.cpp)；
b. 工程目录中的debug目录删除掉，其他文件压缩打包发给我，参照我们提供的code template。

评分标准	分数
程序成功运行（通过所有assert）	60
正确使用new/delete，不出现内存泄漏	10
需要遵循基本的编程规范和风格；	20
按时完整的提交项目文件，不包含无用文件	10

抄袭则 平均分配 成绩 = ?/n

通过完成该作业需达到的目标：

- 体会template的好处
- 快速的STL入门：The Use of STL and STL Extensions in CGAL
- STL字典：The C++ Standard Library