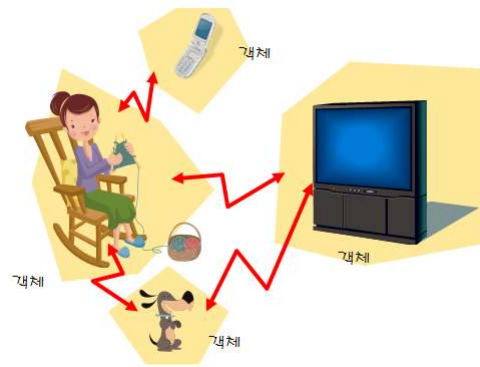




C++ Espresso

제11장 예외 처리와 형변환



© 2010 인피니티박스 All rights reserved



이번 장에서 학습할 내용

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

예외는 오류가
발생하더라도
오류를
우아하게
처리하게
합니다.



© 2010 인피니티박스 All rights reserved





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.

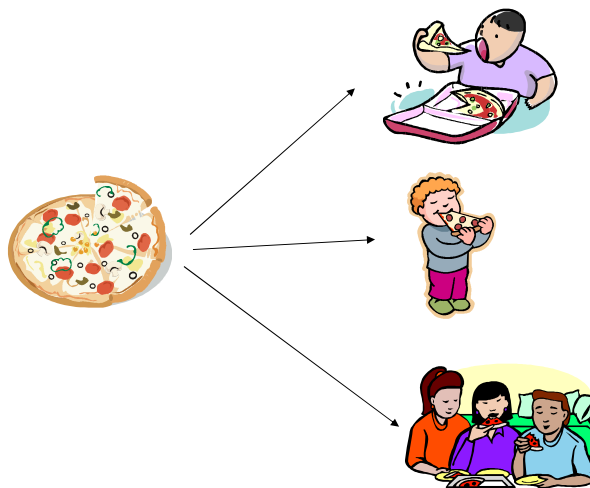


그림 11.1 예외 처리의 개요

© 2010 인피니티박스 All rights reserved



피자 나누기 프로그램



© 2010 인피니티박스 All rights reserved





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

    return 0;
}
```



0 이나 음수가 피자 조각수,
사람수로 입력되는 경우는 ?



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

© 2010 인피니티박스 All rights reserved



전통적인 오류 처리 방식(코드 복잡)

- If-else를 사용하여 조건을 검사
- 정상적인 코드(논리)와 오류 처리 분리가 어려움 → 코드 복잡
 - if 문은 일반적으로 프로그램의 논리를 구현하는데 주로 사용

```
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;
    }
}
```

© 2010 인피니티박스 All rights reserved





예외 처리기

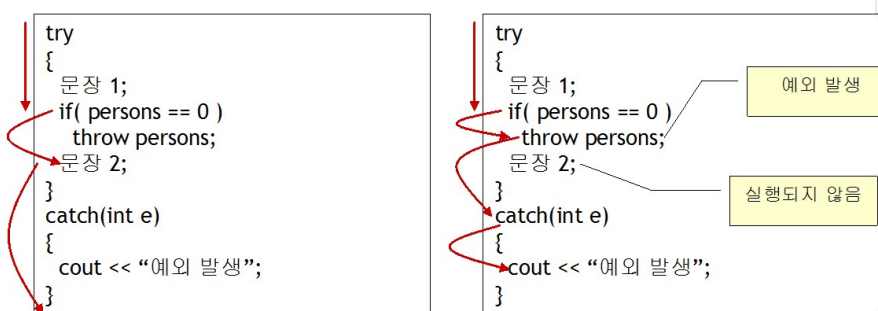


그림 11.2 try블록은 예외가 발생할 수 있는 위험한 코드이다. catch 블록은 예외를 처리하는 코드이다.

© 2010 인피니티박스 All rights reserved



try/catch 블록에서의 실행 흐름



예외가 발생하지 않은 경우

예외가 발생한 경우

그림 15.3 try/catch 블록에서의 실행 흐름

© 2010 인피니티박스 All rights reserved





catch 블록의 매개 변수

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

함수와 비슷

예외 처리기의 매개 변수

© 2010 인피니티박스 All rights reserved



피자 나누기 프로그램



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

    while(1){
        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; //예외처리 코드 부분
        }
    }

    return 0;
}
```



//정상코드 부분(괄弧)

반환형 없는 함수와 비슷

© 2010 인피니티박스 All rights reserved





피자 나누기 프로그램(잘못된 것)

