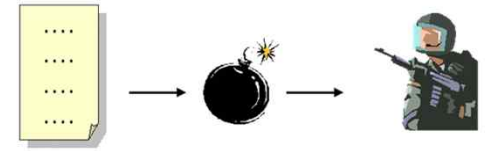
제12장 예외 처리

이번 장에서 학습할 내용

- 예외 처리의 개념
- 예외 처리기 구조
- 예외 전달
- 다중 catch 문장
- 자신의 예외 클래스 작성

예외란?

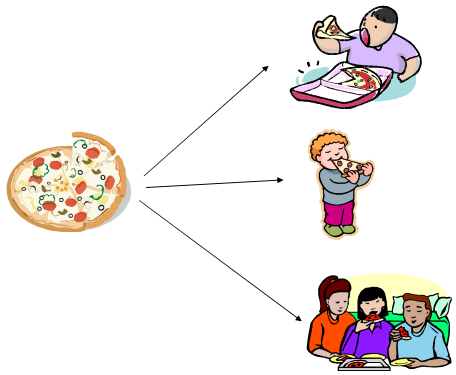
- 예외(exception): 잘못된 코드, 부정확한 데이터, 예외적인 상황에 의하여 발생하는 오류
  - (예) 0으로 나누는 것과 같은 잘못된 연산이나 배열의 인덱스가 한계를 넘을 수도 있고, 디스크에서는 하드웨어 에러가 발생할 수 있다.



예외는 처리하는 것이 좋다.

그림 11.1 예외 처리의 개요

피자 나누기 프로그램



피자 나누기 프로그램

```
#include <iostream>
using namespace std;
int main()
{
        int pizza_slices = 0;
        int persons = -1;
        int slices_per_person=0;

        cout << "피자 조각수를 입력하시오: ";
        cin >> pizza_slices;
        cout << "사람수를 입력하시오: ";
        cin >> persons;
        slices_per_person = pizza_slices / persons;
        cout << "한사람당 피자는" << slices_per_person << "입니다." << endl;

        return 0;
}
```

```
피자 조각수를 입력하시오: 12
사람수를 입력하시오: 4
한사람당 피자는 3입니다.
계속하려면 아무 키나 누르십시오 . . .
```


전통적인 오류 처리 방식

- If-else을 사용하여 조건을 검사
- 정상적인 코드와 오류 처리 처리 분리가 어려움

```
if (pizza_slices < 0) {
                cout << "피자조각이 음수임";
        } else if (pizza_slices == 0) {
                cout << "피자조각이 0임";
        } else {
                if (persons == 0) {
                        cout << "사람이 0명입니다. "<< endl;
                } else if (persons < 0) {
                        cout << "사람이 음수입니다. " << endl;
                } else {
                        slices_per_person = pizza_slices / persons;
                        cout << "한사람당 피자는" << slices_per_person <<
"입니다." << endl;
                }
        }
```


중간 점검 문제

1. 예외는 어떤 경우에 발생하는가?
2. 예외를 처리하는 경우와 처리하는 않은 경우를 비교하여 보라. 장점은 무엇인가?.

예외 처리기

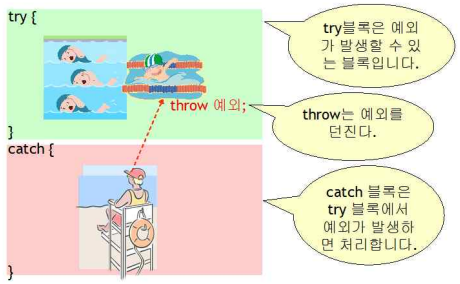


그림 11.2 try블록은 예외가 발생할 수 있는 위험한 코드이다. catch 블록은 예외를 처리하는 코드이다.

피자 나누기 프로그램

```
int main()
{
        int pizza_slices = 0;
        int persons = -1;
        int slices_per_person=0;
        try
        {
                cout << "피자조각 수를 입력하시오: ";
                cin >> pizza_slices;
                cout << "사람수를 입력하시오: ";
                cin >> persons;

                if (persons == 0)
                        throw persons;
                slices_per_person = pizza_slices / persons;
              cout << "한사람당 피자는" << slices_per_person << "입니다." << endl;
        }
        catch(int e)
        {
                cout << "사람이" << e << " 명입니다. "<< endl;
        }
        return 0;
}
```

