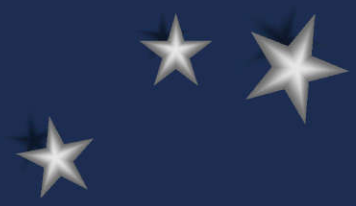
C++语言程序设计

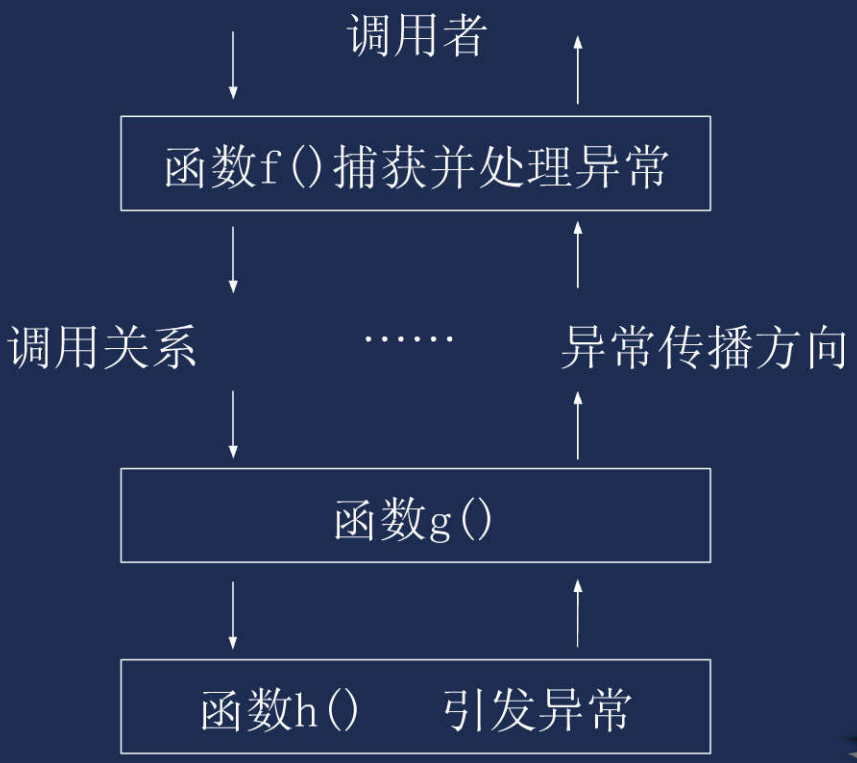
第十二章 异常处理

本章主要内容

- 异常处理的基本思想
- C++异常处理的实现
- 异常处理中的构造与析构



异常处理的基本思想



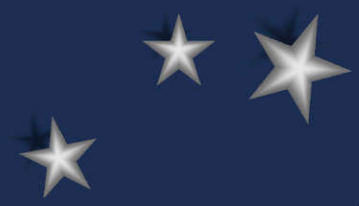
异常处理的实现机制

- 抛掷异常的程序段

```
.....
throw      表达式;
.....
```

- 捕获并处理异常的程序段

```
try
    复合语句
catch（异常类型声明）
    复合语句
catch（异常类型声明）
    复合语句
    ...
```



异常处理的实现机制（续）

- 若有异常则通过throw操作创建一个异常对象并抛掷。
- 将可能抛出异常的程序段嵌在try块之中。控制通过正常的顺序执行到达try语句，然后执行try块内的保护段。
- 如果在保护段执行期间没有引起异常，那么跟在try块后的catch子句就不执行。程序从try块后跟随的最后一个catch子句后面的语句继续执行下去。
- catch子句按其在try块后出现的顺序被检查。匹配的catch子句将捕获并处理异常（或继续抛掷异常）。
- 如果匹配的处理器未找到，则运行函数terminate将被自动调用，其缺省功能是调用abort终止程序。

例12-1处理除零异常

```
#include<iostream.h>
int Div(int x,int y);
int main()
{  try
   { cout<<"5/2="<<Div(5,2)<<endl;
     cout<<"8/0="<<Div(8,0)<<endl;
     cout<<"7/1="<<Div(7,1)<<endl;
   }
   catch(int)
   { cout<<"except of deviding zero.\n"; }
   cout<<"that is ok.\n";
}
int Div(int x,int y)
{  if(y==0) throw y;
   return x/y;
}
```

程序运行结果如下：

```
5/2=2
except of deviding zero.
that is ok.
```

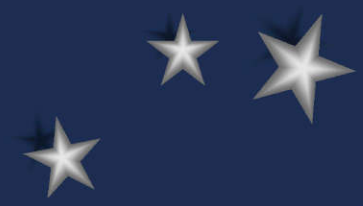
异常接口声明

- 可以在函数的声明中列出这个函数可能抛掷的所有异常类型。
  - 例如：

```
void fun() throw(A, B, C, D);
```

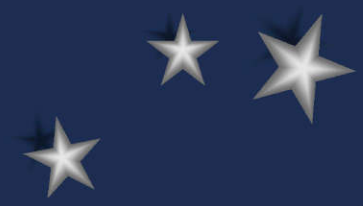
- 若无异常接口声明，则此函数可以抛掷任何类型的异常。
- 不抛掷任何类型异常的函数声明如下：

```
void fun() throw();
```



异常处理中的构造与析构

- 找到一个匹配的catch异常处理后
  - 初始化参数。
  - 将从对应的try块开始到异常被抛掷处之间构造（且尚未析构）的所有自动对象进行析构。
  - 从最后一个catch处理之后开始恢复执行。



例12-2 异常处理时的析构

```
#include <iostream.h>
void MyFunc( void );
class Expt
{ public:
    Expt(){};
    ~Expt(){};
    const char *ShowReason() const
    {  return "Expt类异常。";
    }
};
```

```
class Demo
{ public:
    Demo();
    ~Demo();
};
Demo::Demo()
{
  cout<<"构造 Demo."<<endl;
}
Demo::~Demo()
{
  cout<<"析构 Demo."<<endl;
}
```

```
void MyFunc()
{ Demo D;
  cout<<"在MyFunc()中抛掷Expt类异常。"<<endl;
  throw Expt();
}
int main()
{ cout<<"在main函数中。"<<endl;
  try
  { cout<<"在try块中，调用MyFunc()。" <<endl;
    MyFunc();
  }
```

```
    catch( Expt E )
    { cout<<"在catch异常处理程序中。"<<endl;
      cout<<"捕获到Expt类型异常："；
      cout<<E.ShowReason()<<endl;
    }
    catch( char *str )
    { cout<<"捕获到其他的异常："<<str<<endl;}
    cout<<"回到main函数。从这里恢复执行。"
        <<endl;
    return 0;
}
```

程序运行时输出：

在main函数中。
在try块中，调用MyFunc()。
构造 Demo.
在MyFunc()中抛掷Expt类异常。
析构 Demo.
在catch异常处理程序中。
捕获到Expt类型异常：Expt类异常。
回到main函数。从这里恢复执行。

小结与复习建议

- 主要内容
  - 异常处理的基本思想、C++异常处理的实现、异常处理中的构造与析构
- 达到的目标
  - 简单了解C++的异常处理机制
- 实验任务
  - 实验十二