



Chapter 14. 예외처리

목차

1. 예외처리

학습목표

- 예외에 대한 개념을 알아보고 이에 대한 처리 방법을 학습한다.

01 예외처리

■ 예외처리의 이해

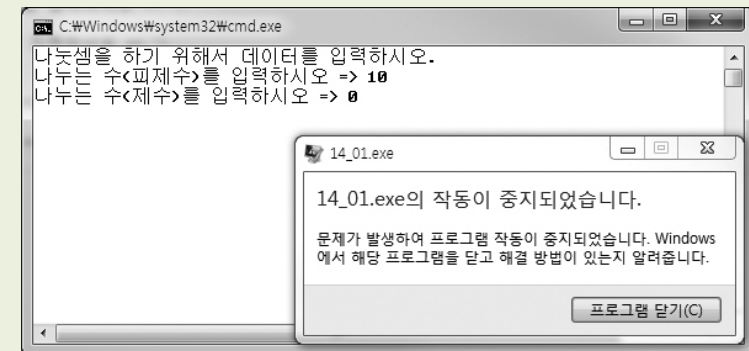
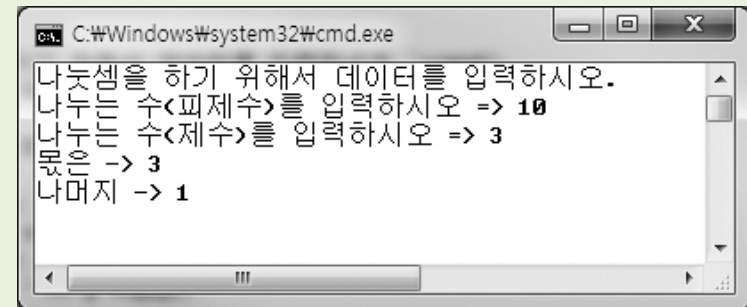
- 예외란 예상하지 못한 일 즉, 프로그램이 실행되는 동안에 발생하는 예기치 않은 에러를 의미한다. 프로그램 진행 중에 예외가 발생하면 그 시점에 프로그램이 바로 종료된다 (예: 0으로 나누는 경우)
- 프로그램을 사용하는 사용자 입장을 고려해서 C++ 프로그래머들은 예상치 못한 예외가 발생할 경우 어떻게 대처할지에 대한 프로그램을 미리 작성해둬야 하는데, 이를 예외처리라 한다.

■ 예외처리 구문

- C++에서는 예외처리가 이루어지는 부분만을 특별히 구분해서 프로그래머가 프로그램을 유지 보수하기 편리할 수 있도록 예외처리 구문이 따로 제공된다.

예제 14-1. 예외가 발생할 여지가 있는 프로그램 작성하기(14_01.cpp)

```
01 #include <iostream>
02 using namespace std;
03 void main()
04 {
05     int a, b, c, d;
06
07     cout<<"나눗셈을 하기 위해서 데이터를 입력하십시오."<<endl;
08     cout<<"나누는 수(피제수)를 입력하십시오 => ";
09     cin>>a;
10     cout<<"나누는 수(제수)를 입력하십시오 => ";
11     cin>>b;
12     c = a / b; // ❶
13     cout<<"몫은 -> " << c <<endl;
14     d = a % b; // ❷
15     cout<<"나머지 -> " << d <<endl;
16 }
```



01 예외처리

■ try, catch, throw

- C++에서는 예외를 검사하고 처리하는 데 사용하는 구문으로 **try~catch**를 제공한다. 예외가 발생할 수 있는 부분만을 특별히 구분해서 try 구문에 기술하고, 발생한 예외를 처리하는 부분은 catch 구문에 기술함.

```
try{  
    예외가 발생할 만한 코드  
} catch(해당_Exception e) {  
    예외처리를 위한 루틴  
}
```

try~catch문 기본 형식

- throw문은 프로그래머가 의도적으로 예외를 발생시키고자 할 때 사용한다.

