Report of Assignment 8

-----Template Class Array

Meng Wei

In this assignment, we set our Array a = [0.5, 1.5, 2.5, 3.5, 4.5], then we write a main function to test our Array with type int, float and double. The results are as follows:

Results:

Original elements of Array a	Int Type	Float Type	Double Type
0.5	0	0.5	0.5
1.5	1	1.5	1.5
2.5	2	2.5	2.5
3.5	3	3.5	3.5
4.5	4	4.5	4.5

PS: Codes as follows:

1.

```
// Template Class Array.h : Defines the entry point for the console application.
//

#pragma once
#include "stdafx.h"
#include <math.h>
#include <iostream>
using namespace std;


template <class Type>
class Array
{
    friend ostream & operator << (ostream &, Array<Type> &);//operator <<
public:
    Array();
    ~Array();
    Array(Array &);

    void push_back(Type i);
```

```
    void pop_back();
    void remove(int i);
    void insert(int n, Type i);

    int capacity() { return cap; };
    int size() { return num; };
    void clear() { num = 0; };



    Array<Type> operator+ (const Array<Type>& a); // Concatenation
    Type& operator [] (int); //operator []

private:
    int num;
    Type * data;
    int cap;
    void check_capacity();
};

template <class Type>
Array<Type>::Array() //default constructor
{
    num = 0;
    cap = 1;
    data = new Type[1];
}

template <class Type>
Array<Type>::~Array()//default destructor
{
    delete[] data;
}

template <class Type>
Array<Type>::Array(Array<Type> & a)//copy constructor
{
    num = a.num;
    cap = a.cap;
    data = new Type[cap];
    memcpy(data, a.data, sizeof(Type)*num);
}

template <class Type>
```

```
void Array<Type>::check_capacity() //define check_capacity function
{
    if (num >= cap) {  //not enough cap, double it
        cap *= 2;
        Type * tmp = new Type[cap];
        memcpy(tmp, data, sizeof(Type)*num);
        delete[] data;
        data = tmp;
    }
}

template <class Type>
void Array<Type>::push_back(Type i)//define push_back function
{
    check_capacity();
    data[num] = i;
    num++;
}

template <class Type>
void Array<Type>::pop_back()//define pop_back function
{
    num--;
    if (num < 0) num = 0;
}

template <class Type>
void Array<Type>::remove(int i)//define remove function
{
    if (i >= 0 && i < num) {
        int j;
        for (j = i; j < num - 1; j++) data[j] = data[j + 1];
        num--;
    }
}

template <class Type>
void Array<Type>::insert(int n, Type i)//define insert function
{
    if (n >= 0 && n < num) {
        check_capacity();
        int j;
        for (j = num; j > n; j--) data[j] = data[j - 1];
        data[n] = i;
```

```
        num++;
    }
}

//define operator + for Array Class
template <class Type>
Array<Type> Array<Type>::operator+(const Array<Type> & a)
{
    Array<Type> temp = *this;
    if (temp.num == a.num)
    {
        for (int i = 0; i< a.num; i++)
        {
            temp.data[i] = temp.data[i] + a.data[i];
        }
    }

    return temp;
}


//define operator [] for Array Class
template <class Type>
Type& Array<Type>::operator [] (int subscript)
{
    if (subscript >= 0 && subscript < num)
        return data[subscript];
}

//define friend function operator << for Array Class
template <class Type>
ostream & operator << (ostream & output, Array<Type> & a)
{
    for (int i = 0; i < a.size(); i++)
        return (output << a.data[i]);
}
```

2. // Template Class Array.cpp : Defines the entry point for the console application.
//

```
#pragma once
#include "stdafx.h"
```

```cpp
#include <math.h>
#include <iostream>
#include "TemplateClassArray.h"
using namespace std;

int _tmain(int argc, char * argv[])//for testing
{
    Array<int> a;
    //Array<float> a;
    //Array<double> a;
    a.push_back(0.5);
    a.push_back(1.5);
    a.push_back(2.5);
    a.push_back(3.5);
    a.push_back(4.5);
    a.remove(2);
    a.insert(3, 10.5);

    cout <<"a: size is: "<< a.size()<<" " << "a: capacity is: "<<a.capacity() <<
endl;

    Array<int> b = a;
    //Array<float> b = a;
    //Array<double> b = a;
    cout <<"b: size is: "<< b.size() << " " << "b: capacity is: " <<b.capacity()
<< endl;

    //test the operator + and << of Array
    cout << "the elements of Array a+b are: \n";
    for (int i = 0; i < a.size(); i++)
    {
        cout << (a + b)[i] << endl;
    }

    //test the operator [] for Array a
    cout << "the elements of Array a are: \n";
    for (int i = 0; i < a.size(); i++)
    {
        cout << a[i] << endl;
    }

    return 0;
}
```