第二次实验报告

20338109
姜博恩

实验一: 编写程序，并测试算法的执行时间

1. 实验目的

用伪代码和C++描述算法, 分析时间复杂度.

2. 实验内容

- 对一个整型数组$A[n]$设计一个排序算法.
- 找出整型数组$A[n]$的最大值和次最大值.

3. 设计与编码

1. 本实验用到的理论知识

- 排序算法: 冒泡排序
- 时间复杂度分析

2. 算法设计

先用随机数生成器生成随机数组.

它重复地走访过要排序的元素列，依次比较两个相邻的元素，如果顺序错误就把他们交换过来。走访元素的工作是重复地进行直到没有相邻元素需要交换，也就是说该元素列已经排序完成。

选最大的两个元素只需要循环两次即可.

用伪代码描述:

```
1. 输入数组和数组长度
2. 循环(数组长度 - 1)次
     依次比较相邻两个元素大小
     若靠前的元素比靠后的元素小则交换两个元素
```

3. 编码

排序代码

```cpp
#include <iostream>
#include <time.h>
#include <stdlib.h>
using namespace std;

const int Max = 20;
const int ITERATION = 100000;
void BubbleSort(int a[], int len);
void CreatRand(int a[], int len);
```

```
int main()
{
    int a[Max + 1] = {0}, b[Max + 1] = {0};
    CreatRand(a, Max);
    for (int i = 0; i <= Max; i++)
    {
        b[i] = a[i];
    }
    cout<<"For the random series: "<<endl;
    for (int i = 1; i <= Max; i++)
    {
        cout<<b[i]<<" ";
    }
    cout<<endl;
    clock_t start, end;
    double total;
    start = clock();
    BubbleSort(b, Max);
    end = clock();
    total = (double)(end - start) / CLOCKS_PER_SEC;
    cout<<"After Bubble Sort, now the series: "<<endl;
    for (int i = 1; i <= Max; i++)
    {
        cout<<b[i]<<" ";
    }
    cout<<endl;
    cout<<"Bubble sort, total = "<<total<<endl;
    start = clock();
    for (int iter = 0; iter < ITERATION; iter++)
    {
        CreatRand(a, Max);
    }
    end = clock();
    double time_1 = (double)(end - start) / CLOCKS_PER_SEC;
    for (int iter = 0; iter < ITERATION; iter++)
    {
        CreatRand(a, Max);
        BubbleSort(b, Max);
    }
    end = clock();
    double time_2 = (double)(end - start) / CLOCKS_PER_SEC;
    double average = (time_2 - time_1) / ITERATION;
    cout<<"Average time after "<<ITERATION<<" iterations: "<<average<<" s."
<<endl;
    return 0;
}

void CreatRand(int a[], int len)
{
    srand(time(NULL));
    for (int i = 1; i <= len; i++)
    {
        a[i] = 1 + rand() % 100;
    }
}

void BubbleSort(int a[], int len)
{
```

```
    int temp;
    for (int j = 0; j < len; j++)
    {
        for (int i = 1; i < len - j; i++)
        {
            if (a[i] > a[i + 1])
            {
                temp = a[i];
                a[i] = a[i + 1];
                a[i + 1] = temp;
            }
        }
    }
}
```

