1 异常的基本概念

异常: 出错后,将出错问题返回给调用处

c语言的异常处理比较简单,容易出错,c++处理异常不容易出错

```
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <string.h>
#include <string>
using namespace std;
int mydive(int a, int b)
{
 if (b == 0)
 return -1; //errno = 2

  return a / b;
}

void test01()
```

```
{
 int ret = mydive(1, -1);
 if (ret == -1)
 {
 cout << "除数为0" << endl;//perror("");

 }
}

int main() {

 test01();

}
```

2 c++处理异常

2.1 c++异常基本语法

```
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <string.h>
#include <string>
using namespace std;
int mydive(int a, int b)
{
 if (b == 0)
 throw 'a';//处理异常 抛出异常 抛出一个类型

 return a / b;
}
void test01()
{
 //尝试捕获异常
 try
 {
 mydive(2,0);
 }
 catch (char)//如果没有捕获的抛出的异常 程序会被终止
 {
 //cout << "捕获了一个char类型的异常" << endl;
```

```
 throw 'a';

 }

}
int main()
{
 try {
 test01();
 }
 catch (char)
 {
 cout << "捕获了一个char类型的异常" << endl;
 }
}
```

2.2异常严格类型匹配

```
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <string.h>
#include <string>
using namespace std;
int mydive(int a, int b)
{
 string str = "hello";
 if (b == 0)
 throw str;//处理异常 抛出异常 抛出一个类型

 return a / b;
}
void test01()
{
 //尝试捕获异常
 try
 {
 mydive(2,0);
 }
 catch (char)//如果没有捕获的抛出的异常 程序会被终止
 {
 //cout << "捕获了一个char类型的异常" << endl;
 throw 'a';
```

```
25
26   }
27   catch (int)
28   {
29   cout << "捕获了一个int类型的异常" << endl;
30   }
31   catch (double)
32   {
33   cout << "捕获了一个double类型的异常" << endl;
34   }
35   catch (...)
36   {
37   cout << "捕获了一个其他类型的异常" << endl;
38   }
39
40
41 }
42 int main()
43 {
44  try {
45  test01();
46  }
47  catch (char)
48  {
49  cout << "捕获了一个char类型的异常" << endl;
50  }
51 }
```

2.3 栈解旋

在try到throw之间定义的对象,在throw之后会被释放

```
1  #define _CRT_SECURE_NO_WARNINGS
2  #include <iostream>
3  #include <string.h>
4  #include <string>
5  using namespace std;
6  class Person
7  {
8  public:
9   Person(string name)
10   {
```

```
 cout << "构造" << endl;
 this->name = name;
 }
 ~Person()
 {
 cout << "析构" << endl;
 }
 string name;
};
void fun()
{
 Person p2("bob");
 Person p3("peter");
 cout << "001" << endl;
 throw 1;
}

void test01()
{
 try
 {
 Person p1("lucy");
 fun();
 }
 catch (int)
 {
 cout << "002" << endl;
 cout << "捕获到异常" << endl;
 }

}

int main()
{
 test01();
 return 0;
}
```

2.4异常接口的声明

```
//可抛出所有类型异常
void TestFunction01(){
```

```
throw 10;
}

//只能抛出int char char*类型异常
void TestFunction02() throw(int,char,char*){
string exception = "error!";
throw exception;
 }

 //不能抛出任何类型异常
 void TestFunction03() throw(){
 throw 10;
 }

 int main(){

 try{
 //TestFunction01();
 //TestFunction02();
 //TestFunction03();
 }
 catch (...){
 cout << "捕获异常!" << endl;
 }

 system("pause");
 return EXIT_SUCCESS;
 }
```

2.5异常变量生命周期

抛出的匿名对象的生命周期在catch里面

```
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <string.h>
#include <string>
using namespace std;
class Myexception
{
public:
 Myexception()
```

```
 {
 cout << "构造函数" << endl;
 }
 ~Myexception()
 {
 cout << "析构函数" << endl;
 }
 void error()
 {
 cout << "my error" << endl;
 }

};
void fun()
{
 Myexception p1;
 //throw Myexception();//如果抛出匿名对象 他的声明周期在catch里面
 throw p1;//p1声明周期在throw之后
}
void test01()
{
 try {
 fun();

 }
 catch (Myexception &p)
 {
 p.error();
 }
}
int main()
{
 test01();
 return 0;
}
```

2..6 异常的多态使用

```
#define _CRT_SECURE_NO_WARNINGS
```

```cpp
#include <iostream>
#include <string.h>
#include <string>
using namespace std;
//基类
class Myexception
{
public:
 virtual void error() = 0;
};

class Out_of_range:public Myexception
{
public:
 void error()
 {
 cout << "Out_of_range" << endl;
 }
};

class Bad_cast :public Myexception
{
public:
 void error()
 {
 cout << "Bad_cast" << endl;
 }
};
void fun()
{
 //throw Out_of_range();
 throw Bad_cast();
}
void test01()
{
 try
 {
 fun();
 }
 catch (Myexception &p)
```

```
 {
 p.error();
 }
}
int main()
{
 test01();
 return 0;
}
```

3 c++的异常库

3.1c++异常库的使用

```
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <string.h>
#include <string>
//exception
#include <stdexcept>


using namespace std;

 void fun()
 {
  /**/
  // throw out_of_range("越界");
  throw invalid_argument("段错误");
 }
 void test01()
 {
  try
  {
  fun();
  }
  catch (exception &p)
  {
  cout << p.what() << endl;
```

```
 }
}
int main()
{
 test01();
 return 0;
}
```

3.2 编写自己的异常类

```
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <string.h>
#include <string>
//exception
#include <stdexcept>
using namespace std;
class Longlongerror :public exception
{
public:
 Longlongerror(string data)
 {
 this->data = data;
 }
 Longlongerror(char * data)
 {
 this->data = data;
 }
 const char * what() const
 {
 return data.c_str();
 }
 string data;

};

void fun()
{
 throw Longlongerror("长长的错误");
}
```

```
31 void test01()
32 {
33  try
34  {
35  fun();
36  }
37  catch (exception &p)
38  {
39  cout << p.what() << endl;
40  }
41 }
42 int main()
43 {
44  test01();
45  return 0;
46 }
```