실행 결과

```
피자 조각수를 입력하시오: 12
사람수를 입력하시오: 4
한사람당 피자는 3입니다.
계속하려면 아무 키나 누르십시오 . . .
```

```
피자 조각수를 입력하시오: 12
사람수를 입력하시오: 0
사람이 0 명 입니다.
계속하려면 아무 키나 누르십시오 . . .
```


try/catch 블록에서의 실행 흐름

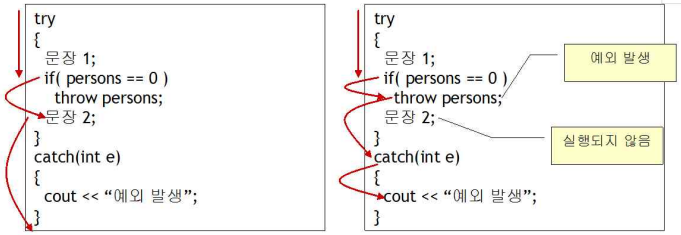


그림 15.3 try/catch 블록에서의 실행 흐름

catch 블록의 매개 변수

```
try
{
  문장 1;
  if( persons == 0 )
    throw persons;
  문장 2;
}
catch(int  e)
{
  cout << “예외 발생”;
}
```

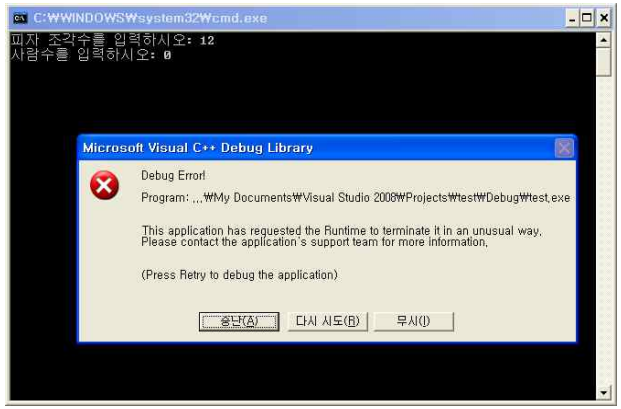
예외 처리기의 매개 변수

타입이 일치되는 예외만 처리

```
try
   {
        int person =0;
        …
        if (persons == 0)
                throw persons;
        …
   }
   catch(char e)
   {
        cout << "사람이 " << e << " 명 입니다. "<< endl;
   }
```

타입이 일치하지 않음

실행 결과



모든 타입의 예외를 잡고 싶으면

```
catch(...)
{
    // 모든 예외를 잡아서 처리할 수 있다.
}
```

중간 점검 문제

1. try 블록에서 예외가 발생한 지점 이후의 문장들은 실행되는가?
2. catch 블록에서 모든 예외를 다 잡으려면 매개 변수를 어떻게 정의하는가?

예외 전달

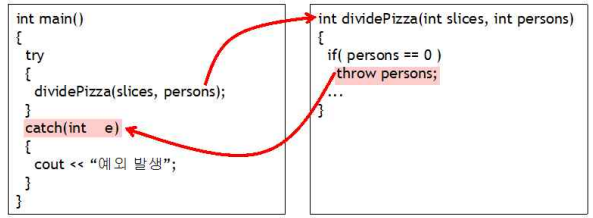


그림 15.5 예외는 함수를 넘어서 전달될 수 있다.

예외 전달

pizza4.cpp

```
#include <iostream>
using namespace std;
int dividePizza(int pizza_slices, int persons);

int main()
{
   int pizza_slices = 0;
   int persons = 0;
   int slices_per_person=0;

   try
   {
        cout << "피자 조각수를 입력하시오: ";
        cin >> pizza_slices;
        cout << "사람수를 입력하시오: ";
        cin >> persons;
        slices_per_person =dividePizza(pizza_slices, persons);
        cout << "한사람당 피자는 " << slices_per_person << "입니다." << endl;
   }
```


예외 전달

```
   catch(int e)
   {
        cout << "사람이 " << e << " 명 입니다. "<< endl;
   }
   return 0;
}
int dividePizza(int pizza_slices, int persons)
{
        if (persons == 0)
                throw persons;
        return  pizza_slices / persons;
}
```

실행 결과

정상적인 실행결과

```
피자 조각수를 입력하시오: 12
사람수를 입력하시오: 4
한사람당 피자는 3입니다.
계속하려면 아무 키나 누르십시오 . . .
```

예외 발생 실행결과

```
피자 조각수를 입력하시오: 12
사람수를 입력하시오: 0
사람이 0 명 입니다.
계속하려면 아무 키나 누르십시오 . . .
```


예외를 처리하고 다시 보내고자 할때

```
int dividePizza(int pizza_slices, int persons)
{
  try
  {
      if (persons == 0)
            throw persons;
      return  pizza_slices / persons;
  }
  catch(int e)
  {
      cout << "사람이  " << e << " 명  입니다(dividePizza). "<< endl;
      throw;
  }
}
```

예외 발생 실행결과

```
피자 조각수를 입력하시오: 12
사람수를 입력하시오: 0
사람이 0 명 입니다(dividePizza).
사람이 0 명 입니다.
계속하려면 아무 키나 누르십시오 . . .
```


함수 헤더에 예외 명시

- int dividePizza(int s, int p) throw () // 예외를 던지지 않는다.
- int dividePizza(int s, int p) throw (..) // 예외를 던진다. 타입은 지정하지않음
- int dividePizza(int s, int p) throw (int) // int 타입의 예외를 던진다.
- int dividePizza(int s, int p) throw (int, double)

중간 점검 문제

1. 예외를 처리했지만 호출한 함수에게도 예외를 보내줄 수 있는가?
2. 예외는 3개의 함수를 거쳐서도 전달될 수 있는가?