取前两位最大的, 代码

```
#include <iostream>
#include <time.h>
#include <stdlib.h>
using namespace std;

const int Max = 20;
const int ITERATION = 100000;
const int FIND = 2;
void BubbleFind(int a[], int len, int goal);
void CreatRand(int a[], int len);

int main()
{
    int a[Max + 1] = {0}, b[Max + 1] = {0};
    CreatRand(a, Max);
    for (int i = 0; i <= Max; i++)
    {
        b[i] = a[i];
    }
    cout<<"For the random series: "<<endl;
    for (int i = 1; i <= Max; i++)
    {
        cout<<b[i]<<" ";
    }
    cout<<endl;
    clock_t start, end;
    double total;
    start = clock();
    BubbleFind(b, Max, FIND);
    end = clock();
    total = (double)(end - start) / CLOCKS_PER_SEC;
    cout<<"The largest & the second largest: "<<endl;
    for (int i = Max; i > Max - FIND; i--)
    {
        cout<<b[i]<<" ";
    }
    cout<<endl;
    cout<<"Bubble sort, total time = "<<total<<endl;
    start = clock();
    for (int iter = 0; iter < ITERATION; iter++)
```

```
    {
        CreatRand(a, Max);
    }
    end = clock();
    double time_1 = (double)(end - start) / CLOCKS_PER_SEC;
    for (int iter = 0; iter < ITERATION; iter++)
    {
        CreatRand(a, Max);
        BubbleFind(b, Max, FIND);
    }
    end = clock();
    double time_2 = (double)(end - start) / CLOCKS_PER_SEC;
    double average = (time_2 - time_1) / ITERATION;
    cout<<"Average time after "<<ITERATION<<" iterations: "<<average<<" s."
<<endl;
    return 0;
}

void CreatRand(int a[], int len)
{
    srand(time(NULL));
    for (int i = 1; i <= len; i++)
    {
        a[i] = 1 + rand() % 100;
    }
}

void BubbleFind(int a[], int len, int goal)
{
    int temp;
    for (int j = 0; j < goal; j++)
    {
        for (int i = 1; i < len - j; i++)
        {
            if (a[i] > a[i + 1])
            {
                temp = a[i];
                a[i] = a[i + 1];
                a[i + 1] = temp;
            }
        }
    }
}
```

4. 运行与测试

排序运行结果如图所示.

```
For the random series:
74 54 75 66 99 23 93 58 7 35 78 52 51 83 20 10 16 43 30 7
After Bubble Sort, now the series:
7 7 10 16 20 23 30 35 43 51 52 54 58 66 74 75 78 83 93 99
Bubble sort, total = 0
Average time after 100000 iterations: 7.1e-007 s.
```

单次运行时间过短, 选择重复10000次取平均值, 得出一次时长约为7.1×10^{-7}秒.

由于本算法有两重到n的循环因此算法时间复杂度为$O(n^2)$.

选前两位运行结果如图所示.

```
For the random series:
79 74 83 32 20 83 74 35 98 56 44 53 4 96 65 65 64 22 52 8
The largest & the second largest:
98 96
Bubble sort, total time = 0
Average time after 100000 iterations: 4.5e-007 s.
```

单次运行时间过短, 选择重复10000次取平均值, 得出一次时长约为4.5×10^{-7}秒.

由于本算法有2次到n的循环因此算法时间复杂度为$O(n)$.

5. 总结与心得

本实验练习了随机数生成, 实现了冒泡排序算法并分析了其时间复杂度, 并在使用clock_t类给出了单次运行时间.

实验二: 用模板实现排序算法

1. 实验目的

复习函数模板.

2. 实验内容

用函数模板实现实验一中的算法.

3. 设计与编码

1.本实验用到的理论知识

- 模板函数的编写

2. 算法设计

- 只需要把前面题的函数改为模板函数.

3. 编码

排序

```
#include <iostream>
#include <time.h>
#include <stdlib.h>
using namespace std;

const int Max = 5;

template <typename T>
void BubbleSort(T a[], int len)
{
    T temp;
    for (int j = 0; j < len; j++)
```

```
    {
        for (int i = 1; i < len - j; i++)
        {
            if (a[i] > a[i + 1])
            {
                temp = a[i];
                a[i] = a[i + 1];
                a[i + 1] = temp;
            }
        }
    }
}

void CreatRand(int a[], int len)
{
    srand(time(NULL));
    for (int i = 1; i <= len; i++)
    {
        a[i] = 1 + rand() % 100;
    }
}

template <typename T>
void CreatRandx(T c[], int len)
{
    srand(time(NULL));
    for (int i = 1; i <= len; i++)
    {
        c[i] = rand() / T(RAND_MAX);
    }
}


int main()
{
    int a[Max + 1] = {0}, b[Max + 1] = {0};
    CreatRand(a, Max);
    for (int i = 0; i <= Max; i++)
    {
        b[i] = a[i];
    }
    cout<<"For the int random series: "<<endl;
    for (int i = 1; i <= Max; i++)
    {
        cout<<b[i]<<" ";
    }
    cout<<endl;
    BubbleSort(b, Max);
    cout<<"After Bubble Sort, now the series: "<<endl;
    for (int i = 1; i <= Max; i++)
    {
        cout<<b[i]<<" ";
    }
    cout<<endl;

    float c[Max + 1] = {0.0}, d[Max + 1] = {0.0};
    CreatRandx(c, Max);
    for (int i = 0; i <= Max; i++)
```

```
    {
        d[i] = c[i];
    }
    cout<<"For the float random series: "<<endl;
    for (int i = 1; i <= Max; i++)
    {
        cout<<d[i]<<" ";
    }
    cout<<endl;
    BubbleSort(d, Max);
    cout<<"After Bubble Sort, now the series: "<<endl;
    for (int i = 1; i <= Max; i++)
    {
        cout<<d[i]<<" ";
    }
    cout<<endl;

    double e[Max + 1] = {0.0}, f[Max + 1] = {0.0};
    CreatRandx(e, Max);
    for (int i = 0; i <= Max; i++)
    {
        f[i] = e[i];
    }
    cout<<"For the double random series: "<<endl;
    for (int i = 1; i <= Max; i++)
    {
        cout<<f[i]<<" ";
    }
    cout<<endl;
    BubbleSort(f, Max);
    cout<<"After Bubble Sort, now the series: "<<endl;
    for (int i = 1; i <= Max; i++)
    {
        cout<<f[i]<<" ";
    }
    cout<<endl;
    return 0;
}
```

