14장 예외처리와 템플릿(1)

2020. 11. 26

순천향대학교 컴퓨터 공학과

내용

- C++의 예외 처리 개념
- 예외처리 고려사항
- 예외처리기 형식
- 다중 catch 문
- 예외 전파
- 예외 재발생
- 예외 클래스 작성
- 표준 예외

예외란?

- 예외(exception)는 프로그램 실행중에 잘못된 코드, 부정확한 데이터, 예외적인 상황이 발생되는 비정상적 이벤트
  - 하드웨어나 소프트웨어로 탐지 가능
  - Ex. 0으로 나누는 것, 실수 오버플로우, 배열의 인덱스가 한계를 넘을 수도 있고, 디스크에서는 하드웨어 읽기 에러, end-of-file 등
- 초창기 언어에서 예외가 발생하면, 프로그램 실행을 종료시키고, 제어를 운영체제에 넘김 (운영체제는 간단한 오류 메시지 출력)
- 예외를 탐지하고, 이에 대한 특별한 처리를 예외 처리(exception handling)라고 하며, 이러한 처리기를 예외 처리기(exception handler)라 한다.
- C++는 프로그램에서 예외를 처리할 수 있는 기능 제공

예외처리란?

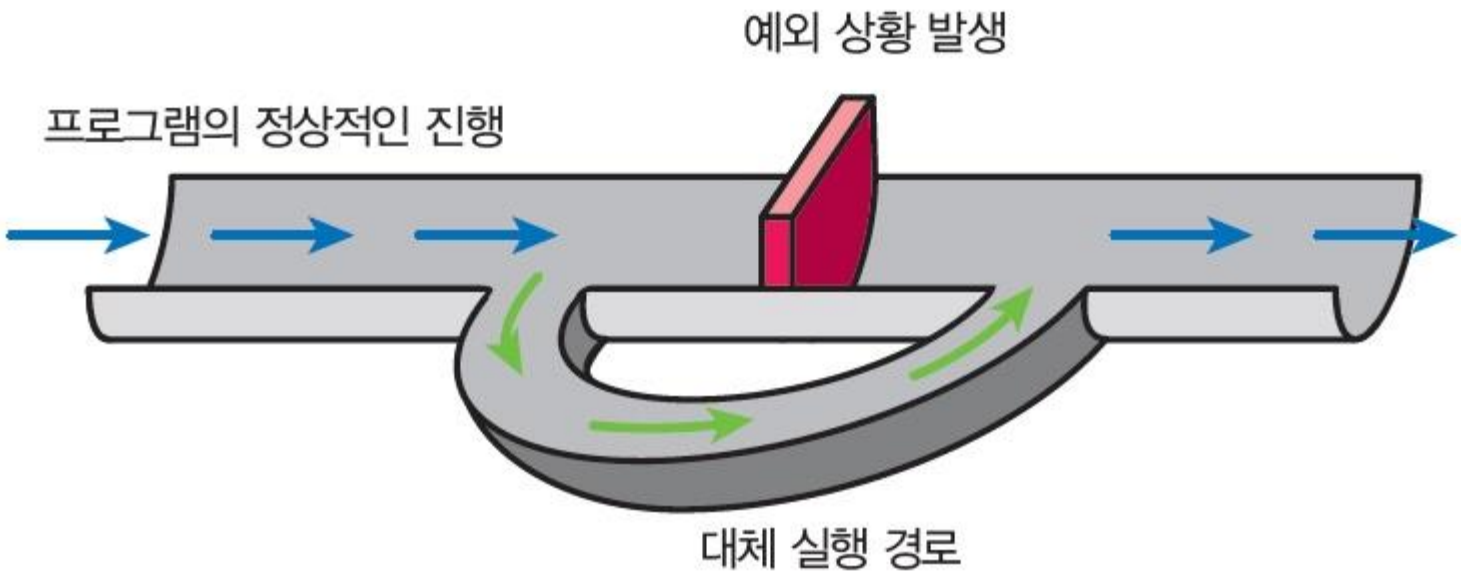
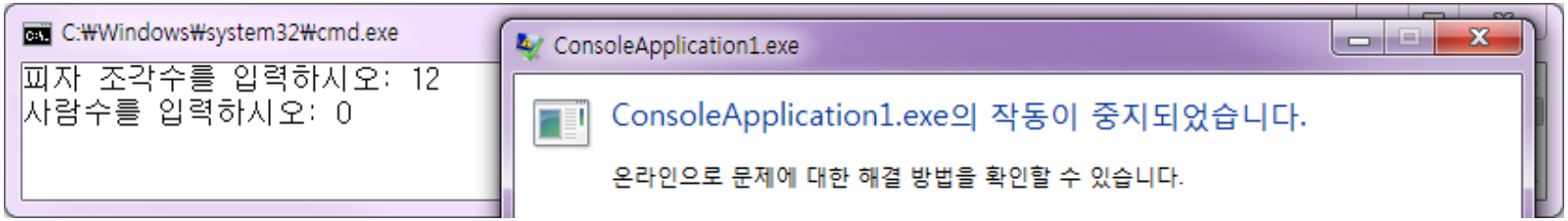


