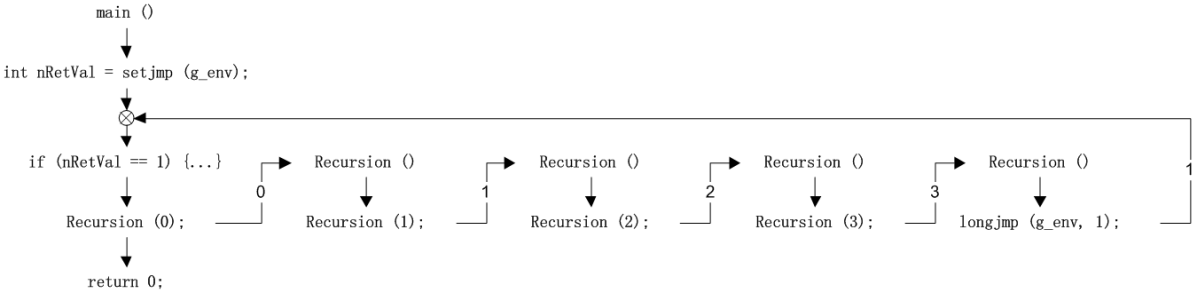
6 异常

6.1 非本地控制转移

6.1.1 setjmp-longjmp机制

```cpp
// longjmp.cpp

// setjmp-longjmp机制

#include <setjmp.h>

#include <iostream>

using namespace std;

jmp_buf jb;

class A {
public:
    A(int n) : n(new int(n)) {
        cout << "A::A(" << *this->n << ") invoked" << endl;
    }

    ~A(void) {
        cout << "A::~A(" << *n << ") invoked" << endl;
        delete n;
    }

private:
    int* n;
};

void foo(void) {
    A a(3);
    longjmp(jb, 404);
}

void bar(void) {
    A a(2);
    foo();
}

void hum(void) {
    A a(1);
    bar();
}

int main(void) {
    int errcode = 0;

    if (errcode = setjmp(jb)) {
        cout << "Some error(" << errcode << ") happened!" << endl;
```

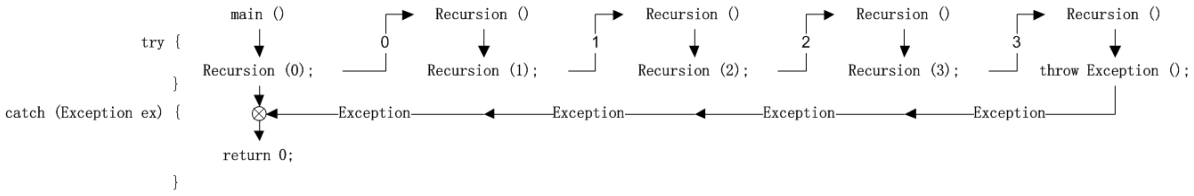
```
48          return -1;
49      }
50
51      hum();
52
53      return 0;
54  }
```



6.2.2 try-catch机制

```
1   // exception.cpp
2
3   // try-catch机制
4
5   #include <iostream>
6
7   using namespace std;
8
9   class A {
10  public:
11      A(int n) : n(new int(n)) {
12          cout << "A::A(" << *this->n << ") invoked" << endl;
13      }
14
15      ~A(void) {
16          cout << "A::~A(" << *n << ") invoked" << endl;
17          delete n;
18      }
19
20  private:
21      int* n;
22  };
23
24  void foo(void) {
25      A a(3);
26      throw 404;
27  }
28
29  void bar(void) {
30      A a(2);
31      foo();
32  }
33
34  void hum(void) {
35      A a(1);
36      bar();
```

```
}

int main(void) {
    try {
        hum();
    }
    catch (int errcode) {
        cout << "Some error(" << errcode << ") happened!" << endl;
        return -1;
    }

    return 0;
}
```