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

    while(1){
        try {
            cout << "피자조각 수를 입력하시오: ";
            cin >> pizza_slices;
            cout << "사람수를 입력하시오: ";
            cin >> persons;

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

    // person=0 인 경우, catch 수행 후 문장들 수행 → 오류
    return 0;
}
```



연관있는 것들을 하나의
try 블록으로 묶어야 한다.

© 2010 인피니티박스 All rights reserved



실행 결과



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



피자 조각수를 입력하시오: 12
사람수를 입력하시오: 0
사람이 0 명입니다.
계속하려면 아무 키나 누르십시오... → 오류 발생, 종료

© 2010 인피니티박스 All rights reserved





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, ERROR

© 2010 인피니티북스 All rights reserved




모든 타입의 예외를 잡고 싶으면



```
catch(...) // 매개변수를 “...” 로 표기하면 모든 종류의 예외를 처리
{
    // 모든 type의 예외를 잡아서 처리할 수 있다.
}
```

© 2010 인피니티북스 All rights reserved





예외 전달

예외 던지는 부분 없다

```

int main()
{
    try
    {
        dividePizza(slices, persons);
    }
    catch(int e)
    {
        cout << "예외 발생";
    }
}

```

```


int dividePizza(int slices, int persons)
{
    if( persons == 0 )
        throw persons;
    ...
}

```

그림 15.5 예외는 함수를 넘어서 전달될 수 있다.

🔊

© 2010 인피니티박스 All rights reserved



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;

```

🔊

© 2010 인피니티박스 All rights reserved



예외 전달

```
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;
}
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;
}
```

예외 던지는 부분 없다

호출한 함수에서 예외 전달

© 2010 인피니티박스 All rights reserved



실행 결과

정상적인 실행결과

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

예외 발생 실행결과

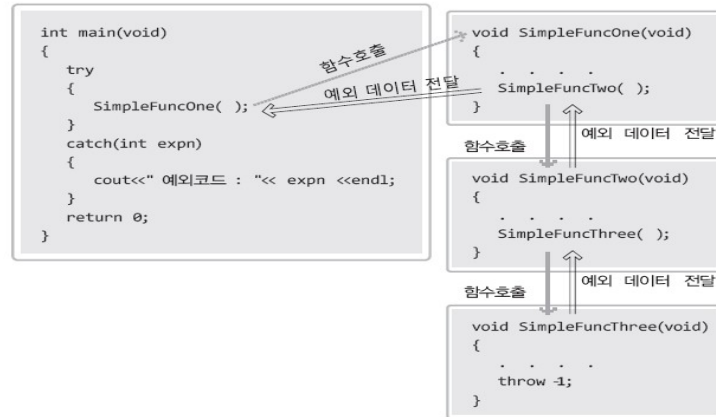
피자 조각수를 입력하시오: 12
사람수를 입력하시오: 0
사람이 0명 입니다.
계속하려면 아무 키나 누르십시오 ...

© 2010 인피니티박스 All rights reserved





함수를 통한 예외 전달



© 2010 인피니티박스 All rights reserved



함수를 통한 예외전달 예

```

class Resource {
public:
    Resource(int id) : id_(id) {}
    ~Resource() {
        cout << "리소스 해제 : " << id_ << endl;
    }
private:
    int id_;
};

int func3() {
    Resource r3(3);
    // runtime_error : 예외 객체 중 하나
    throw runtime_error("Exception from 3!\n");
}

int func2() {
    Resource r2(2);
    func3();
    cout << "실행 안됨!" << endl;
    return 0;
}

int func1() {
    Resource r1(1);
    func2();
    cout << "실행 안됨!" << endl;
    return 0;
}

int main() {
    try {
        func1();
    }
    catch (std::exception &e) {
        cout << "Exception : " << e.what();
    }
}
    
```

// 예외 전달되면 이후 문장 수행 안함,

std::exception : std 에 있는 예외 클래스 exception → 대부분 예외를 받을 수 있음

// 출력 ??
리소스 해제 : 3 ← func3의 r3 제거
리소스 해제 : 2 ← func3의 r2 제거
리소스 해제 : 1 ← func3의 r1 제거
Exception : Exception from 3!