选最大的两个

```
#include <iostream>
#include <time.h>
#include <stdlib.h>
using namespace std;

const int Max = 10;

const int FIND = 2;

template <typename T>
void BubbleFind(T a[], int len, int goal)
{
    T temp;
    for (int j = 0; j < goal; j++)
    {
        for (int i = 1; i < len - j; i++)
```

```
        {
            if (a[i] > a[i + 1])
            {
                temp = a[i];
                a[i] = a[i + 1];
                a[i + 1] = temp;
            }
        }
    }
}

void CreatRand(int a[], int len)
{
    srand(time(NULL));
    for (int i = 1; i <= len; i++)
    {
        a[i] = 1 + rand() % 100;
    }
}

void CreatRandDouble(double c[], int len)
{
    srand(time(NULL));
    for (int i = 1; i <= len; i++)
    {
        c[i] = rand() / double(RAND_MAX);
    }
}


int main()
{
    int a[Max + 1] = {0}, b[Max + 1] = {0};
    CreatRand(a, Max);
    for (int i = 0; i <= Max; i++)
    {
        b[i] = a[i];
    }
    cout<<"For the random series: "<<endl;
    for (int i = 1; i <= Max; i++)
    {
        cout<<b[i]<<" ";
    }
    cout<<endl;
    BubbleFind(b, Max, FIND);
    cout<<"The largest & the second largest: "<<endl;
    for (int i = Max; i > Max - FIND; i--)
    {
        cout<<b[i]<<" ";
    }
    cout<<endl;

    double c[Max + 1] = {0.0}, d[Max + 1] = {0.0};
    CreatRandDouble(c, Max);
    for (int i = 0; i <= Max; i++)
    {
        d[i] = c[i];
    }
```

```
    cout<<"For the double random series: "<<endl;
    for (int i = 1; i <= Max; i++)
    {
        cout<<d[i]<<" ";
    }
    cout<<endl;
    BubbleFind(d, Max, FIND);
    cout<<"The largest & the second largest: "<<endl;
    for (int i = Max; i > Max - FIND; i--)
    {
        cout<<d[i]<<" ";
    }
    cout<<endl;

    return 0;
}
```

4. 运行与测试

排序运行结果如图所示.

```
For the int random series:
92 56 48 52 28
After Bubble Sort, now the series:
28 48 52 56 92
For the float random series:
0.921384 0.111545 0.718619 0.444075 0.888913
After Bubble Sort, now the series:
0.111545 0.444075 0.718619 0.888913 0.921384
For the double random series:
0.921384 0.111545 0.718619 0.444075 0.888913
After Bubble Sort, now the series:
0.111545 0.444075 0.718619 0.888913 0.921384
```

选取int, float, double型数据进行测试.

选前两位运行结果如图所示.

```
For the int random series:
92 56 48 52 28
After Bubble Sort, now the series:
28 48 52 56 92
For the float random series:
0.921384 0.111545 0.718619 0.444075 0.888913
After Bubble Sort, now the series:
0.111545 0.444075 0.718619 0.888913 0.921384
For the double random series:
0.921384 0.111545 0.718619 0.444075 0.888913
After Bubble Sort, now the series:
0.111545 0.444075 0.718619 0.888913 0.921384
```

选取int, float, double型数据进行测试.