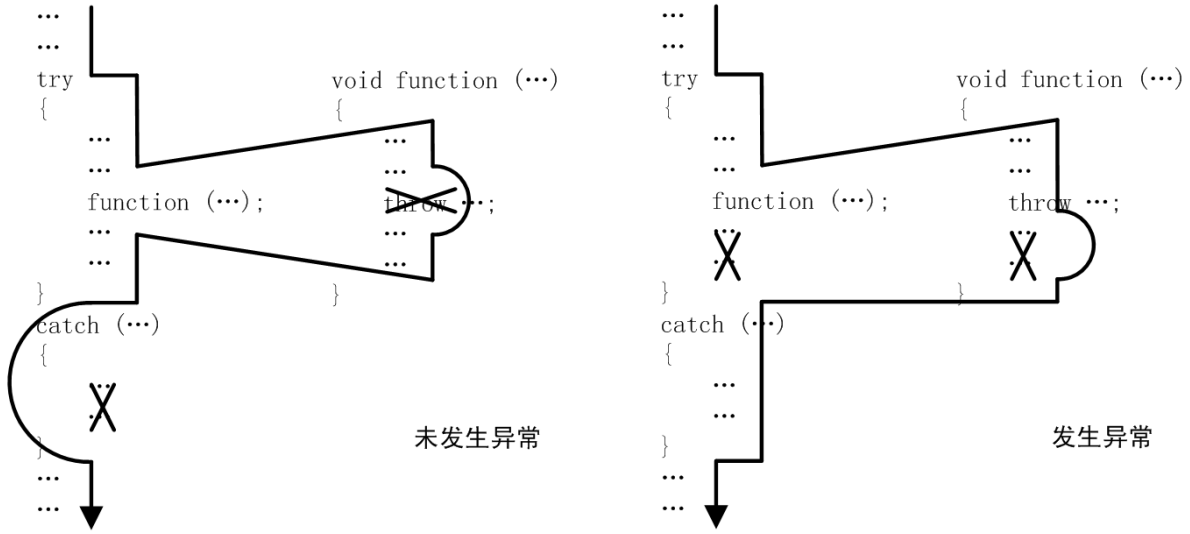
6.2.3 几种错误处理机制对比

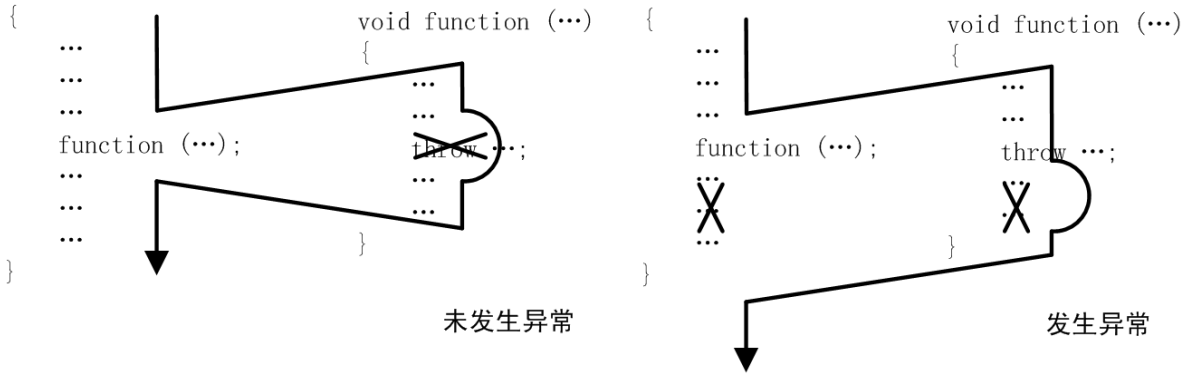
对比项	逐层检查返回值	setjmp-longjmp机制	try-catch机制
流程	复杂	简单	简单
栈对象	全析构	不析构	全析构

6.2 异常处理流程

```
// exflow.cpp

// 异常处理流程

#include <iostream>

using namespace std;

void foo(void) {
    cout << "Before throw" << endl; // 执行

    throw -1;

    cout << "After throw" << endl; // 不执行
}

int main(void) {
    try {
        cout << "Before invoke" << endl; // 执行

        foo();

        cout << "After invoke" << endl; // 不执行
    }
```

```
    catch (int ex) {
        cout << "Caught exception" << endl; // 执行
    }

    cout << "After try-catch" << endl; // 执行

    return 0;
}
```





