

LAB #12



● 실습 목표 – 신뢰성 있는 프로그램 작성을 위한 예외처리 기법 습득

1. 전통적인 에러 처리 기법과 예외처리 장치의 이용
2. 예외처리 클래스의 작성법
3. 예외 명세를 사용한 예외처리 기법
4. 클래스에서의 예외처리 기법
5. 표준 예외들

● 실습 문제 1

목표 : 전통적인 예외 처리 기법을 알아보고 예외처리 장치를 이용한 기법을 살펴본다.

문제 : 전통적인 예외 처리 기법이 적용된 프로그램을 예외처리 장치를 이용한 프로그램으로 수정하시오.

```
try
{
    //예외가발생할지역
}
catch(처리 되어야 할 예외의 종류)
{
    //발생한예외에대한처리
}
```

● 실습 문제 1 – 전통적인 예외처리 기법

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

int main()
{
    cout << "Enter the dividend: " ;
    double dividend ;
    cin >> dividend ;
    cout << "Enter the divisor : " ;
    double divisor ;
    cin >> divisor ;

    if( divisor == 0 ) // 이 부분을 예외처리 장치를 사용하여 수정하십시오.
    {
        cout << "***Error 100: divisor 0\n" ;
        exit(100) ;
    }
    double quotient = dividend / divisor ;
    cout << "Quotient is : " << quotient << endl ;
    return 0 ;
}
```

● 실습 문제 2 < Section 15-1 >

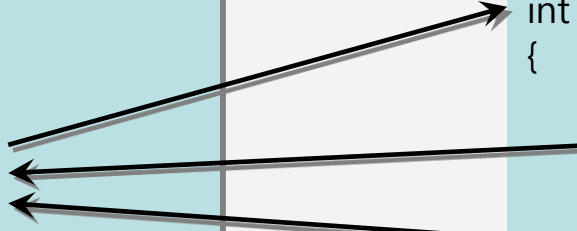
목표 : 예외의 전파에 대한 특징을 이해한다.

문제 : 다음 프로그램을 완성하고, 예외의 전파에 대한 부분을 이해한다.

분리된 함수에서 예외 throw

```
int callingFun(...)
{
    try {
        calledFun(...) ;
        ...
    } // try
    catch(object & error)
    {
        ...
    } // catch
    return(value) ;
} // callingFun
```

```
int calledFun(...)
{
    ...
    throw object ;
    ...
    return value ;
} // calledFun
```



● 실습 문제 2

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

void func(int a,int b)
{
    if( b==0)
        throw b;
    cout<<"a 나누기 b의 몫은"<<a/b<<"나머지는"<<a%b<<"입니다."<<endl;
}

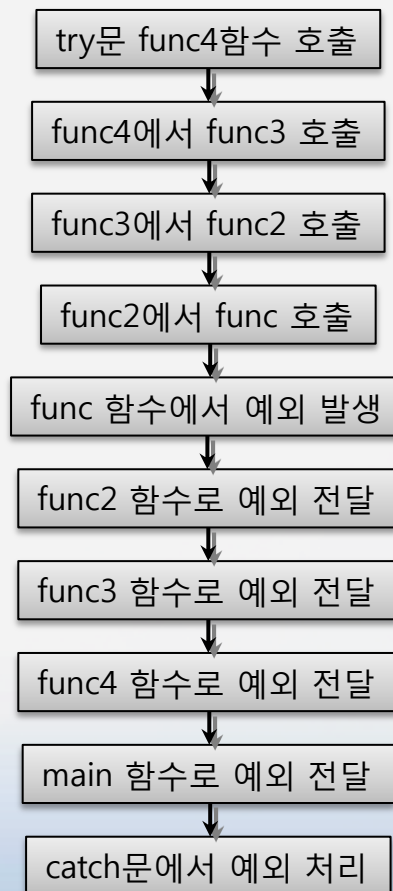
int main(void)
{
    int a,b;

    try
    {
        cout<<"두개의수를입력해주세요:";
        cin>>a>>b;
        func(a,b);
    }
    catch(int ex)
    {
        cout<<"b에"<<ex<<"이입력되었습니다. 다시실행하세요"<<endl;
    }
    return 0;
}
```

● 실습 문제 3

목표 : 예외의 재전달 과정을 이해한다.

다음 프로그램을 분석하여, 예외처리의 재전달 과정과 비교하여 이해한다.