그림 14.1 예외 처리의 개요

예제

```
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;
}
```



예외 처리 고려사항

- 예외 발생시에 예외 처리기에 어떻게 바인딩되는가?
- 예외 정보를 어떻게 예외 처리기에 전달하는가?
- 예외가 발생한 곳에 예외처리기가 없을 경우에는?
- 예외처리기가 실행된 후에 프로그램이 계속 실행될 것인가? (연속성: continuation)
  - 프로그램이 단순히 종료되는가?
  - 실행이 계속된다면 어디에서?
    - 예외가 발생한 문장에서
    - 예외가 발생한 문장의 다음 문장에서
    - 다른 프로그램 단위에서

예외 처리기

- C++ 예외 처리 형식
  - try 블록: 예외 발생 가능 코드 블럭
  - throw 문: try 블록 내에서 예외 발생을 알리는 문장
  - catch 블록: 예외를 처리하는 블럭

문법 13.1 예외 처리

```
try {
    // 예외가 발생할 수 있는 코드
    if(예외가 발생하면)
        throw exception;
} catch (예외타입    매개변수) {
    // 예외를  처리하는  코드
}
```

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

try/catch 블록에서의 실행 흐름

- 예외가 처리된 후에는 제어가 try-catch 블록 다음에 위치한 첫 문장으로 이동한다.

