**实验13 12.24-oj**

1. 实验的目的和要求

学习c++编程的原理和方法

1. 实验内容
2. 实验准备

复习函数模板的使用

C++中类模板的使用

C++中异常处理，异常抛出，异常捕获和处理

1. 实验项目

A题目描述

实现一个二分查找函数模板 find, 输入一个数组、数组长度和需要找的元素，返回元素下标（未找到返回-1）

分析

复习函数模板，注意将需要变化的参数类型处理为class T,并且注意在调用函数的过程中那些地方要使用T，同时还用到了函数重载以便于递归使用二分查找函数

源代码

#include<iostream>

using namespace std;

template<class T>

int find(T Array[10],int l,int r,T number)

{

if((l==r)&&(Array[l]!=number))return (-1);

int mid=(l+r)/2;

if(Array[mid]==number)return(mid);

else if(Array[mid]<number)return(find(Array,mid+1,r,number));

else return(find(Array,l,mid,number));

}

template<class T>

int find(T Array[10],int n,T number)

{

return(find(Array,0,9,number));

}

测试数据

int main(){

int a[10] = {0,1,2,3,4,5,6,7,8,9};

char b[10] = {'a','b','c','d','e','f','g','h','i','j'};

double c[10] = {1.1,1.2,1.3,1.4,1.5,1.6,1.7,1.8,1.9,2};

cout<<find(a, 10, 10)<<endl;

cout<<find(a, 10, 0)<<endl;

cout<<find(b, 10, 'a')<<endl;

cout<<find(b, 10, 'k')<<endl;

cout<<find(c, 10, 1.0)<<endl;

cout<<find(c, 10, 1.1)<<endl;

return 0;

}

Output

-1   
0   
0   
-1   
-1   
0

B题目描述

实现一个有序表的类模板（始终维持表内数据按从大到小排列），提供功能有插入元素、输入第N大元素、输出所有元素。   
template <typename T>   
class SortList{   
    void print(); //输出所有元素   
     T Kth(int k); //输出第k大元素   
    void insert(T a); //插入元素a   
};

 分析

这里主要考察的是类模板，注意两点一个是类模板的结构是什么，其次在类模板函数的实现的时候的写法是 函数类型 Sortlist<T> 函数名 （参数表）{}，同时注意到类模板中的构造函数和析构函数，为了操作方便在私有成员中添加一个排序成员函数。

源程序

#include<iostream>

using namespace std;

template <typename T>

class SortList{

private:

int p;

T\*Array;

void qsort(T Array[],int l,int r);

public:

SortList(){Array=new T [100000];p=0;}

void print(); //输出所有元素

T Kth(int k); //输出第k大元素

void insert(T a); //插入元素a

~SortList(){if (Array)delete[]Array;}

} ;

template<typename T>

void SortList<T>::qsort(T Array[],int l,int r)

{

int i=l,j=r;

T mid=Array[(l+r)/2];

do{

while(Array[i]>mid)i++;

while(Array[j]<mid)j--;

if(i<=j)

{

T temp;

temp=Array[i];

Array[i]=Array[j];

Array[j]=temp;

i++;j--;

}

}while (i<=j);

if(l<j)qsort(Array,l,j);

if(i<r)qsort(Array,i,r);

}

template<typename T>

void SortList<T>::print()

{

for(int i=0;i<p;i++)

cout<<Array[i]<<' ';

cout<<endl;

}

template<typename T>

T SortList<T>::Kth(int k)

{

return(Array[k-1]);

}

template<typename T>

void SortList<T>::insert(T a)

{

Array[p]=a;

qsort(Array,0,p);

p++;

}

测试数据

int main(){

SortList<int> int\_list;

SortList<char> char\_list;

for(int i=0;i<5;i++){

int\_list.insert(i);

char\_list.insert('a'+i);

}

int\_list.print();

char\_list.print();

cout<<int\_list.Kth(3)<<endl;

cout<<char\_list.Kth(4)<<endl;

return 0;

}

Output

4 3 2 1 0   
e d c b a   
2   
b

C

题目描述

设计一个计算n!的函数 int factorial(int n), 当n是负数或结果超过int范围时抛出异常。

int main() {

int x;

while (cin >> x) {

try {

cout << factorial(x) << endl;

}

catch (const char \*err) {

cout << err << endl;

}

}

return 0;

}

分析

一道常见的异常处理，对于函数先判断是否大于零，如果小于零直接抛出，然后开始计算阶乘，由于在C++中当数据超过Int的范围并不会报错而是将其减去maxint,因而在处理该异常的时候，判断如果result\*i<result那就抛出。

对于捕获的过程则不需任何处理，题目已经给出。

源代码

#include<iostream>

using namespace std;

int factorial(int x)

{

int result=1;

if(x<0)throw("n should be greater than zero");

for(int i=1;i<=x;i++)

{

if(result\*i<result)throw("n is too large");

result\*=i;

}

return result;

}

测试数据

Input

-5

-2

10

100

Output

n should be greater than zero

n should be greater than zero

3628800

n is too large

三小结

这次主要着眼于面向对象的过程中的一些小的地方，首先复习了之前内容函数模板，然后进一步实现了类模板，类模板的应用很好的过充了类的功能，便利程序员和使用的人，当然可以再课后加入类模板与重载函数的使用，以及继承的结合。最后一部分是关于异常的处理，简单的了解了如何正确的使用异常抛出，异常的捕获和处理，并且理清程序运行的结构。