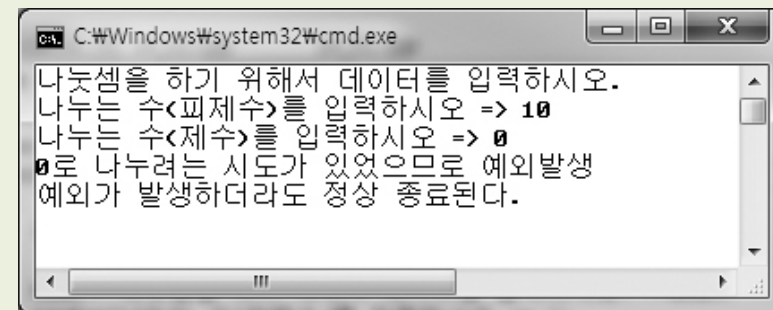
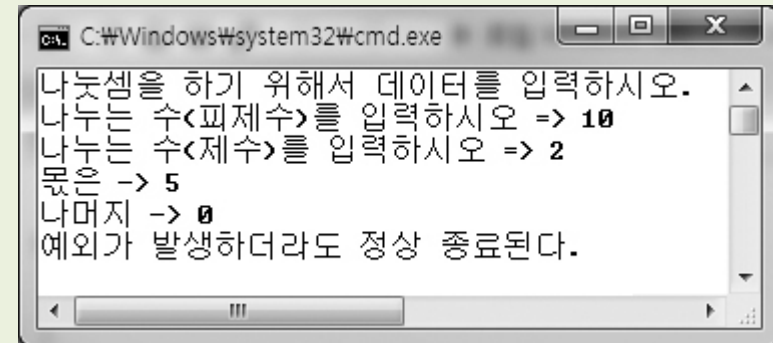
```
throw 예외객체;
```

throw문 기본 형식

- 호출한 함수의 예외 전달
 - C++에서 예외를 처리하는 방법은 크게 2가지다.
 - ❶ 예외가 발생한 함수 내에서 try문으로 직접 처리하는 방법
 - ❷ 예외가 발생한 함수를 호출한 함수로 예외를 전달하는 방법

예제 14-3. try~catch문으로 예외 처리하기(14_03.cpp)

```
01 #include <iostream>
02 using namespace std;
03 void main()
04 {
05     int a, b, c, d;
06
07     cout<<"나눗셈을 하기 위해서 데이터를 입력하시오."<<endl;
08     cout<<"나누는 수(피제수)를 입력하시오 => ";
09     cin>>a;
10     cout<<"나누는 수(제수)를 입력하시오 => ";
11     cin>>b;
12     try {
13         if(b==0){
14             throw b;
15         }
16         c = a / b;
17         cout<<"몫은 -> " << c <<endl;
18         d = a % b;
19         cout<<"나머지 -> " << d <<endl;
20     }
21     catch(int ex) {
22         cout<< ex <<"로 나누려는 시도가 있었으므로 예외발생"<<endl;
23     }
24     cout<<"예외가 발생하더라도 정상 종료된다."<<endl;
25 }
```



예제 14-5. 호출한 함수에 예외 전달하기(14_05.cpp)

```
01 #include <iostream>
02 using namespace std;
03
04 void divide(int a, int b)
05 {
06     int c, d;
07
08     cout<<" *** divide 함수 *** "<<endl;
09
10     if(b==0)
11         throw b;
12     c = a / b;
13     cout<<"몫은 -> " << c <<endl;
14     d = a % b;
15     cout<<"나머지 -> " << d <<endl;
16     cout<<endl;
17 }
18
19 void main()
20 {
21     try {
22         divide(10, 2);
23         divide(10, 0);
```

```
24         divide(10, 4);
25     }
26     catch(int ex) {
27         cout<< ex <<"로 나누려는 시도가 있었으므로 예외발생"<<endl;
28     }
29     cout<<"\n예외가 발생하더라도 정상 종료된다."<<endl;
30 }
```

```
C:\Windows\system32\cmd.exe
*** divide 함수 ***
몫은 -> 5
나머지 -> 0

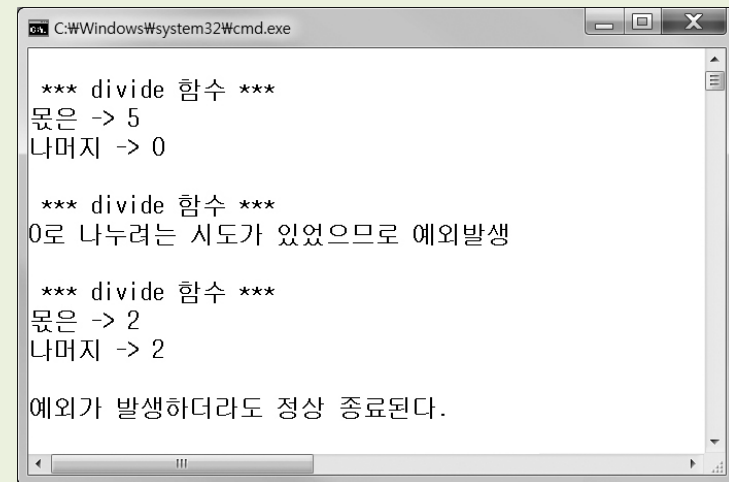
*** divide 함수 ***
0로 나누려는 시도가 있었으므로 예외발생

예외가 발생하더라도 정상 종료된다.
```