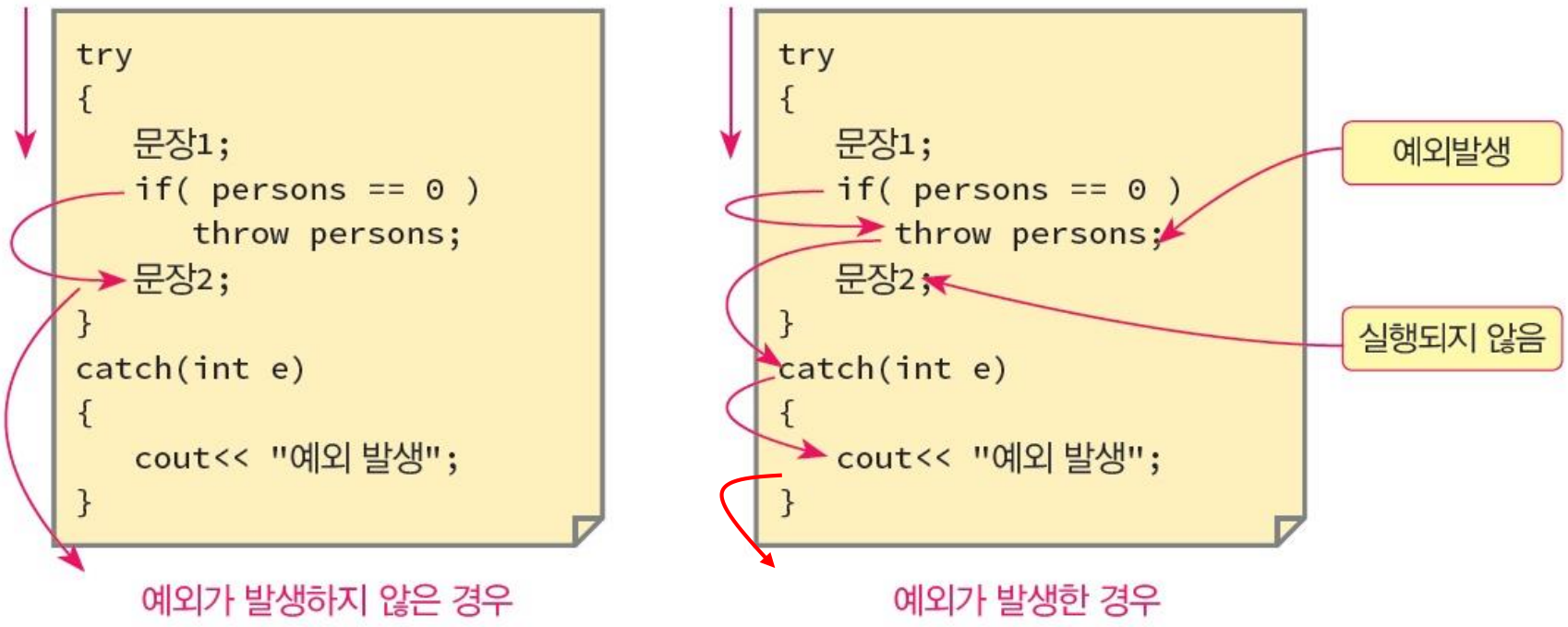


그림 14.3 try/catch 블록에서의 실행 흐름

try-throw-catch의 예외 처리 과정

n 5
sum 15

```
try {
   if(n == 0)
      throw n;
   else {
      average = sum / n;
      .....
      .....
   }
}
catch(int x) {
   cout << "예외 발생!! ";
   cout << x << "으로 나눌 수 없음"
<< endl;
   average = 0;
}
cout << "평균 = " << average <<
endl;
```

(a) 예외가 발생하지 않은 경우

n 0
sum 15
x 0

오류 탐지 코드

예외 발생. n을 x에 전달

예외 처리 코드

```
try {
   if(n == 0)
      throw n;
   else {
      average = sum / n;
      .....
      .....
   }
}
catch(int x) {
   cout << "예외 발생!! ";
   cout << x << "으로 나눌 수 없음" << endl;
   average = 0;
}
cout << "평균 = " << average << endl;
```

(b) 예외가 발생한 경우

catch 블록의 매개 변수

- throw 문장에서 던진 값이 catch의 매개변수로 전달
- catch의 매개변수는 단지 한 개만 가능하며, 예외정보 전달 및 예외처리기 매칭에 사용

예제

```
int main() {
   int n, sum, average;

   while(true) {
      cout << "합을 입력하세요>>";
      cin >> sum;
      cout << "인원수를 입력하세요>>";
      cin >> n;
      try {
         if(n <= 0) // 오류 탐지
         throw n; // 예외 발생.
      else
         average = sum / n;
      }
      catch(int x) {
         cout << "예외 발생!! " << x << "으로 나눌 수 없음" << endl;
         average = 0;
         cout << endl;
         continue; // 이 문장이 생략될 경우에는?
      }
      cout << "평균 = " << average << endl << endl;
   }
}
```

```
합을 입력하세요>>15
인원수를 입력하세요>>5

합을 입력하세요>>12
인원수를 입력하세요>>-3

합을 입력하세요>>25
인원수를 입력하세요>>0
```

다중 catch 문장

- 하나의 try 블록에서 여러 개의 throw 문 존재 가능
- 각 throw 문이 다른 타입의 값을 던질 수 있고,
- 이러한 다양한 타입의 오류 값을 처리하기 위해서 여러 개의 catch 블록 존재 가능 => 이를 다중 catch 문이라 함