• 실습 문제 3

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

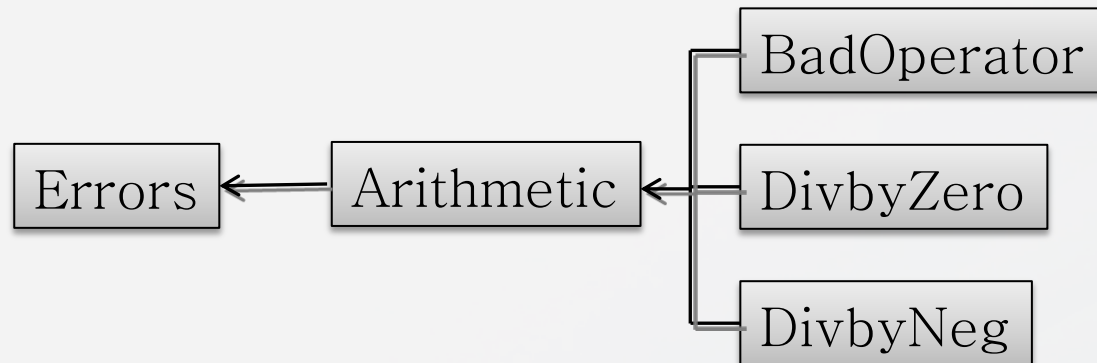
void func(){ throw 100; }
void func2(){ func(); }
void func3(){ func2(); }
void func4(){ func3(); }

int main(void)
{
    try
    {
        func4();
    }
    catch(int ex)
    {
        cout<<"예외처리입력:"<<ex<<"이 입력되었습니다."<<endl;
    }
    return 0;
}
```


● 실습 문제 4

목표 : 에러 클래스를 정의하고, 이를 이용하여 에러를 처리하는 방법에 대해 이해한다.

문제 : 주어진 프로그램 소스를 완성하여 에러클래스를 이용한 에러처리 사용법을 익히며, 이는 어떠한 장점이 있는지 설명하시오.



● 실습 문제 4 – Error.h

```
using namespace std ;

class Error
{
public:
    virtual void printMessage()
    {
        cout << "***Error : type Error\n" ;
    }
};

class Arithmetic: public Error
{
public:
    virtual void printMessage()
    {
        cout << "***Error : type Arithmetic\n" ;
    }
};

class DivbyZero : public Arithmetic
{
public:
    virtual void printMessage()
    {
        cout << "Error : 100 divisor 0\n" ;
    }
};
```

```
class DivbyNeg : public Arithmetic
{
public:
    virtual void printMessage()
    {
        cout << "***Error : 101 negative divisor\n" ;
    }
};

class BadOperator : public Arithmetic
{
public:
    virtual void printMessage()
    {
        cout << "***Error : 102 invalid operator\n" ;
    }
};
```

● 실습 문제 4 – math.cpp

```
double math( char oper, double data1, double data2 )
{
    double result ;
    switch(oper)
    {
        case '+' :
            result = data1 + data2 ;
            break ;

        case '-' :
            result = data1 - data2 ;
            break ;

        case '*' :
            result = data1 * data2 ;
            break ;

        case '/' :
            if( data2 == 0 )
                throw DivbyZero() ;
            if( data2 < 0 )
                throw DivbyNeg() ;
            result = data1 / data2 ;
            break ;

        default:
            throw BadOperator() ;
            break ;
    }
    return result ;
}
```

● 실습 문제 4 – Lab04.cpp

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

#include "Error.h"
#include "math.h"
#define FLUSH while( cin.get() != '\n')

int main()
{
    cout << "Begin Error class demonstration\n" ;

    cout << "Enter the first data : " ;
    double data1 ;
    cin >> data1 ;
    cout << "Enter the second data : " ;
    double data2 ;
    cin >> data2 ;
    cout << "Enter the operator : " ;
    char oper ;
    cin >> oper ;
    FLUSH ;
```

```
try
{
    double result = math( oper, data1, data2 ) ;
    cout << "result : " << result << endl ;
}
catch( Error & error )
{
    error.printMessage() ;
    exit(100) ;
}
cout << "Normal end of demonstration\n" ;
return 0 ;
}
```

● 실습 문제 5

목표 : 표준 예외 클래스에 대하여 알아본다.

문제 : 표준 예외 클래스들과 이에 대한 사용법을 익히고, 프로그램을 실행시켜 결과를 확인한다.

예외	설명
논리 에러들 -domain_error -invalid_argument -length_error -out_of_range	프로그램 내부에서 일어나는 논리적인 에러들 -함수 인수를 잘못 기재 -C++ 표준 함수의 인수를 잘못 기재 -메모리를 차지하는 객체의 길이가 허용된 최대 길이를 초과함 -배열의 인덱스가 배열 요소의 범위를 초과함
bad_alloc	New 연산자가 메모리 할당을 할 수 없는 경우
bad_exception	발생한 예외가 어떤 catch 문과도 부합되지 않음
ios_base::failure	외부 파일의 처리시 발생한 에러
bad_typeid	Typeid에[서의 에러
bad_case	Dynamic cast의 실패
Run-time 에러 -range_error -overflow_error -underflow_error	프로그램 범위를 넘어선 에러 -표준 템플릿 라이브러리(STL) 컨테이너의 에러 -표준 템플릿 라이브러리(STL) 컨테이너의 오버플로우 에러 -표준 템플릿 라이브러리(STL) 컨테이너의 언더플로우 에러