5. 总结与心得

本实验练习了随机数生成, 实现了用函数模板解决问题.

实验三: 循环左移

1. 实验目的

编写循环左移的函数, 编写函数并验证.

2. 实验内容

设计一个算法复杂度为$O(n)$的算法, 实现将数组$A[n]$中所有元素循环左移k个位置, 再用模板实现.

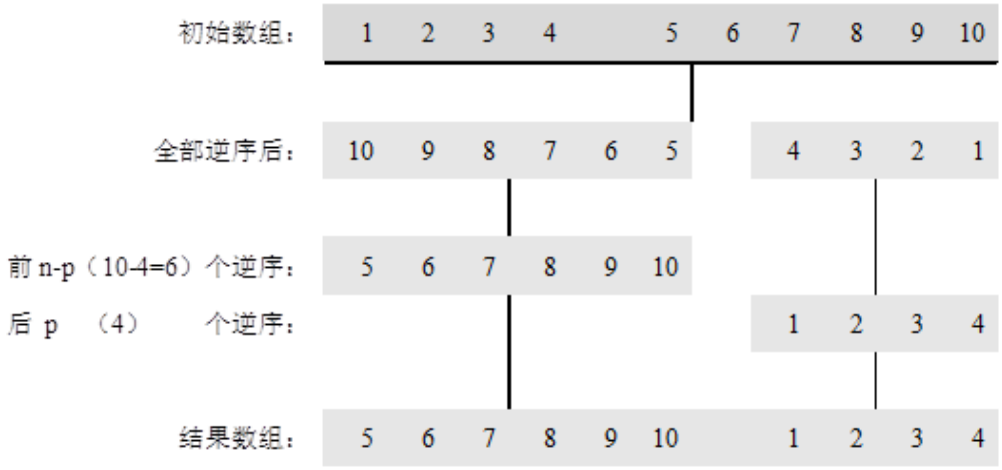
3. 设计与编码

1.本实验用到的理论知识

- 算法复杂度为$O(n)$的逆转算法.
- 逆置函数的编写

2. 算法设计

- 先用随机数生成器生成随机数组.
- 进行三次转换, 第一次换前面, 第二次换后面, 第三次换总体, 示意如图.



- 逆序算法是由前到后经过一半的数组长度交换数组内容

3. 编码

循环左移

```
#include <iostream>
#include <stdlib.h>
#include <time.h>
using namespace std;

const int Max = 7;
const int k = 2;

void CreatRand(int a[], int len)
{
    srand(time(NULL));
    for (int i = 0; i < len; i++)
    {
        a[i] = 1 + rand() % 100;
    }
```

```
}

void Reverse(int input[], int from, int to)
{
    int temp;
    for (int i = 0; i < (to - from) / 2 + 1; i++)  // 注意: 这个条件判断改为 +1
    {
        temp = input[from +i];
        input[from + i] = input[to - i];
        input[to - i] = temp;
    }
}

void Converse(int array[], int len, int k)
{
    Reverse(array, 0, k - 1);
    Reverse(array, k, len - 1);
    Reverse(array, 0, len - 1);
}

int main()
{
    int a[Max] = {};
    CreatRand(a, Max);
    cout<<"For the series:"<<endl;
    for (int i = 0; i < Max; i++)
    {
        cout<<a[i]<<" ";
    }
    Converse(a, Max, k);
    cout<<"Output:"<<endl;
    for (int i = 0; i < Max; i++)
    {
        cout<<a[i]<<" ";
    }
    return 0;
}
```