try-다중 catch 블록

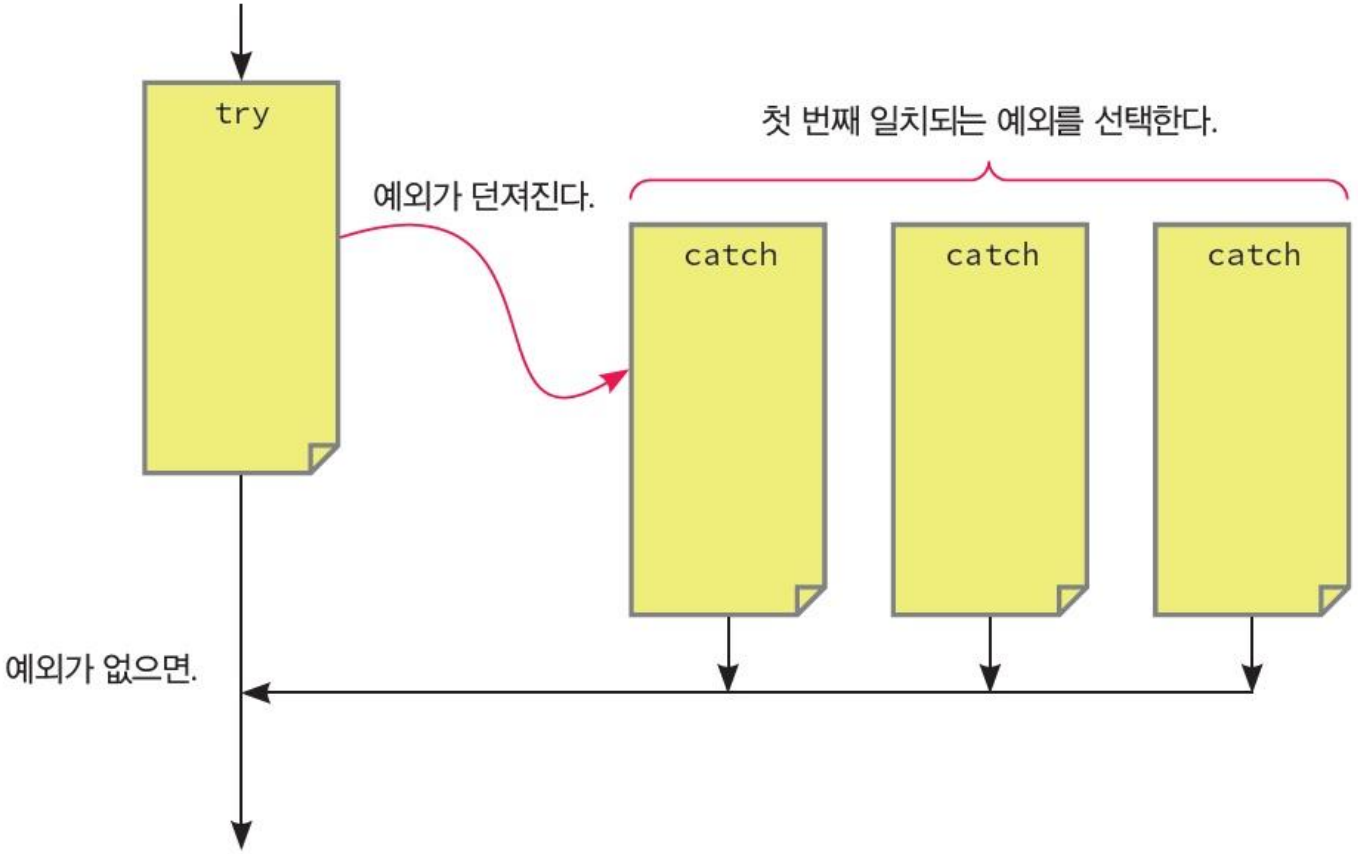


그림 14.2 try블록은 예외가 발생할 수 있는 위험한 코드이다. catch 블록은 예외를 처리하는 코드이다.

다중 catch 문장

- 예외가 발생되면 try 코드의 실행이 끝나고, 그 아래 위치한 예외 처리기와 바인딩하기 위해서 trow의 식과 catch의 형식 매개변수를 매칭한다.
- 매칭은 위에서부터 순서대로 catch의 형식 매개변수에 대해서 이루어진다.
- 정확히 매칭되는 처리기와 바인딩되기 위해서 특정 예외 처리기는 앞부분에, 더 포괄적인 예외 처리기는 그 다음에, 마지막에는 모든 예외를 처리할 수 있는 처리기를 배치한다.

```
try {
  -- 예외가 발생될 수 있는 코드
}
catch (형식 매개변수) {
  -- 특정 예외를 처리 코드
}
...
catch (형식 매개변수) {
  -- 더 포괄적인 예외 처리

catch(...)
  -- 모든 예외를 처리
}
```

예제: 다중 catch 문장

```
…
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) {
        cout << "오류: 사람수가 " << e << "입니다" << endl;
}
catch (int e) {
        cout << "오류: 사람이 " << e << " 명입니다." << endl;
}
```