● 실습 문제 5-1 – out_of_range

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

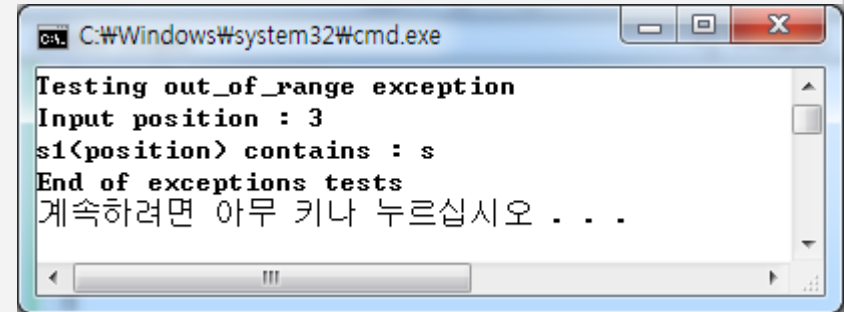
int main()
{
    string s1("This is the string");

    cout << "Testing out_of_range exception\n";
    int position;
    cout << "Input position : ";
    cin >> position;

    try
    {
        cout << "s1(position) contains : "
              << s1.at(position) << endl;
    }
    catch( exception & err )
    {
        cout << err.what() << endl;
    }
    cout << "End of exceptions tests\n";

    return 0;
}
```

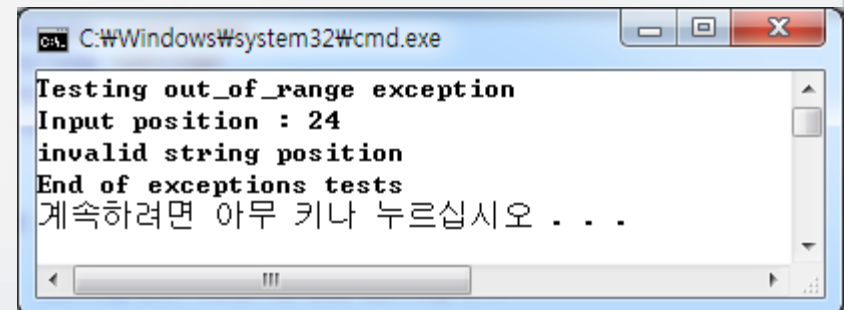
정상적인 프로그램 수행



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

Testing out_of_range exception
Input position : 3
s1(position) contains : s
End of exceptions tests
계속하려면 아무 키나 누르십시오 . . .
```

인덱스의 범위를 벗어났을 경우



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

Testing out_of_range exception
Input position : 24
invalid string position
End of exceptions tests
계속하려면 아무 키나 누르십시오 . . .
```

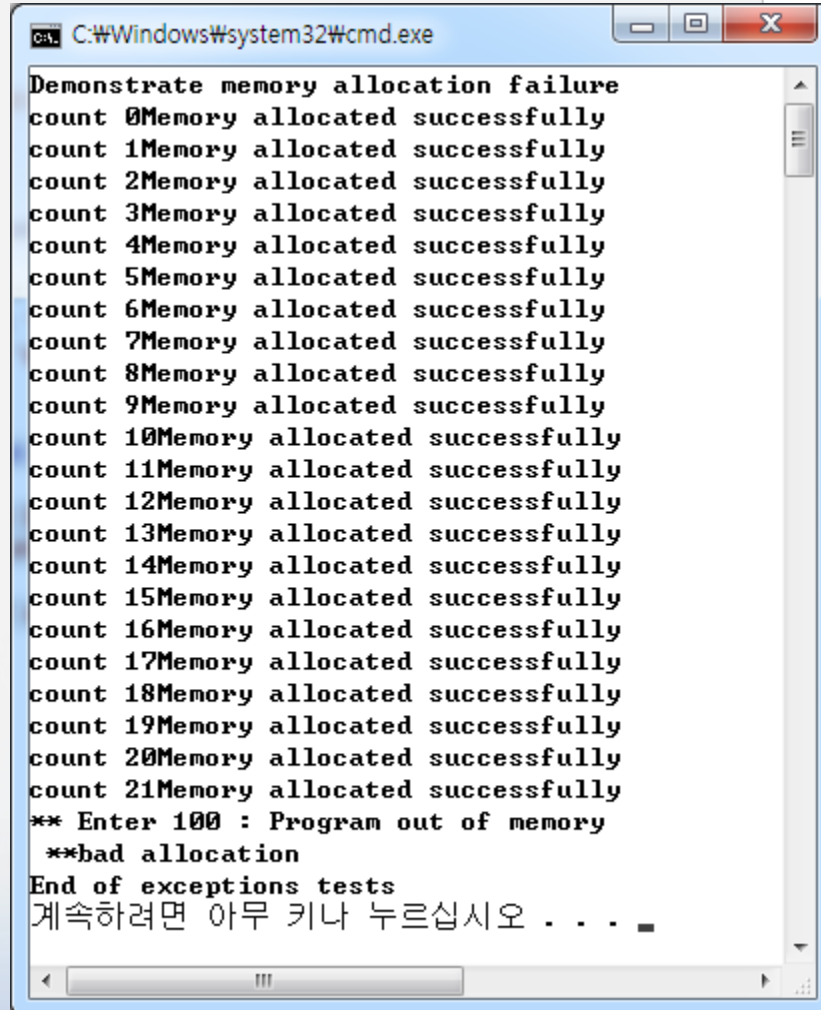
• 실습 문제 5-2 – bad_alloc

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

int main()
{
    cout << "Demonstrate memory allocation failure\n" ;

    try
    {
        for( int count = 0 ; ; count++ ) {
            double * Arr = new double[1024*1024*10] ;
            cout << "count " << count
                << "Memory allocated successfully\n" ;
        }
    }
    catch( exception & err )
    {
        cout << "*** Enter 100 : Program out of memory\n ***"
            << err.what() << endl ;
    }
    cout << "End of exceptions tests\n" ;

    return 0 ;
}
```