变为模板

```
#include <iostream>
#include <stdlib.h>
#include <time.h>
using namespace std;

const int Max = 7;
const int k = 3;

void CreatRand(int a[], int len)
{
    srand(time(NULL));
    for (int i = 0; i < len; i++)
    {
        a[i] = 1 + rand() % 100;
    }
}
void CreatRandDouble(double c[], int len)
```

```
{
    srand(time(NULL));
    for (int i = 0; i < len; i++)
    {
        c[i] = rand() / double(RAND_MAX);
    }
}

template <typename T>
void Reverse(T input[], int from, int to)
{
    T temp;
    for (int i = 0; i < (to - from) / 2 + 1; i++)  // 注意: 这个条件判断改为 +1
    {
        temp = input[from +i];
        input[from + i] = input[to - i];
        input[to - i] = temp;
    }
}
template <typename T>
void Converse(T array[], int len, int k)
{
    Reverse(array, 0, k - 1);
    Reverse(array, k, len - 1);
    Reverse(array, 0, len - 1);
}

int main()
{
    int a[Max] = {};
    CreatRand(a, Max);
    cout<<"For the series:"<<endl;
    for (int i = 0; i < Max; i++)
    {
        cout<<a[i]<<" ";
    }
    cout<<endl;
    Converse(a, Max, k);
    cout<<"Output:"<<endl;
    for (int i = 0; i < Max; i++)
    {
        cout<<a[i]<<" ";
    }
    cout<<endl;
    double b[Max] = {};
    CreatRandDouble(b, Max);
    cout<<"For the series:"<<endl;
    for (int i = 0; i < Max; i++)
    {
        cout<<b[i]<<" ";
    }
    cout<<endl;
    Converse(b, Max, k);
    cout<<"Output:"<<endl;
    for (int i = 0; i < Max; i++)
    {
        cout<<b[i]<<" ";
    }
```

```
    return 0;
}
```

4. 运行与测试

若取向左循环2位, 运行结果如图所示.

```
For the series:
63 46 57 75 9 7 80 Output:
57 75 9 7 80 63 46
```

用模板实现向左循环移动3位, 运行结果如图所示.

```
For the series:
25 16 4 96 16 1 16
Output:
96 16 1 16 25 16 4
For the series:
0.281503 0.558947 0.967528 0.39964 0.488754 0.488296 0.919065
Output:
0.39964 0.488754 0.488296 0.919065 0.281503 0.558947 0.967528
```

在这里我选用了int和double两种数据类型.

5. 总结与心得

本题实现了一个算法复杂度为$O(n)$的算法, 实现将数组$A[n]$中所有元素循环左移k个位置, 而且用模板实现.

注意到书上的Reverse()函数的判断条件有误, 应将判断条件改为

```
for (int i = 0; i < (to - from) / 2 + 1; i++)
```

实验四: 编写算法实现数组调整

1. 实验目的

编写算法实现数组调整, 使得将数组调整为左奇数,右偶数.

2. 实验内容

设计一个时间复杂度为$O(n)$的算法, 使得将数组调整为左奇数,右偶数.

3. 设计与编码

1.本实验用到的理论知识

- 循环体的判断与编写

2. 算法设计

- 先用随机数生成器生成随机数组.
- 从数组的两段向中间比较, 设置两个变量i和j初始时$i=0, j=n-1$, 若$A[i]$偶数, $A[j]$奇数则两者交换.

3. 编码

代码

```
#include <iostream>
#include <stdlib.h>
#include <time.h>
using namespace std;

const int LEN = 10;

void CreatRand(int a[], int len)
{
    srand(time(NULL));
    for (int i = 0; i < len; i++)
    {
        a[i] = 1 + rand() % 100;
    }
}
void Adjust(int a[], int len)
{
    int temp;
    int i = 0, j = len - 1;
    while (i < j)
    {
        while (a[i] % 2 != 0)
        {
            i++;
        }
        while (a[j] % 2 == 0)
        {
            j--;
        }
        if (i < j)
        {
            temp = a[j];
            a[j] = a[i];
            a[i] = temp;
        }
    }
}
int main()
{
    int a[LEN] = {};
    CreatRand(a, LEN);
    cout<<"For the random series: "<<endl;
    for (int i = 0; i < LEN; i++)
    {
        cout<<a[i]<<" ";
    }
    cout<<endl;
```

```
    Adjust(a, LEN);
    cout<<"After adjust: "<<endl;
    for (int i = 0; i < LEN; i++)
    {
        cout<<a[i]<<" ";
    }
    cout<<endl;
    return 0;
}
```

4. 运行与测试

运行结果如图所示.

```
For the random series:
50 63 13 13 58 39 8 81 26 13
After adjust:
13 63 13 13 81 39 8 58 26 50
```

生成了长度为10的数组, 左奇数, 有偶数.

5. 总结与心得

本题的关键在于写清楚判断条件和循环关系. 用while方便做逻辑判断.

实验五: 验证线性表实验

1. 实验目的

复习类模板, try-catch结构, 异常机制.

2. 实验内容

- 加入求线性表的长度等操作.
- 重新给定测试数据，验证抛出异常机制.