예제: 다중 catch 문장

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

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

예제

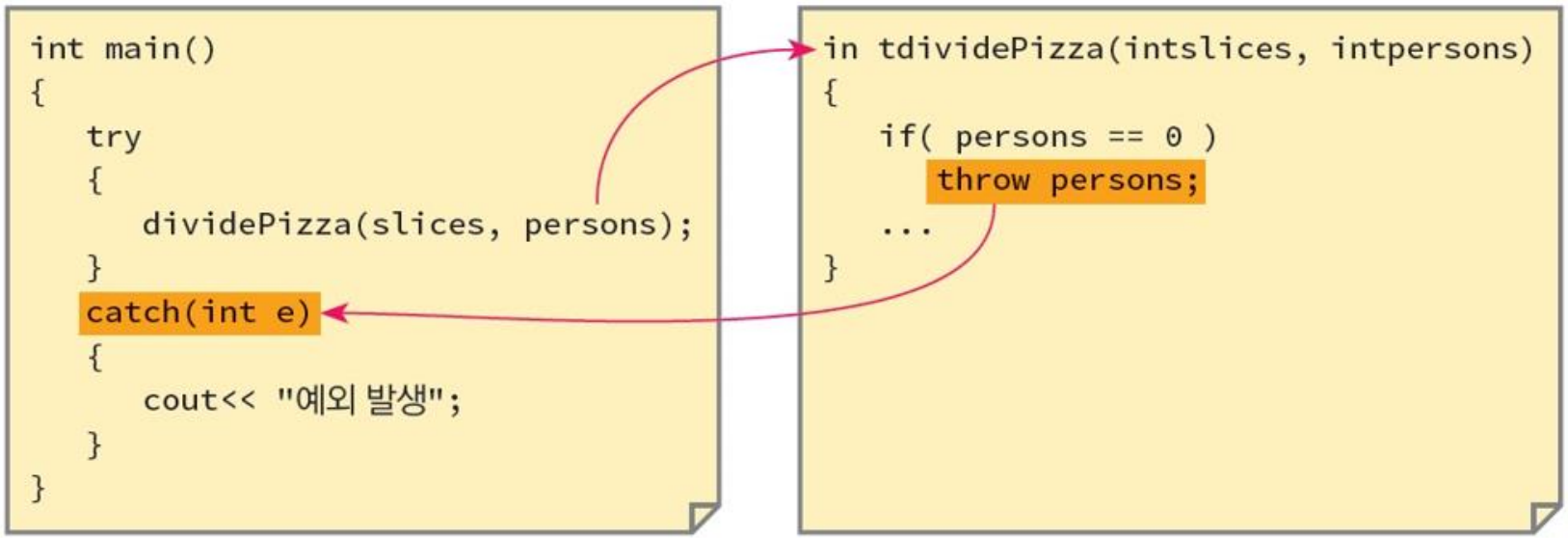
```
int person;
float num;
char any = ‘a’;

try {
        cout << "input a number of persons:";
        cin >> person;
        cout << "input a number: ";
        cin >> num;
        if (person == 0)
                throw person;
        if (num == 0)
                throw num;
        if (num <0)
                throw any;
}
catch (float e) {
        cout << "error:person " << person << endl;
}
catch (int e) {
        cout << "error:number " << num << endl;
}
catch (...) {
        cout << "you entered " << person << endl;
}
```

다음 입력에 대한 출력은?

person	num
0	3
3	0
3	-5
0	-5
0	0

예외 전달(전파: propagation)

- 호출한 함수에서 발생한 예외가 처리되지 않으면 호출자에게 전파 가능
- 던져진 예외는 처리될 때까지 함수 호출 체인을 거슬러가면서 예외의 타입과 같은 예외 처리기에 전달



```
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;
        }
        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;
}
```

예제: 예외 전파

```
void f() {
  int person;
  float num;
  char any = 'a';

  try {
    cout << "input a number of persons:";
    cin >> person;
    cout << "input a number: ";
    cin >> num;
    if (person == 0)
            throw person;
    if (num == 0)
            throw num;
    if (num < 0)
            throw any;
  }
  catch (int e) {
    cout << "error:person " << e << endl;
  }
}
```

```
int main() {

  try {
            f();
  }
  catch (float e) {
            cout << "error:number " << e << endl;
  }
  catch (...) {
            cout << "you entered a negative number"
                     << endl;
  }
  return 0;
}
```

