燕山大學
YANSHAN UNIVERSITY

C++面向对象程序设计

实验指导书

（8）函数模板、类模板与 STL 库

燕山大学软件工程系

目　录

实验 08 函数模板、类模板与 STL 库

1.1 时间安排 2 学时

本实验安排 2 个实验课时。

1.2 实验目的和要求

1．理解使用函数模板、类模板的作用及意义。

2．掌握函数模板、类模板的声明及使用方法。

3．理解 C++STL 库中容器、算法、迭代器、函数对象的作用及功能，并熟悉使用 STL 库的方法。

1.3 实验内容 I　(调试、理解、体会、掌握)

（1）定义两个数交换的函数模板 Swap 和能够进行升序、降序排序的函数模板 BubbleSort；从而实现对数组元素的排序及输出。

程序源码：

```
#include   <iostream>
using namespace std;
 // 函数模板：交换x 和y 的值
template   <class T>
void Swap   (T &x, T &y)
{
      T temp;
      temp   = x;
      x   = y;
      y   = temp;
}
// 函数模板：用起泡法对数组A 的n 个元素进行排序
template   <class T>
void BubbleSort(T A[], int n, bool SortFlag)   //SortFlag:true为升序，false为降序
{
      int i,j;
      int lastExchangeIndex; //用于记录每趟被交换的最后一对元素中较小或较大的下标
      i   = n-1;   // i 是下一趟需参与排序交换的元素之最大或最小下标
      while   (i   >   0) //持续排序过程，直到最后一趟排序没有交换发生，或已达n-1趟
      {
            lastExchangeIndex   =   0;
            //每一趟开始时，设置交换标志为0（未交换）
            for   (j = 0; j < i; j++)   //每一趟对元素A[0]..A[i]进行比较和交换
            {
```

```
            if    (!SortFlag)   //降序
            {
                  if   (A[j+1]>A[j])   //如果元素A[j+1]   > A[j]，交换之，即大数向前移
                   {
                        Swap(A[j],A[j+1]);
                        lastExchangeIndex    = j; //记录被交换的一对元素中较小的下标
                   }
            }
            else   //升序
            {
                   if   (A[j+1]<A[j])   //如果元素A[j+1]   < A[j]，交换之，即大数向后移
                  {
                        Swap(A[j],A[j+1]);
                        lastExchangeIndex    = j; //记录被交换的一对元素中较大的下标
                  }

            }
        }
        // 若发生交换，则将i 设置为本趟被交换的最后一对元素中较小或较大的下标
        //若不发生交换，则lastExchangeIndex为，i随即也变为，此时将退出while循环。
        i    = lastExchangeIndex;
        //输出本次交换完的数据序列
        for(int k=0;k<n;k++) cout    << A[k]    << " ";
                cout    << endl;
    }
}
void main()
{    int i;
     int data1[]={1,3,5,7,9,11,13,15,17,19,2,4,6,8,10,12,14,16,18,20};
     int count=sizeof(data1)/sizeof(int);   //获取数组元素的个数
     cout    << "排序前的数据："    << endl;
     for(i=0;i<count;i++)
          cout    << data1[i]    << " ";
     cout    << endl;
     cout    << "开始排序..."    << endl;
     BubbleSort(data1, count,false);     //降序排序
     cout    << "排序后的数据："    << endl;
     for(i=0;i<count;i++)
          cout    << data1[i]    << " ";
     cout    << endl;
}
```

（2）阅读下列程序，体会类模板的声明与使用。

```
#include <iostream>
#include <string>
using namespace std;
//模板的声明和定义只能在全局、命名空间或者类范围内进行。
template<class T1,typename T2>     //注意class、typename的异同
class A
{
public:
    void f(T1 a, T2 b);
};
//类模板函数成员的声明
template<class T1,class T2>
void A<T1,T2>::f(T1 a,T2 b)
{
    cout << "class A--->T1:" << a <<";T2:" << b << endl;
}
//定义类模板的默认类型形参，默认类型形参不适用于函数模板。
template<typename T3, typename T4=int>//T4是默认模板类型形参
class B
{
private:
    T3 t3;
    T4 t4;
public:
    B(T3 a, T4 b);
    void show();
};
template<typename T3,typename T4>
B<T3,T4>::B(T3 a, T4 b):t3(a),t4(b){}
/*template<class T3,class T4=int> B<T3,T4>::B(T3 a, T4 b):t3(a),t4(b){},这样是错误的，
在类模板外部定义带有默认类型的形参时，在template的形参表中默认值应该省略*/
template<class T3,class T4>
void B<T3,T4>::show()
{
    cout << "class B--->T3:" << t3 <<";T4:" << t4 << endl;
}
/*非类型模板参数: 非类型形参只能是整型、指针和引用，像double,string,string **这样的
类型是不允许的，但是double &,double *对象的引用或指针是正确的。*/
template<class T5,int a>
class C
{
private:
    T5 max[a];
```

```
public:
    void cshow()
    {
        cout << "class C--->T5:" << typeid(T5).name()<< endl;   //typeid获取变量类型的函数
    }
};
int main( )
{
 /*类模板--将参数绑定到形式参数(int-->T1、int-->T2)--类模板的实例化A<int,int>-->实例化对象a1   */
    A<int,int> a1;
    a1.f(2,3);

    A<int,char> a2;
    a2.f(2,'a');

    A<string,int> a3;
    a3.f("hello word!",5);

    //带有默认类型形参的模板类
    B<char,char> b1('a','b');
    b1.show();

    B<string,string> b2("你好","测试中.....");
    b2.show();

    B<int,char> b3(25,'F');
    b3.show();

   //非类型模板参数
    const int i = 5;
    C<int,i> c1;
    c1.cshow();

    //int j = 5;
    //C<int,j> c2; //错误，调用非类型模板形参的实参必须是常量表达式

    C<char,i> c2;
    c2.cshow();
    return 0;
}
```