3. 设计与编码

1.本实验用到的理论知识

- 模板的编写
- C++异常机制

2. 算法设计

- 只需要已经给出的代码稍作修改.

3. 编码

头文件: SeqList.h

```
#ifndef SeqList_H
#define SeqList_H
const int MaxSize=10;
template <typename T>
```

```
class SeqList
{
public:
   SeqList();
   SeqList(T a[], int n);
   ~SeqList();
   void Insert(int i, T x);
   T Delete(int i);
   T Locate(T x);
   void PrintList();
   void len();
private:
   T data[MaxSize];
   int length;
};
#endif
```

实现的函数: SeqList.cpp

```
#include <iostream>
using namespace std;
#include "SeqList.h"

template <typename T>
SeqList<T>::SeqList()
{
  length = 0;
}
template <typename T>
SeqList<T>::SeqList(T a[], int n)
{
      if (n>MaxSize)
      {
        throw "ParameterError";
      }
      else
      {
        for (int i=0; i<n; i++)
        data[i]=a[i];
        length=n;
      }
}
template <typename T>
void SeqList<T>::Insert(int i, T x)
{
      if (length>=MaxSize) throw "OverFlow";
      if (i<1 || i>length+1) throw "SynaxError";
      for (int j=length; j>=i; j--)
        data[j]=data[j-1];   //
      data[i-1]=x;
      length++;
}

template <typename T>
SeqList<T>::~SeqList(){}
template <typename T>
T SeqList<T>::Delete(int i)
```

```
{
      if (length==0) throw "Underflow";
      if (i<1 || i>length) throw "SynaxError";
      int x=data[i-1];
      for (int j=i; j<length; j++)
        data[j-1]=data[j];
      length--;
      return x;
}
template <typename T>
T SeqList<T>::Locate(T x)
{
      for (int i=0; i<length; i++)
        if (data[i]==x)  return i+1 ;
      return 0;
}
template <typename T>
void SeqList<T>::PrintList()
{
  for (int i=0; i<length; i++)
    cout<<data[i]<<"  ";
  cout<<endl;
}
template <typename T>
void SeqList<T>::len()
{
  cout<<"Length: "<<length<<endl;
}
```

主函数: SeqList_main.cpp

```
#include <iostream>
using namespace std;
#include "SeqList.h"
#include "SeqList.cpp"
int main()
{
    int r[5]={1, 2, 3, 4, 5};
    SeqList<int>L(r, 5);
    L.PrintList();
    L.len();
    try
    {
        SeqList<int>M(r, 20);
    }
    catch(const char* s)
    {
        cout<<s<<endl;
    }
    cout<<"Before insert:"<<endl;
    L.PrintList();
    try
    {
        L.Insert(2,3);
    }
    catch (const char *s)
    {
```

```
        cout<<"InsertError: "<<s<<endl;
    }
    try
    {
        L.Insert(60,3);
    }
    catch (const char *s)
    {
        cout<<"InsertError: "<<s<<endl;
    }
    cout<<"After insert: "<<endl;
    L.PrintList();
    L.len();
    cout<<"value is 3, location: ";
    cout<<L.Locate(3)<<endl;

    cout<<"delete the first element, before delete:"<<endl;
    L.PrintList();
    L.len();
    try
    {
        L.Delete(1);
    }
    catch (const char *s)
    {
        cout<<"DeleteError: "<<s<<endl;
    }
    cout<<"After delete: "<<endl;
    L.PrintList();
    L.len();
    try
    {
        L.Delete(9);
    }
    catch (const char *s)
    {
        cout<<"DeleteError: "<<s<<endl;
    }
    return 0;
}
```

4. 运行与测试

运行结果如图所示.

```
1  2  3  4  5
Length: 5
ParameterError
Before insert:
1  2  3  4  5
InsertError: SynaxError
After insert:
1  3  2  3  4  5
Length: 6
value is 3, location: 2
delete the first element, before delete:
1  3  2  3  4  5
Length: 6
After delete:
3  2  3  4  5
Length: 5
DeleteError: SynaxError

--------------------------------
Process exited after 0.627 seconds with return value 0
请按任意键继续. . .
```

5. 总结与心得

通过本实验复习了C++的异常机制, 编写了模板类, 在main函数中实验了异常的情况.

实验六: 完成顺序表的逆序

1. 实验目的

复习验证类和模板类.