6.3 异常处理模式

- throw语句抛出基本类型对象，catch子句捕获基本类型异常

```
// usage1.cpp

// throw语句抛出基本类型对象，catch子句捕获基本类型异常

#include <iostream>

using namespace std;

void login(const string& username, const string& password) {
    if (username != "tarena")
        throw 1;

    if (password != "123456")
        throw 2;
```

```
}

int main(int argc, char* argv[]) {
    if (argc < 3) {
        cout << "Usage: " << argv[0] << " <username> <password>" <<
endl;
        return -1;
    }

    try {
        login(argv[1], argv[2]);
    }
    catch (int ex) {
        if (ex == 1)
            cout << "Username error" << endl;
        else
        if (ex == 2)
            cout << "Password error" << endl;

        cout << "Login fail" << endl;

        return -1;
    }

    cout << "Login success" << endl;

    return 0;
}
```

- throw语句抛出类类型对象，catch子句捕获类类型异常

```
// usage2.cpp

// throw语句抛出类类型对象，catch子句捕获类类型异常

#include <iostream>

using namespace std;

class UsernameException {};
class PasswordException {};

void login(const string& username, const string& password) {
    if (username != "tarena")
        throw UsernameException();

    if (password != "123456")
        throw PasswordException();
}

int main(int argc, char* argv[]) {
    if (argc < 3) {
        cout << "Usage: " << argv[0] << " <username> <password>" <<
endl;
        return -1;
    }
```

```
25
26      try {
27          login(argv[1], argv[2]);
28      }
29      catch (UsernameException) {
30          cout << "Username error" << endl;
31          cout << "Login fail" << endl;
32          return -1;
33      }
34      catch (PasswordException) {
35          cout << "Password error" << endl;
36          cout << "Login fail" << endl;
37          return -1;
38      }
39
40      cout << "Login success" << endl;
41
42      return 0;
43  }
```

- 对于存在继承关系的异常，对子类异常的捕获一定要先于对基类异常的捕获

```
1   // usage3.cpp
2
3   // 对于存在继承关系的异常，对子类异常的捕获一定要先于对基类异常的捕获
4
5   #include <iostream>
6
7   using namespace std;
8
9   class UsernameException {};
10  class PasswordException : public UsernameException {};
11
12  void login(const string& username, const string& password) {
13      if (username != "tarena")
14          throw UsernameException();
15
16      if (password != "123456")
17          throw PasswordException();
18  }
19
20  int main(int argc, char* argv[]) {
21      if (argc < 3) {
22          cout << "Usage: " << argv[0] << " <username> <password>" <<
    endl;
23          return -1;
24      }
25
26      try {
27          login(argv[1], argv[2]);
28      }
29      catch (PasswordException) { // 子类
30          cout << "Password error" << endl;
31          cout << "Login fail" << endl;
32          return -1;
33      }
```

```
    catch (UsernameException) { // 基类
        cout << "Username error" << endl;
        cout << "Login fail" << endl;
        return -1;
    }

    cout << "Login success" << endl;

    return 0;
}
```