예제 14-6. 예외를 발생시킨 함수에서 예외처리하기(14_06.cpp)

```
01 #include <iostream>
02 using namespace std;
03
04 void divide(int a, int b)
05 {
06     int c, d;
07
08     cout<<"\n *** divide 함수 *** "<<endl;
09
10     try {
11         if(b==0)
12             throw b;
13         c = a / b;
14         cout<<"몫은 -> " << c <<endl;
15         d = a % b;
16         cout<<"나머지 -> " << d <<endl;
17     }
18     catch(int ex) {
19         cout<< ex <<"로 나누려는 시도가 있었으므로 예외발생"<<endl;
20     }
21 }
22
23 void main()
```

```
24 {
25     divide(10, 2);
26     divide(10, 0);
27     divide(10, 4);
28     cout<<"\n예외가 발생하더라도 정상 종료된다."<<endl;
29 }
```



```
C:\Windows\system32\cmd.exe

*** divide 함수 ***
몫은 -> 5
나머지 -> 0

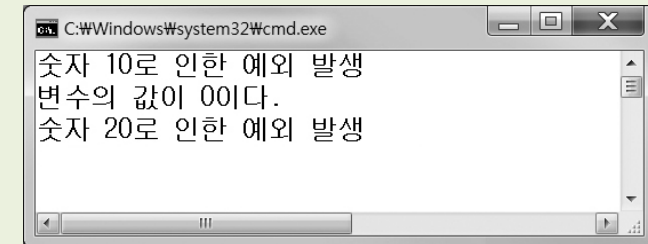
*** divide 함수 ***
0로 나누려는 시도가 있었으므로 예외발생

*** divide 함수 ***
몫은 -> 2
나머지 -> 2

예외가 발생하더라도 정상 종료된다.
```

예제 14-7. try문 하나와 catch 블록 여러 개 살펴보기(14_07.cpp)

```
01 #include <iostream>
02 using namespace std;
03
04 void func(int a)
05 {
06     try {
07         if(a==0)
08             throw "변수의 값이 0이다. ";
09         else
10             throw a;
11     }
12     catch(char *str) {
13         cout<<str<<endl;
14     }
15     catch(int ex) {
16         cout<<"숫자 "<< ex <<"로 인한 예외발생"<<endl;
17     }
18 }
19
20 void main()
21 {
22     func(10);
23     func(0);
24     func(20);
25 }
```



01 예외처리

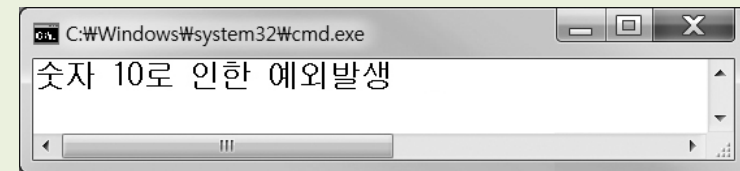
■ 함수에 예외 발생 명시하기

- 예외가 발생할 가능성이 있는 함수에서 예외를 직접 처리하지 않고,
호출한 함수로 옮길 수 있는 예외 자료형을 함수 정의를 할 때 명시적으로 나타낼 수 있다.
- 0으로 나눌 경우 불능으로 인해 문제가 발생하므로 throw문을 사용해서 프로그래머가 예외를 던져 준다.
그리고, 함수 머리 부분에 throw문으로 던지고자 하는 예외를 명시하고 던져질 예외 자료형을 제한함.

```
void func(int a) throw(char *, int)
{
    if(a==0)
        throw "변수의 값이 0이다. ";
    else
        throw a;
}
```

예제 14-8. 함수에서 발생하는 예외 자료형 명시하기(14_08.cpp)

```
01 #include <iostream>
02 using namespace std;
03
04 void func(int a) throw(char *, int)
05 {
06     if(a==0)
07         throw "변수의 값이 0 이다. ";
08     else
09         throw a;
10 }
11
12 void main()
13 {
14     try {
15         func(10);
16     }
17     catch(char *str) {
18         cout<<str<<endl;
19     }
20     catch(int ex) {
21         cout<<"숫자 "<< ex <<"로 인한 예외발생"<<endl;
22     }
23 }
```



