C++工具

异常处理

异常处理的任务

> 程序中有三类错误:

1. 语法错误
2. 运行错误
3. 逻辑错误

异常处理的方法

> 对于异常,C++采取的方法是:将信息传给它的上一级

> C++处理异常的机制有三个部分

1. 检查try
2. 抛出throw
3. 捕捉catch

以下为一个例子

```
#include <iostream>
#include <stdexcept> // 包含标准异常类

// 检查三边长是否能构成三角形
bool canFormTriangle(int a, int b, int c) {
    if (a + b <= c || a + c <= b || b + c <= a) {
        return false; // 不满足三角形不等式定理
    }
    return true; // 满足三角形不等式定理
}

int main() {
    int a, b, c;
    std::cout << "Enter the lengths of the three sides: ";
    std::cin >> a >> b >> c;

    try {
        if (!canFormTriangle(a, b, c)) {
            throw std::invalid_argument("The given sides do not form a
triangle.");
        }
        std::cout << "The given sides can form a triangle." << std::endl;
    } catch (const std::invalid_argument& e) { // 捕获无效参数异常
        std::cerr << "Error: " << e.what() << std::endl;
    } catch (...) { // 捕获所有其他类型的异常
        std::cerr << "An unexpected error occurred." << std::endl;
    }

    return 0;
```

```
}
```

从上面,我们可以得到程序分析是如何进行异常处理的

1. 首先把可能出现异常的语句放在try后的花括号中。
2. 程序开始运行后，按正常的顺序执行到try块。如果在执行try块内的语句过程中没有发生异常，则catch子句不起作用，流程转到catch子句后面的语句继续执行。
3. 如果在执行try块内的语句(包括其所调用的函数)过程中发生异常，则trow运算符抛出一个异常double类型的信息a。throw抛出异常信息后，流程立即离开所在函数，转到其上一级的函数(main函数)，因此不会执行triangle函数中if语句之后的return语句。
4. 这个异常信息提供给try-catch结构，系统会寻找与之匹配的catch子句。这时执行catch(double)子句中的语句
5. 在进行异常处理后，程序并不会自动终止，继续执行catch子句后面的语句。

> 注意

- catch后面的圆括号中，一般只写异常信息的类型名
- catch 只检查所捕获异常信息的类型，而不检查它们的值
- 异常信息可以是C++系统预定义的标准类型，也可以是用户自定义的类型(如结构体或类)

在函数声明中进行异常情况指定

> throw(int,double,float,char)可指定抛出异常的类型

若不抛出异常,可改为throw()

在异常处理中处理析构函数

C++的异常处理机制会在throw
抛出异常信息被catch捕获时，对有关的局部对象进行析构(调用类的析构函数)，析构对象的顺序与构造的顺序相反，然后执行与异常信息匹配的catch块中的语句。

例子如下:

```
#include <iostream>
#include <string>

using namespace std;

class Student {
public:
    Student(int n, string nam) {
        cout << "constructor-" << n << endl;
        num = n;
        name = nam;
    }
```

```
    ~Student() {
        cout << "destructor-" << num << endl;
    }

    void get_data() {
        if (num == 0) {
            throw num; // 抛出整数0，这在C++中是允许的
        } else {
            cout << num << " " << name << endl;
        }
        cout << "in get_data()" << endl;
    }

private:
    int num;
    string name;
};

void fun() {
    try {
        Student stud1(1101, "Tan");
        stud1.get_data();
        Student stud2(0, "Li"); // 这将导致异常
        stud2.get_data();
    } catch (int e) { // 捕获整数异常
        cout << "Caught an exception: " << e << endl;
    }
}

int main() {
    fun();
    return 0;
}
```

命名空间

为什么需要命名空间

> 作用域是定义的变量可以被应用的有效区域

> 命名空间的作用是用来处理$\blue{同名冲突}$

> C语言定义了三个层次的作用域

1. 文件
2. 函数
3. 复合语句

关于命名空间例子如下:

```
#include <iostream>
using namespace std;

namespace A {
    int a = 10;
```

```
}

namespace B {
    int a = 20;
}

int main() {
    cout << A::a << endl;
    cout << B::a << endl;
    return 0;
}
```

要注意运算符::的用法，::是作用域运算符，用于访问命名空间中的变量。

使用早期的函数库

C++新方法：

```
#include <cstdio>
#include <cmath>
#include <cstring>
using namespace std;

//C语言方法：
```cpp
#include <stdio.h>
#include <math.h>
#include <string.h>
```
```