- 在catch块中重新抛出异常

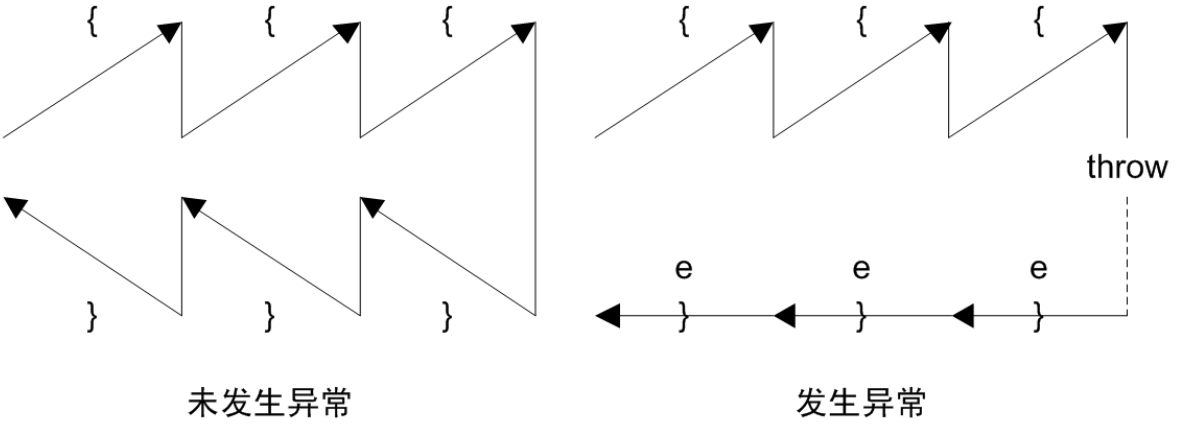
```
// usage4.cpp

// 在catch块中重新抛出异常

#include <iostream>

using namespace std;

class UsernameException {};
class PasswordException {};
class DownloadException {};

void login(const string& username, const string& password) {
    if (username != "tarena")
        throw UsernameException();

    if (password != "123456")
        throw PasswordException();
}

void download(const string& username, const string& password) {
    try {
        login(username, password);
    }
    catch (UsernameException) {
        cout << "Username error" << endl;
        cout << "Login fail" << endl;
        throw DownloadException();
    }
    catch (PasswordException) {
        cout << "Password error" << endl;
        cout << "Login fail" << endl;
        throw DownloadException();
    }

    cout << "Login success" << endl;

    cout << "Downloading..." << endl;
}

int main(int argc, char* argv[]) {
    if (argc < 3) {
```

```
        cout << "Usage: " << argv[0] << " <username> <password>" <<
endl;
        return -1;
    }

    try {
        download(argv[1], argv[2]);
    }
    catch (DownloadException) {
        cout << "Download fail" << endl;
        return -1;
    }

    cout << "Download success" << endl;

    return 0;
}
```

- 一个函数可以收到被其调用的函数所抛出的异常，但不一定非要捕获并处理该异常

```
// usage5.cpp

// 一个函数可以收到被其调用的函数所抛出的异常，但不一定
// 非要捕获并处理该异常。该异常将从此函数中继续向上抛出

#include <iostream>

using namespace std;

class UsernameException {};
class PasswordException {};

void login(const string& username, const string& password) {
    if (username != "tarena")
        throw UsernameException();

    if (password != "123456")
        throw PasswordException();
}

void download(const string& username, const string& password) {
    login(username, password);

    cout << "Downloading..." << endl;
}

int main(int argc, char* argv[]) {
    if (argc < 3) {
        cout << "Usage: " << argv[0] << " <username> <password>" <<
endl;
        return -1;
    }

    try {
        download(argv[1], argv[2]);
    }
```

```
    catch (UsernameException) {
        cout << "Username error" << endl;
        cout << "Download fail" << endl;
        return -1;
    }
    catch (PasswordException) {
        cout << "Password error" << endl;
        cout << "Download fail" << endl;
        return -1;
    }

    cout << "Download success" << endl;

    return 0;
}
```



- 包含更多有用信息的异常

```
// usage6.cpp

// 包含更多有用信息的异常

#include <iostream>

using namespace std;

class Exception {
public:
    Exception(const char* file, int line, const char* func, const char*
desc)
        : file(file), line(line), func(func), desc(desc) {}

private:
    string file;
    int line;
    string func;
    string desc;

    friend ostream& operator<<(ostream& os, const Exception& ex) {
        return os << ex.file << ':' << ex.line << '(' << ex.func << "):
" << ex.desc;
    }
};
```

```
24
25  class UsernameException : public Exception {
26  public:
27      UsernameException(const char* file, int line, const char* func)
28          : Exception(file, line, func, "Username error") {}
29  };
30
31  class PasswordException : public Exception {
32  public:
33      PasswordException(const char* file, int line, const char* func)
34          : Exception(file, line, func, "Password error") {}
35  };
36
37  void login(const string& username, const string& password) {
38      if (username != "tarena")
39          throw UsernameException(__FILE__, __LINE__, __func__);
40
41      if (password != "123456")
42          throw PasswordException(__FILE__, __LINE__, __func__);
43  }
44
45  int main(int argc, char* argv[]) {
46      if (argc < 3) {
47          cout << "Usage: " << argv[0] << " <username> <password>" <<
    endl;
48          return -1;
49      }
50
51      try {
52          login(argv[1], argv[2]);
53      }
54      catch (const UsernameException& ex) {
55          cout << ex << endl;
56          cout << "Login fail" << endl;
57          return -1;
58      }
59      catch (const PasswordException& ex) {
60          cout << ex << endl;
61          cout << "Login fail" << endl;
62          return -1;
63      }
64
65      cout << "Login success" << endl;
66
67      return 0;
68  }
```