01 예외처리

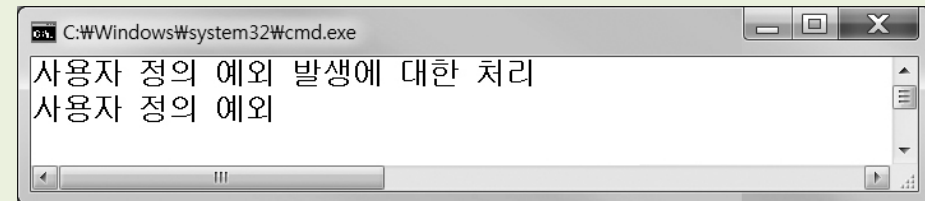
■ 사용자 정의 예외

- C++에서는 예외처리를 위해 ❶처럼 사용자 정의 예외 클래스를 정의할 수 있다.
그리고, ❷처럼 throw문을 이용해서 필요에 따라 예외를 발생시킬 수 있다.

```
❶ class UserError {  
    void func();  
}  
  
❷ throw UserError();
```

예제 14-9. 사용자 정의 예외 클래스(UserError) 정의하기(14_09.cpp)

```
01 #include <iostream>
02 using namespace std;
03 // 사용자 정의 예외 클래스(UserError) 정의
04 class UserError
05 {
06 public :
07     void func();
08 };
09 void UserError::func()
10 {
11     cout<<"사용자 정의 예외"<<endl;
12 }
13 void main()
14 {
15     try {
16         throw UserError();      // 사용자 정의 예외 클래스의 생성자 호출
17     } catch(UserError &ex) {
18         cout<<"사용자 정의 예외발생에 대한 처리"<<endl;
19         ex.func();
20     }
21 }
```



예제 14-10. 예외처리하는 클래스 정의하기(14_10.cpp)

```
#include <iostream>
using namespace std;
class UserError {
public:
    UserError(char * message);
    UserError() { }
};

//전달인자를 문자열로 갖는 생성자
UserError::UserError(char * message)
{
    cout << message << endl;
}

class Human {
    char name[20];
    int age;
public:
    Human(char *name, int age);
    void setName(char *name);
    void setAge(int age);
    void prn();
};
```

```
Human::Human(char *name, int age)
{
    setName(name);
    setAge(age);
    prn();
}

void Human::setName(char *name)
{
    //이름이 2자 이하거나 8자 이상이면
    if (!(strlen(name) >= 2 && strlen(name) <= 8))
        //사용자가 Exception 예외를 발생시킴
        throw UserError("이름은 2자이상 8자이하로만 기술하세요.");
    strcpy_s(this->name, name);
}

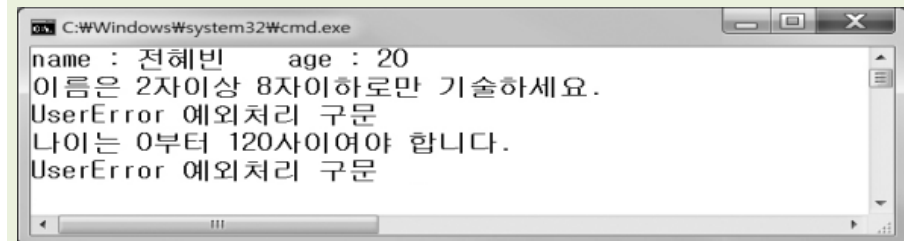
void Human::setAge(int age)
{
    //나이가 0부터 120사이가 아니면 UserError 예외 발생
    if (!(age >= 0 && age <= 120))
        throw UserError("나이는 0부터 120사이여야 합니다.");
    this->age = age;
}
```

예제 14-10. 예외처리하는 클래스 정의하기(14_10.cpp)

```
void Human::prn()
{
    cout << "name : " << name << " age : " << age << endl;
}
```

```
void main() {
    Human *p1, *p2, *p3;
    try {
        p1 = new Human("전혜빈", 20);
    }
    catch (UserError a) {
        cout << "UserError 예외처리 구문" << endl;
    }
    try {
        p2 = new Human("Angelina Jolie", 10);
    }
    catch (UserError a) {
        cout << "UserError 예외처리 구문" << endl;
    }
    try {
        p3 = new Human("고은아", 220);
    }
```

```
    catch (UserError a) {
        cout << "UserError 예외처리 구문" << endl;
    }
}
```



```
C:\Windows\system32\cmd.exe
name : 전혜빈      age : 20
이름은 2자이상 8자이하로만 기술하세요.
UserError 예외처리 구문
나이는 0부터 120사이여야 합니다.
UserError 예외처리 구문
```