

## 1 异常的基本概念

---

## 2 c++处理异常

### 2.1 c++异常基本语法

---

### 2.2异常严格类型匹配

---

### 2.3 栈解旋

---

### 2.4异常接口的声明

### 2.5异常变量生命周期

---

### 2.6 异常的多态使用

---

## 3 c++的异常库

---

### 3.1c++异常库的使用

### 3.2 编写自己的异常类

# 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

```
1  #define _CRT_SECURE_NO_WARNINGS
2  #include <iostream>
3  #include <string.h>
4  #include <string>
5  using namespace std;
6  int mydive(int a, int b)
7  {
8      if (b == 0)
9          return -1; //errno = 2
10
11     return a / b;
12 }
13
14 void test01()
```

```

15 {
16     int ret = mydive(1, -1);
17     if (ret == -1)
18     {
19         cout << "除数为0" << endl; //perror("");
20
21     }
22 }
23
24 int main() {
25
26     test01();
27
28 }

```

## 2 c++处理异常

### 2.1 c++异常基本语法

```

1 #define _CRT_SECURE_NO_WARNINGS
2 #include <iostream>
3 #include <string.h>
4 #include <string>
5 using namespace std;
6 int mydive(int a, int b)
7 {
8     if (b == 0)
9         throw 'a'; //处理异常 抛出异常 抛出一个类型
10
11     return a / b;
12 }
13 void test01()
14 {
15     //尝试捕获异常
16     try
17     {
18         mydive(2,0);
19     }
20     catch (char) //如果没有捕获的抛出的异常 程序会被终止
21     {
22         //cout << "捕获了一个char类型的异常" << endl;

```

```

23     throw 'a';
24
25 }
26
27 }
28 int main()
29 {
30     try {
31         test01();
32     }
33     catch (char)
34     {
35         cout << "捕获了一个char类型的异常" << endl;
36     }
37 }

```

## 2.2异常严格类型匹配

```

1  #define _CRT_SECURE_NO_WARNINGS
2  #include <iostream>
3  #include <string.h>
4  #include <string>
5  using namespace std;
6  int mydive(int a, int b)
7  {
8      string str = "hello";
9      if (b == 0)
10         throw str; //处理异常 抛出异常 抛出一个类型
11
12     return a / b;
13 }
14 void test01()
15 {
16     //尝试捕获异常
17     try
18     {
19         mydive(2,0);
20     }
21     catch (char) //如果没有捕获的抛出的异常 程序会被终止
22     {
23         //cout << "捕获了一个char类型的异常" << endl;
24         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    {

```

```

11  cout << "构造" << endl;
12  this->name = name;
13  }
14  ~Person()
15  {
16  cout << "析构" << endl;
17  }
18  string name;
19  };
20  void fun()
21  {
22  Person p2("bob");
23  Person p3("peter");
24  cout << "001" << endl;
25  throw 1;
26  }
27
28  void test01()
29  {
30  try
31  {
32  Person p1("lucy");
33  fun();
34  }
35  catch (int)
36  {
37  cout << "002" << endl;
38  cout << "捕获到异常" << endl;
39  }
40
41  }
42
43  int main()
44  {
45  test01();
46  return 0;
47  }

```

## 2.4异常接口的声明

```

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

```

```

3  throw 10;
4  }
5
6  //只能抛出int char char*类型异常
7  void TestFunction02() throw(int,char,char*){
8  string exception = "error!";
9  throw exception;
10 }
11
12 //不能抛出任何类型异常
13 void TestFunction03() throw(){
14 throw 10;
15 }
16
17 int main(){
18
19 try{
20 //TestFunction01();
21 //TestFunction02();
22 //TestFunction03();
23 }
24 catch (...){
25 cout << "捕获异常!" << endl;
26 }
27
28 system("pause");
29 return EXIT_SUCCESS;
30 }

```

## 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

```

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

```

```

10  {
11  cout << "构造函数" << endl;
12  }
13  ~Myexception()
14  {
15  cout << "析构函数" << endl;
16  }
17  void error()
18  {
19  cout << "my error" << endl;
20  }
21
22 };
23 void fun()
24 {
25  Myexception p1;
26  //throw Myexception();//如果抛出匿名对象 他的声明周期在catch里面
27  throw p1;//p1声明周期在throw之后
28 }
29 void test01()
30 {
31  try {
32  fun();
33
34  }
35  catch (Myexception &p)
36  {
37  p.error();
38  }
39 }
40 int main()
41 {
42  test01();
43  return 0;
44 }

```

## 2..6 异常的多态使用

```

1  #define _CRT_SECURE_NO_WARNINGS

```

```
2 #include <iostream>
3 #include <string.h>
4 #include <string>
5 using namespace std;
6 //基类
7 class Myexception
8 {
9 public:
10     virtual void error() = 0;
11 };
12
13 class Out_of_range:public Myexception
14 {
15 public:
16     void error()
17     {
18         cout << "Out_of_range" << endl;
19     }
20 };
21
22 class Bad_cast :public Myexception
23 {
24 public:
25     void error()
26     {
27         cout << "Bad_cast" << endl;
28     }
29 };
30 void fun()
31 {
32     //throw Out_of_range();
33     throw Bad_cast();
34 }
35 void test01()
36 {
37     try
38     {
39         fun();
40     }
41     catch (Myexception &p)
```



```
42 {
43     p.error();
44 }
45 }
46 int main()
47 {
48     test01();
49     return 0;
50 }
```

## 3 c++的异常库

### 3.1c++异常库的使用

```
1 #define _CRT_SECURE_NO_WARNINGS
2 #include <iostream>
3 #include <string.h>
4 #include <string>
5 //exception
6 #include <stdexcept>
7
8
9 using namespace std;
10
11 void fun()
12 {
13     /**/
14     // throw out_of_range("越界");
15     throw invalid_argument("段错误");
16 }
17 void test01()
18 {
19     try
20     {
21         fun();
22     }
23     catch (exception &p)
24     {
25         cout << p.what() << endl;
```

```

26 }
27 }
28 int main()
29 {
30     test01();
31     return 0;
32 }

```

## 3.2 编写自己的异常类

```

1  #define _CRT_SECURE_NO_WARNINGS
2  #include <iostream>
3  #include <string.h>
4  #include <string>
5  //exception
6  #include <stdexcept>
7  using namespace std;
8  class Longlongerror :public exception
9  {
10 public:
11     Longlongerror(string data)
12     {
13         this->data = data;
14     }
15     Longlongerror(char * data)
16     {
17         this->data = data;
18     }
19     const char * what() const
20     {
21         return data.c_str();
22     }
23     string data;
24
25 };
26
27 void fun()
28 {
29     throw Longlongerror("长长的错误");
30 }

```

```
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 }
```