© 2010 인피니티박스 All rights reserved





함수를 통한 예외 전달

```
main(...){
    ...
    try {
        dividePizza(...)
    }
    catch(...){
        ...
    }
}
```

```
dividePizza(...){
    ...
    try {
        ...
        throw person;
    }
    catch(...){
        ...
        throw;
    }
}
```

다음쪽 예제 :
함수에서
예외를
처리하고 전달

```
main(...){
    ...
    try {
        dividePizza(...)
    }
    catch(...){
        ...
    }
}
```

```
dividePizza(...){
    ...
    throw person;
}
```

앞의 예제 :
함수에서
예외를 전달

© 2010 인피니티박스 All rights reserved



예외 전달

자신이 예외를 처리하고 호출한 함수로 예외를 다시 보내고자 할때

```
int main()
{
    int pizza_slices = 0;
    int persons = 0;
    int slices_per_person=0;
    try
    {
        cout << "피자 조각수를 입력하십시오: ";
        cin >> pizza_slices;
        cout << "사람수를 입력하십시오: ";
        cin >> persons;
        slices_per_person = dividePizza(pizza_slices, persons);
        cout << "한사람당 피자는 " << slices_per_person << "입니다." << endl;
    }
    catch(int e)
    {
        cout << "사람이 " << e << " 명 입니다. " << endl;
    }
    return 0;
}
```

예외 던지는 부분 없다

© 2010 인피니티박스 All rights reserved





자신이 예외를 처리하고 호출한 함수로 예외를 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;
    }
}
```

가장 가까운 catch 로 이동

- 호출한 함수 Main 의 catch 로 이동
- throw 에서 e 없는 경우 자신이 받은 e 를 던짐

예외 발생 실행결과

피자 조각수를 입력하시오: 12

사람수를 입력하시오: 0

사람이 0 명 입니다(dividePizza).

사람이 0 명 입니다.

계속하려면 아무 키나 누르십시오 . . .

© 2010 인피니티박스 All rights reserved



함수 헤더에 예외 명시(의무 아님, 생략)

- `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)`

© 2010 인피니티박스 All rights reserved





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.

© 2010 인피니티박스 All rights reserved



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;
}
```

던지는 예외 type을 다르게 하여 구별하도록 작성

© 2010





실행 결과

예외 발생 실행결과

피자 조각수를 입력하시오: 12

사람수를 입력하시오: 0

사람이 0 명 입니다.

계속하려면 아무 키나 누르십시오 . . .

© 2010 인피니티박스 All rights reserved



구체적인 예외를 먼저 잡는다.

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

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

반대로 하면

© 2010 인피니티박스 All rights reserved





자신의 예외 클래스 작성

- **throw** 문장은 클래스 타입의 객체도 던질 수 있다.
 - 예외에 대한 여러 정보를 묶어서 클래스를 구성하고 던질 수 있다.
- 예외 클래스도 단지 하나의 클래스에 불과

(예) **throw NoPersonException(persons);**

예외를 위한 클래스의 객체
(개발자가 작성한 클래스, 다음쪽)

std::exception : std 에 있는 예외 클래스 exception → 대부분의 예외를 받을 수 있음

© 2010 인피니티박스 All rights reserved



예제

pizza4.cpp

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

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

© 2010 인피니티박스 All rights reserved





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;
}

```

- 생성자 호출하여 객체를 생성하고 전달, **person** 은 객체의 멤버변수에 저장
- 클래스의 멤버변수에 다양한 정보 저장하고 전달 가능



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; }
};
```

© 2010 인피니티박스 All rights reserved



상속 관계에 있는 예외 클래스

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

부모 것은 자신 것 뿐만 아니라
자식 것도 잡는다.
→ 자식 예외도 부모에 걸린다.
→ 자식에 걸리지 않음

예외 발생 실행결과

ParentException

계속하려면 아무 키나 누르십시오 . . .

© 2010 인피니티박스 All rights reserved





형변환(생략)

```
double f = 3.141592;  
int i = (int) f;
```

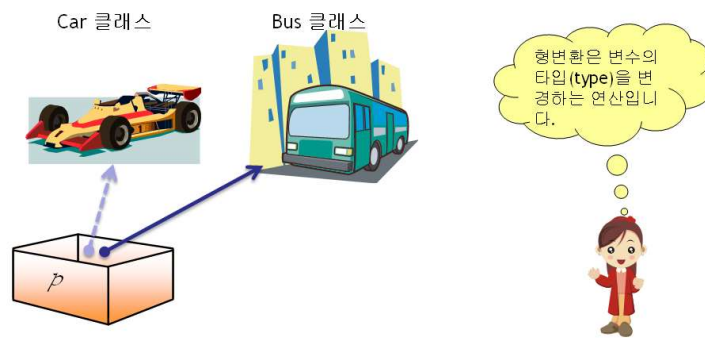


그림 11.6 형변환이란?

© 2010 인피니티박스 All rights reserved