    - 异常对象最好以引用而非对象的形式捕获，以避免因对象复制增加额外的开销，或引发新的异常
- 通过修改异常对象的状态，跟踪异常处理过程

```
1  // usage7.cpp
2
3  // 通过修改异常对象的状态，跟踪异常处理的过程
4
```

```cpp
#include <iostream>

using namespace std;

void foo(void) {
    throw string(__func__);
}

void bar(void) {
    try {
        foo();
    }
    catch (string& ex) {
        throw (ex += "->") += __func__;
    }
}

void hum(void) {
    try {
        bar();
    }
    catch (string& ex) {
        throw (ex += "->") += __func__;
    }
}

int main(void) {
    try {
        hum();
    }
    catch (string& ex) {
        cout << ((ex += "->") += __func__) << endl;
        return -1;
    }

    return 0;
}
```

- 捕获其它异常

```cpp
// usage8.cpp

// 捕获其它异常

#include <iostream>

using namespace std;

class UsernameException {};
class PasswordException {};
class HostnameException {};

void login(const string& username, const string& password) {
    if (username != "tarena")
        throw UsernameException();

```

```
    if (password != "123456")
        throw PasswordException();
}

void download(const string& username, const string& password, const
string& hostname) {
    login(username, password);

    if (hostname != "www.tedu.cn")
        throw HostnameException();

    cout << "Downloading..." << endl;
}

int main(int argc, char* argv[]) {
    if (argc < 4) {
        cout << "Usage: " << argv[0] << " <username> <password>
<hostname>" << endl;
        return -1;
    }

    try {
        download(argv[1], argv[2], argv[3]);
    }
    catch (UsernameException) {
        cout << "Username error" << endl;
        cout << "Download fail" << endl;
        return -1;
    }
    catch (PasswordException) {
        cout << "Password error" << endl;
        cout << "Download fail" << endl;
        return -1;
    }
    catch (...) {
        cout << "Other error" << endl;
        cout << "Download fail" << endl;
        return -1;
    }

    cout << "Download success" << endl;

    return 0;
}
```

6.4 构造函数中的异常

- 构造函数中出现无法处理的问题时，可以抛出异常以告知对象的创建者
- 构造函数抛出异常，引发不完整构造，析构函数不会被调用，已经分配的资源需要在构造函数中释放，或使用智能指针

```
// consex.cpp

// 构造函数抛出异常
```

```
4
5  #include <memory>
6  #include <iostream>
7
8  using namespace std;
9
10 class A {
11 public:
12     A(void) {
13         cout << "A::A() invoked" << endl;
14     }
15
16     ~A(void) {
17         cout << "A::~A() invoked" << endl;
18     }
19 };
20
21 class B {
22 public:
23     B(void) {
24         cout << "B::B() invoked" << endl;
25     }
26
27     ~B(void) {
28         cout << "B::~B() invoked" << endl;
29     }
30 };
31
32 class C {
33 public:
34     C(void) {
35         cout << "C::C() invoked" << endl;
36     }
37
38     ~C(void) {
39         cout << "C::~C() invoked" << endl;
40     }
41 };
42
43 class D {
44 public:
45     D(void) {
46         cout << "D::D() invoked" << endl;
47     }
48
49     ~D(void) {
50         cout << "D::~D() invoked" << endl;
51     }
52 };
53
54 class E : public D {
55 public:
56     E(void) : b(new B), a(new A) {
57         cout << "E::E() invoked" << endl;
58
59         delete b; // 注意!
```

```
        throw 1;
    }

    ~E(void) {
        cout << "E::~E() invoked" << endl;

        delete b;
    }

private:
    C c;
    B* b;
    auto_ptr<A> a;
};

int main(void) {
    try {
        E e;
    }
    catch (int ex) {
        cout << "Caught an exception: " << ex << endl;
        return -1;
    }

    return 0;
}
```