2. 实验内容

- 完成顺序表的逆序
- 使用类和模板类

3. 设计与编码

1.本实验用到的理论知识

- 验证类和模板类

2. 算法设计

- 只需要把前面题的SeqList改为模板类.

3. 编码

头文件: SeqList.h

```
#ifndef SeqList_H
#define SeqList_H
const int MaxSize=10;
template <typename T>
class SeqList
{
public:
   SeqList();
```

```
   SeqList(T a[], int n);
   ~SeqList();
   void Insert(int i, T x);
   T Delete(int i);
   T Locate(T x);
   void PrintList();
   void Reverse();
private:
   T data[MaxSize];
   int length;
};
#endif
```

实现的函数: SeqList.cpp

```
#include <iostream>
using namespace std;
#include "SeqList.h"

template <typename T>
SeqList<T>::SeqList()
{
  length = 0;
}
template <typename T>
SeqList<T>::SeqList(T a[], int n)
{
      if (n>MaxSize) throw "ParameterError";
      for (int i=0; i<n; i++)
        data[i]=a[i];
      length=n;
}
template <typename T>
void SeqList<T>::Insert(int i, T x)
{
      if (length>=MaxSize) throw "OverFlow";
      if (i<1 || i>length+1) throw "SynaxError";
      for (int j=length; j>=i; j--)
        data[j]=data[j-1];   //
      data[i-1]=x;
      length++;
}

template <typename T>
SeqList<T>::~SeqList(){}
template <typename T>
T SeqList<T>::Delete(int i)
{
      if (length==0) throw "Underflow";
      if (i<1 || i>length) throw "SynaxError";
      int x=data[i-1];
      for (int j=i; j<length; j++)
        data[j-1]=data[j];
      length--;
      return x;
}
template <typename T>
```

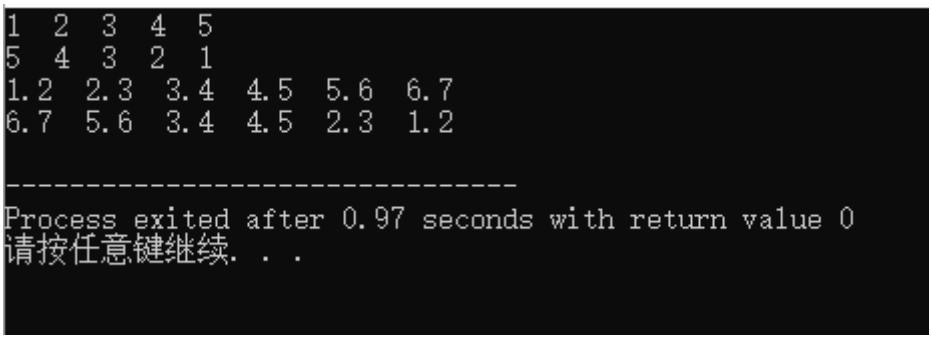
```
T SeqList<T>::Locate(T x)
{
      for (int i=0; i<length; i++)
        if (data[i]==x)  return i+1 ;
      return 0;
}
template <typename T>
void SeqList<T>::PrintList()
{
  for (int i=0; i<length; i++)
    cout<<data[i]<<"  ";
  cout<<endl;
}

template <typename T>
void SeqList<T>::Reverse()
{
  T temp;
  for (int i = 0; i <= length / 2; i++)
  {
    temp = data[i];
    data[i] = data[length - i - 1];
    data[length - i - 1] = temp;
  }

}
```

主函数: SeqList_main.cpp

```
#include <iostream>
using namespace std;
#include "SeqList.h"
#include "SeqList.cpp"
int main()
{
    int r[5]={1, 2, 3, 4, 5};
    SeqList<int>L(r, 5);
    L.PrintList();
    L.Reverse();
    L.PrintList();
    double s[6] = {1.2, 2.3, 3.4, 4.5, 5.6, 6.7};
    SeqList<double>W(s, 6);
    W.PrintList();
    W.Reverse();
    W.PrintList();
    return 0;
}
```

4. 运行与测试

运行结果如图所示.



第一次为长度为5的整型数组, 第二次为长度为6的double型数组.

5. 总结与心得

在用模板类写函数的时候要加上

```
template <typename T>
```

在声明对象的时候要加上

```
SeqList<类型> 对象
```