다음 입력에 대한 출력은?

person	num
0	3
3	0
3	-5
0	-5
0	0

함수에서 발생가능한 예외 선언

- 함수 내부에서 예외가 발생될 수 있으나 그곳에서 처리하지 않을 경우에 다른 곳으로 전파할 수 있음을 알려주는 것이 좋다.
  - 이러한 명세는 선택적(이 경우에 모든 예외가 명세되는 효과)
  - 사용자에게 함수에서 어떠한 예외가 전파될 수 있는지를 알려줌

```
int max(int x, int y) throw(int) {
   if(x < 0) throw x;
   else if(y < 0) throw y;
   else if(x > y) return x;
   else return y;
}
```

```
double valueAt(double *p, int index) throw(int, char*) {
   if(index < 0)
      throw "index out of bounds exception";
   else if(p == NULL)
      throw 0;
   else
      return p[index];
}
```

예제

```
void f() throw(float, char) {
  int person;
  float num;
  char any = 'a';

  try {
    cout << "input a number of persons:";
    cin >> person;
    cout << "input a number: ";
    cin >> num;
    if (person == 0)
            throw person;
    if (num == 0)
            throw num;
    if (num < 0)
            throw any;
  }
  catch (int e) {
    cout << "error:person " << e << endl;
  }
  cout << “The end of function << endl;
}
```

```
int main() {

  try {
            f();
  }
  catch (float e) {
            cout << "error:number " << e << endl;
  }
  catch (...) {
            cout << "you entered a negative number"
                      << endl;
  }
  cout <<“The end of program”<<endl;
  return 0;
}
```

다음 입력에 대한 출력은?

person	num
0	3
3	0
3	-5
0	-5
0	0

예외의 재발생

- 예외 처리기가 예외를 다시 발생시킬 수 있다.

형식:

```
throw;
```

식이 없음에 유의
(앞서 발생한 예외 식 타입과 일치함)

```
void f() throw(int, float, char) {
  int person;
  float num;
  char any = 'a';

  try {
    cout << "input a number of persons:";
    cin >> person;
    cout << "input a number: ";
    cin >> num;
    if (person == 0)
          throw person;
    if (num == 0)
          throw num;
    if (num < 0)
          throw any;
  }
  catch (int e) {
    cout << "error:person "<< e << endl;
    throw;
  }
  cout << “The end of function” << endl;
}
```

```
int main() {

  try {
          f();
  }
  catch (float e) {
     cout << "error:number " << e << endl;
  }
  catch (...) {
     cout << "you entered a negative number" <<
          endl;
  }
  cout <<“The end of program”<<endl;
  return 0;
}
```

* 다음 입력에 대한 출력은?