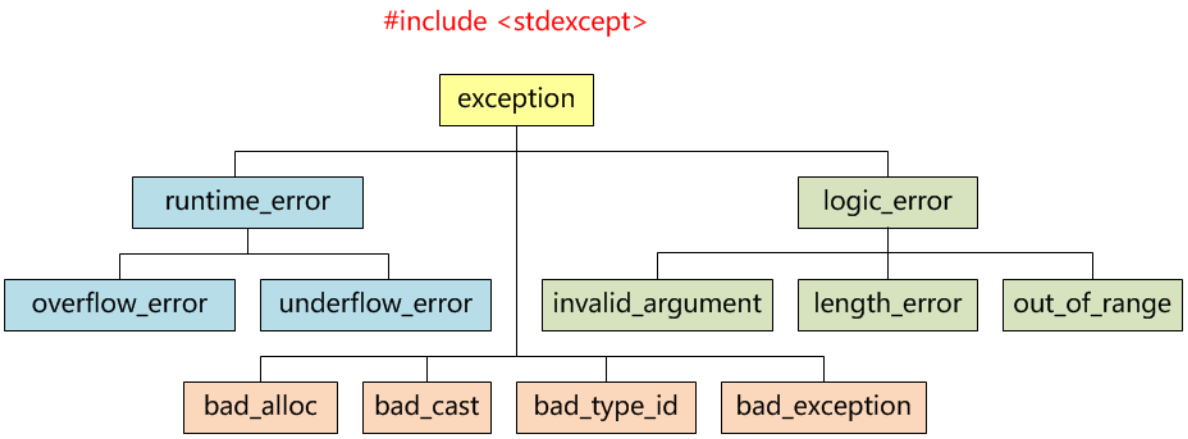
6.5 析构函数中的异常

- 不要主动抛出
- 尽量内部消化

```
// desex.cpp

// 析构函数不要抛出异常

#include <iostream>

using namespace std;

class A {
public:
    ~A(void) {
        cout << "A::~A() invoked" << endl;

        throw 1;
    }

    void foo(void) {
        cout << "A::foo() invoked" << endl;

        throw 2;
    }
};
```

```
23
24 int main(void) {
25     try {
26         A a;
27         a.foo();
28     }
29     catch (int ex) {
30         cout << "Caught an exception: " << ex << endl;
31         return -1;
32     }
33
34     return 0;
35 }
```

6.6 标准库异常



- 从标准库异常中派生自己的异常，并覆盖exception异常基类中的虚函数what，返回有关异常的描述字符串，同时以多态方式，将自己的异常和标准库中的各种异常一并捕获

```
1  // usage9.cpp
2
3  // 从标准库异常中派生自己的异常，并覆盖exception异常基
4  // 类中的虚函数what，返回有关异常的描述字符串，同时以
5  // 多态方式，将自己的异常和标准库中的各种异常一并捕获
6
7  #include <iostream>
8
9  using namespace std;
10
11 class UsernameException : public exception {
12 public:
13     const char* what(void) const noexcept {
14         return "Username error";
15     }
16 };
17
18 class PasswordException : public exception {
19 public:
20     const char* what(void) const noexcept {
21         return "Password error";
22     }
23 };
```

```cpp

void login(const string& username, const string& password) {
    if (username != "tarena")
        throw UsernameException();

    if (password != "123456")
        throw PasswordException();
}

int main(int argc, char* argv[]) {
    if (argc < 3) {
        cout << "Usage: " << argv[0] << " <username> <password>" << endl;
        return -1;
    }

    try {
        login(argv[1], argv[2]);
    }
    catch (const exception& ex) {
        cout << ex.what() << endl;
        cout << "Login fail" << endl;
        return -1;
    }

    cout << "Login success" << endl;

    return 0;
}
```