（3）在标准 C++类库中 ，双向队列类（deque）的成员函数 queue::insert()往一个双向队列中插入元素， queue::push_front(const T& x)往一个双向队列的头端插入一个元素，queue::pop_front()从一个双向队列的头端删除一个元素，queue::push_back(const T& x) 往一个双向队列的尾端插入一个元素，queue::pop_back(const T& x)从一个双向队列的尾端删除一个元素。利用算法 transform、函数对象 operation()将容器中的大写字母变为小写字母。请构造一个字符型双向队列，体会 STL 模板中成员函数、算法、迭代器、函数对象的用法。

程序源码：

```
#include   <iostream>
#include   <deque>
#include <algorithm>
using namespace std;
//通过类模板deque实例化一个队列CHARDEQUE，类型是char
typedef deque<char> CHARDEQUE;
void print_contents   (CHARDEQUE deque); //函数原型声明
char operation (char & ch) ;//函数原型声明
void main()
{
      /************初始化****************/
      cout<<"队列初始化"<<endl;
      CHARDEQUE a(3,'A');        //create a with   3 A's       实例化一个对象a
      CHARDEQUE b(2,'?');   //create b with   2 B's.
      print_contents   (a); //print out the contents
      print_contents   (b);
      /************插入操作****************/
      cout<<"插入元素"<<endl;
      a.insert(a.begin(),'X');     //insert 'X' to the beginning of a
      print_contents   (a);
      a.insert(a.end(),'Y');       //insert 'Y' to the end of a
      print_contents   (a);
      a.insert(a.end()-1,3,'Z'); //inset   3 'Z's to one item before the end of a
      print_contents   (a);
      a.insert(a.end(),b.begin(),b.end()); //insert to the end of a from b
      print_contents   (a);
   /************入队*****************/
      cout<<"入队"<<endl;
     a.push_back('m');
      print_contents(a);
     a.push_front('?');
      print_contents(a);
   /************出队*****************/
      cout<<"出队"<<endl;
      a.pop_back();
      print_contents(a);
      a.pop_front ();
      print_contents(a);
    /**容器*算法*迭代器*函数对象：将容器中的大写字母变为小写字母*****/
      transform (a.begin(), a.end(), a.begin(),   operation);
      print_contents(a);
}
//将大写字母转换为小写字母
char operation (char & ch)
```

```
{
    ch=(ch>='A'&&ch<='Z')?(ch+32):ch;
    return ch;
}

void print_contents    (CHARDEQUE deque) //function to print the contents of deque
{
       CHARDEQUE::iterator pdeque;    //迭代器
       cout    <<"The output is: ";
       for(pdeque    = deque.begin(); pdeque    != deque.end(); pdeque++)
        {
            cout    <<    *pdeque    <<" "    ;
         }
        cout<<endl;
}
```

1.4 实验内容 II (自主完成)

注：将题目的构思过程、源码、运行结果（截图）、心得体会等内容按要求填写，详见实验报告模板。

功能要求：声明一个单向链表类模板，使之支持节点的升序插入、查找、删除等操作。

实现要求：用函数模板、类模板实现，以支持多种数据类型。

分析：

节点类模板 **NodeCls**：

数据成员包括:数据 **Data**、指向下一节点的指针 **next**

函数成员包括：构造函数

获取下一节点地址的函数 **NextNode**

获取当前节点数据的函数 **GetData**

单向链表类模板 **SinglyLinkedlist:**

数据成员：表头指针 **head**,表尾指针 **rear**, 当前位置指针 **curpos**,

上一结点位置指针 **fontpos**, 节点个数 **NCount**

函数成员包括：产生新节点函数 **NewNode**

升序插入节点函数 **AscendingInsert**

重置位置指针函数 **Reset**

按值查找函数 **Find**

输出链表节点函数 **Show**

在具体实现时，可自主决定节点类、链表类的具体数据成员、函数成员及其功能；

上述分析仅供参考。