다중 catch 문장

- 하나의 try 블록에서는 여러 개의 throw 문장을 가질 수 있다.
- 여러 가지 타입의 값을 처리하려면 여러 개의 catch 블록을 두어야 한다.
- 예를 들어서 피자 나누기 예제에서 사람 수가 0이 될 수도 있고 사람 수가 음수가 될 수도 있다. 이것을 구분하여서 처리하려면 다음과 같이 두 개의 catch 블록을 정의하여야 한다.

pizza4.cpp

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

int main()
{
   int pizza_slices = 12;
   int persons = 0;
   int slices_per_person=0;

   try {
        cout << "피자 조각수를 입력하시오: ";
        cin >> pizza_slices;
        cout << "사람수를 입력하시오: ";
        cin >> persons;

        if( persons < 0 ) throw "negative";          // 예외 발생!
        if( persons == 0 ) throw persons;            // 예외 발생!
        slices_per_person = pizza_slices / persons;
        cout << "한사람당 피자는 " << slices_per_person << "입니다." << endl;
   }
   catch (const char *e) {      char 타입의 예외만 처리
        cout << "오류: 사람수가 " << e << "입니다" << endl;
   }
   catch (int e) {              int 타입의 예외만 처리
        cout << "오류: 사람이 " << e << " 명입니다." << endl;
   }
   return 0;
}
```


실행 결과

예외 발생 실행결과

```
피자 조각수를 입력하시오: 12
사람수를 입력하시오: 0
사람이 0 명 입니다.
계속하려면 아무 키나 누르십시오 . . .
```


구체적인 예외를 먼저 잡는다.

```
try {
   getIput();
}
catch(TooSmallException e) {
   //TooSmallException만 잡힌다.
}
catch(...) {
   //TooSmallException을 제외한 나머지 예외들이 잡힌다.
}
```

반대로 하면

```
try {
   getInput();
}
catch(...) {
   //모든 예외들이 잡힌다.
}
catch(TooSmallException e) {
   //아무 것도 잡히지 않는다!
}
```


중간 점검 문제

1. 발생된 예외와 catch 블록의 매개 변수는 어떤 규칙에 의하여 매칭되는가?
2. 어떤 타입이라도 catch 블록의 매개 변수로 지정할 수 있는가?

자신의 예외 클래스 작성

- throw 문장은 클래스 타입의 객체도 던질 수 있다.
- 예외 클래스도 단지 하나의 클래스에 불과

(예) throw NoPersonException(persons);

예외를 위한
클래스의 객체

예제

pizza4.cpp

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


class NoPersonException
{
public:
   NoPersonException();
   NoPersonException(int p) { persons = p; };
   int get_persons() { return persons; };
private:
   int persons;
};
```

예제

```
int main()
{
   int pizza_slices = 12;
   int persons = -1;
   int slices_per_person=0;

   try {
        cout << "피자 조각수를 입력하시오: ";
        cin >> pizza_slices;
        cout << "사람수를 입력하시오: ";
        cin >> persons;
        if( persons <= 0 ) throw NoPersonException(persons);          // 예외 발생!
        slices_per_person = pizza_slices / persons;
        cout << "한사람당 피자는 " << slices_per_person << "입니다." << endl;
   }
   catch (NoPersonException e)
   {
        cout << "오류: 사람이 " << e.get_persons() << "명 입니다" << endl;
   }
   return 0;
}
```

실행 결과

예외 발생 실행결과

```
피자 조각수를 입력하시오: 12
사람수를 입력하시오: 0
사람이 0 명 입니다.
계속하려면 아무 키나 누르십시오 . . .
```


상속 관계에 있는 예외 클래스

pizza4.cpp

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

class ParentException
{
public:
   void display() { cout << "ParentException" << endl; }
};
class ChildException : public ParentException
{
public:
   void display() { cout << "ChildException" << endl; }
};
```


상속 관계에 있는 예외 클래스

```
int main()
{
   try {
        throw ChildException();
   }
   catch (ParentException& e)
   {
        e.display();
   }
   catch (ChildException& e)
   {
        e.display();
   }
   return 0;
}
```

예외 발생 실행결과

```
ParentException
계속하려면 아무 키나 누르십시오 . . .
```

중간 점검 문제

1. catch 블록의 매개 변수를 레퍼런스로 정의하게 되면 어떤 경우에 유용한가?
2. 클래스 A의 객체를 매개 변수로 받는 catch 블록을 정의하여 보라.

37/40