# 第17次实验

# C++的异常处理

**1、实验学时：**2个课时。

**2、实验类型：**设计性实验。

**3、实验目的：**

（1）了解C++异常的种类，理解异常继承体系和异常处理机制的思想；

（2）了解C++的Exception类、Try和Catch机制、程序异常的捕获方式；

（3）掌握C++中异常的定义方式和在程序中加入异常处理代码的方法。

**4、实验内容：**

**实验题目1：结合C++的异常处理，求解三角形的面积**

（1）程序功能简介

给出三角形的三边a、b、c，求三角形的面积。只有a+b>c，b+c>a，c+a>b时才能构成三角形。设置异常处理，对不符合三角形条件的输出警告信息，不予计算。

（2）程序设计说明

① 求解三角形面积的函数可参考如下代码：

double triangle(double a,double b,double c)

{

double area;

if(a+b>c&&a+c>b&&b+c>a)

{

double s=(a+b+c)/2;

area=sqrt(s\*(s-a)\*(s-b)\*(s-c));

}

else

throw a;

return area;

}

② 测试函数可参考如下代码：

void main()

{

double triangle(double,double,double);

double a,b,c;

cout<<"请输入三角形的三边："<<endl;

cin>>a>>b>>c;

try

{

//判断三角形的三边是否均大于0；如是，则求解面积；

}

catch(double)

{ cout<<"输入的三边有误！"<<endl; }

}

③ 在测试函数中，添加相应的异常处理代码，形成更为健壮的程序。

（3）程序调试运行

运行程序查看结果，并进行源代码调试和优化。

**参考代码如下：**

#include <iostream>

#include <string>

using namespace std;

double triangle(double a,double b,double c)

{

double area;

if(a+b>c&&a+c>b&&b+c>a)

{

double s=(a+b+c)/2;

area=sqrt(s\*(s-a)\*(s-b)\*(s-c));

}

else

throw "It’s not a triangle!";

return area;

}

void main()

{

double a,b,c;

cout<<"请输入三角形的三边："<<endl;

cin>>a>>b>>c;

try

{

cout<<"三角形面积为："<<triangle(a,b,c)<<endl;

//判断三角形的三边是否均大于0；如是，则求解面积；

}

catch(string)

{ cout<<"输入的三边有误！"<<endl; }

}

**执行结果如下：**

略。

**实验题目2：结合C++的异常处理，求解一个数的平方根**

（1）程序功能简介

编写一个程序，求输入数的平方根。设置异常处理，对输入负数的情况给出提示。

（2）程序设计说明

① 求解一个数的平方根的函数可参考如下代码：

double squareroot(double &a)

{

double n;

if(a>=0)

n=sqrt(a);

else

throw a;

return n;

}

② 测试函数可参考如下代码：

void main()

{

double m;

char h='Y';

try

{

//判断输入的数是否为负数；如是，则抛出异常；

}

catch(double)

{

cout<<"输入的数为负数，无法开方！"<<endl;

}

system("pause");

}

③ 在测试函数中，添加相应的异常处理代码，形成更为健壮的程序。

（3）程序调试运行

运行程序查看结果，并进行源代码调试和优化。

**参考代码如下：**

略。

**执行结果如下：**

略。

**实验题目3：结合C++的异常处理，求解用户输入的n!的值**

（1）程序功能简介

求n!的函数，当用户的输入太大（例如50）时，会出现错误。编写一个程序，使用异常处理机制来解决这一问题。

（2）程序设计说明

① 求解n!的函数可参考如下代码：

int fac(int n)

{

int result=1;

if(n<0)

throw "Argument for factorial negative";

else if(n>16)

throw n;

while(n)

{

result\*=n;

n--;

}

return result;

}

② 测试函数可参考如下代码：

void main()

{

int n;

char h='Y';

while(h=='Y')

{

cout<<"Please input a number n to calculte n!:";

cin>>n;

cout<<n<<"!="<<fac(n)<<endl;

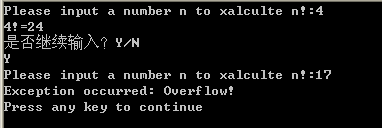
cout<<"是否继续输入？Y/N"<<endl;

cin>>h;

}

}

③ 在测试函数中，添加相应的异常处理代码，使得程序运行结果如下图所示：



（3）程序调试运行

运行程序查看结果，并进行源代码调试和优化。

**参考代码如下：**

略。

**执行结果如下：**

略。

**实验题目4：结合C++的异常处理，设计能够判断栈空和栈溢出的栈类**

（1）程序功能简介

设计一个关于栈的类，并且添加异常处理，判断栈空和栈溢出，在主程序中对其进行测试。

（2）程序设计说明

① 栈类的定义可参考如下代码：

class Stack

{

private:

int stack[SIZE];

int tos;

public:

Stack(){ tos=-1; }

~Stack(){}

void push(int a)

{

if(tos>=SIZE-1)

throw Full(a);

stack[++tos]=a;

}

int pop()

{

if(tos<0)

throw Empty();

return stack[tos--];

}

int top()const { return tos; }

};

② 判断栈空的类定义可参考如下代码：

class Empty

{

public:

void show(){ cout<<"stack is empty!"<<endl; }

};

③ 栈溢出的类定义可参考如下代码：

class Full

{

private:

int a;

public:

Full(int i):a(i){}

int getValue(){ return a; }

};

④ 测试函数可参考如下代码：

void main()

{

Stack s;

int b[200],i=0;

ifstream infile("data.dat",ios::in);

if(!infile)

{

cerr<<"open data.dat error!"<<endl;

exit(1);

}

}

⑤ 在测试函数中，添加相应的异常处理代码，形成更为健壮的程序。

（3）程序调试运行

运行程序查看结果，并进行源代码调试和优化。

**参考代码如下：**

略。

**执行结果如下：**

略。

**5、实验结果：**

参考实验报告模板，撰写本次实验的实验报告，并提交实验报告和实验题目的源代码。