person	num
0	0
2	0
2	-1

예외 클래스 작성

- 사용자는 자신 만의 예외 정보를 포함하는 예외 클래스 작성 가능
- throw에서 객체를 예외 값으로 전달

예제: 예외 클래스 작성

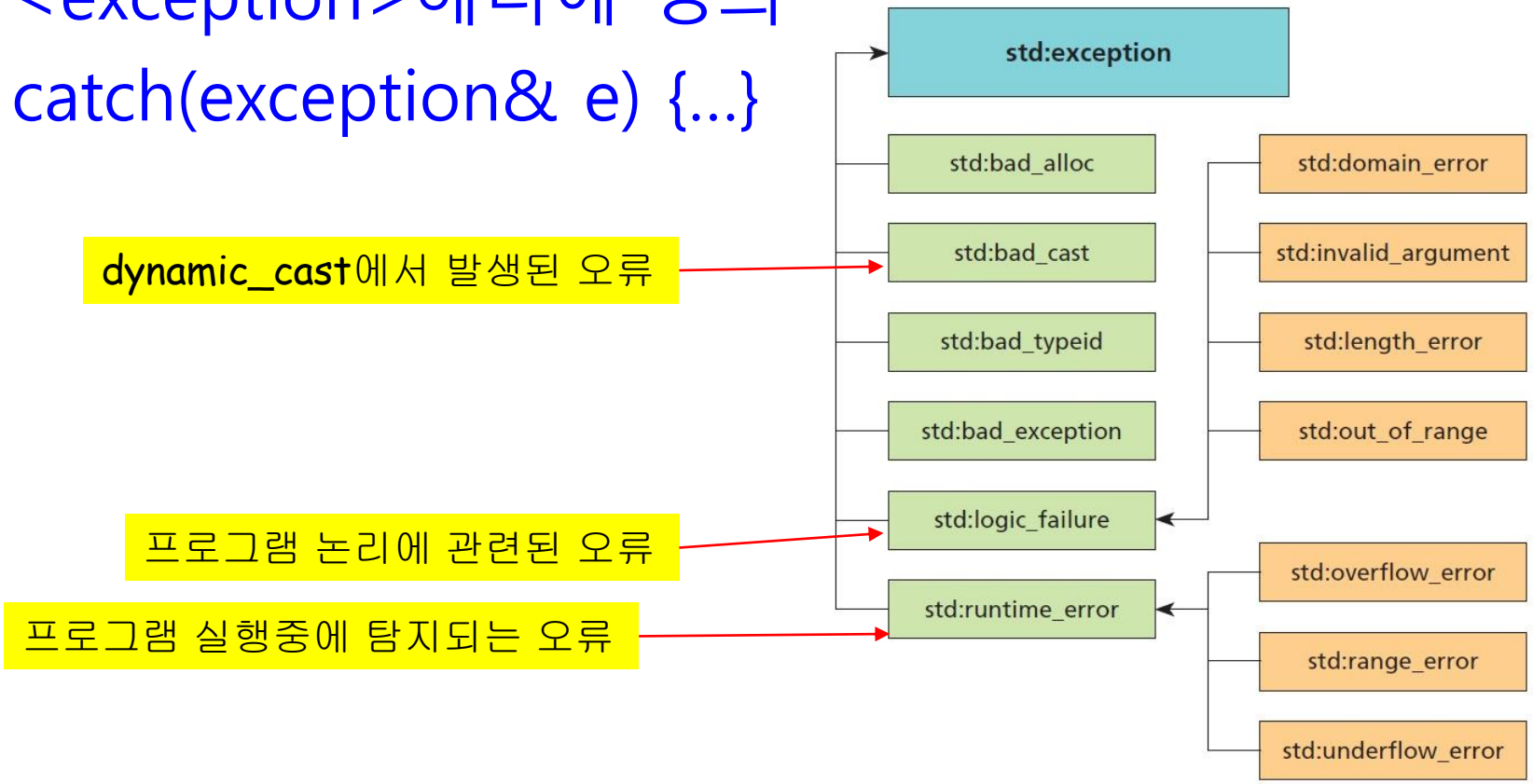
```
class MyException { // 사용자가 만드는 기본 예외 클래스 선언
  int lineNo;
  string func, msg;
public:
  MyException(int n, string f, string m) { // 라인 번호, 함수, 오류 메시지 전달
    lineNo = n; func = f; msg = m;
  }
  void print() { cout << func << ":" << lineNo << " ," << msg << endl; }
};

class DivideByZeroException : public MyException { // 0으로 나누는 예외 클래스
public:
  DivideByZeroException(int lineNo, string func, string msg)
    : MyException(lineNo, func, msg) { }
};
class InvalidInputException : public MyException { // 잘못된 입력 예외 클래스
public:
  InvalidInputException(int lineNo, string func, string msg)
    : MyException(lineNo, func, msg) { }
};
```

예제 (2)

```
int main() {
  int x, y;
  try {
    cout << "나눗셈을 합니다. 두 개의 양의 정수를 입력하세요>>";
    cin >> x >> y;
    if(x < 0 || y < 0)
      throw InvalidInputException(32, "main()", "음수 입력 예외 발생");
    if(y == 0)
      throw DivideByZeroException(34, "main()", "0으로 나누는 예외 발생");
    cout << (double)x / (double)y;
  }
  catch(DivideByZeroException &e) {
    e.print();
  }
  catch(InvalidInputException &e) {
    e.print();
  }
}
```

표준 예외

- C++ 표준 라이브러리에서 예외가 발생하면 std::exception이라는 특별한 예외가 발생
- <exception>헤더에 정의
- catch(exception& e) {...}



예제

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

int main() {
	try {
		int* p = new int[100000];
		delete p;
	}
	catch (exception& e) { // new에 의한 메모리 할당시 오류 처리
		cout << "표준 예외가 발생했습니다. " << e.what() << endl;
	}
	return 0;
}
```