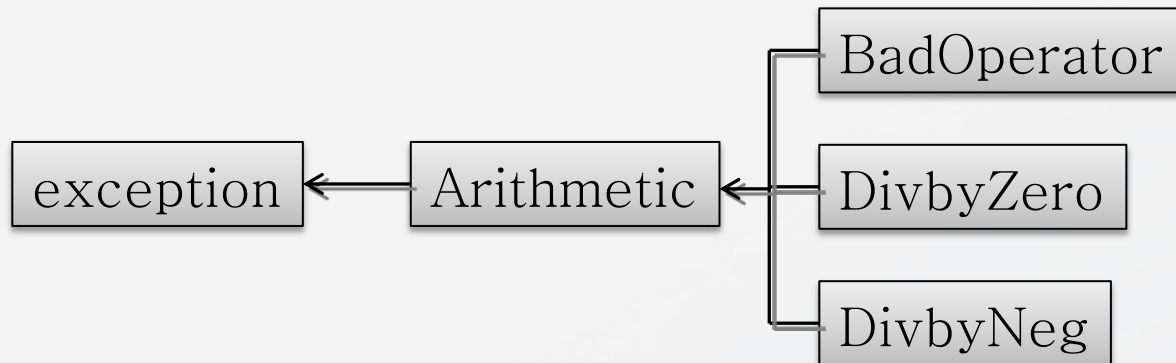
```
C:\Windows\system32\cmd.exe

Demonstrate memory allocation failure
count 0Memory allocated successfully
count 1Memory allocated successfully
count 2Memory allocated successfully
count 3Memory allocated successfully
count 4Memory allocated successfully
count 5Memory allocated successfully
count 6Memory allocated successfully
count 7Memory allocated successfully
count 8Memory allocated successfully
count 9Memory allocated successfully
count 10Memory allocated successfully
count 11Memory allocated successfully
count 12Memory allocated successfully
count 13Memory allocated successfully
count 14Memory allocated successfully
count 15Memory allocated successfully
count 16Memory allocated successfully
count 17Memory allocated successfully
count 18Memory allocated successfully
count 19Memory allocated successfully
count 20Memory allocated successfully
count 21Memory allocated successfully
*** Enter 100 : Program out of memory
***bad allocation
End of exceptions tests
계속하려면 아무 키나 누르십시오 . . .
```

● 실습 문제 6

목표 : 표준 예외 클래스에 새로운 클래스의 추가하는 방법에 대한 이해

문제 : 실습문제 4의 프로그램을 수정하여, 시스템 예외 클래스인 `exception` 클래스를 이용하여 프로그램 한 뒤 어떠한 차이점이 있는지 설명하시오.



● 실습 문제 6 – Error.h

```
#include <exception>
using namespace std ;
class Arithmetic: public exception
{
public:
    virtual const char * what() const throw()
    { return "***Error : type Arithmetic\n" ; }
};
class DivbyZero: public Arithmetic
{
public:
    virtual const char * what() const throw()
    { return "***Error : 100 Divisor 0\n" ; }
};
class DivbyNeg: public Arithmetic
{
public:
    // 실습 4의 내용과 DivbyZero 함수를 참조하여 what 함수
    // 를 작성하시오.
};
class BadOperator : public Arithmetic
{
public:
    // 실습 4의 내용과 DivbyZero 함수를 참조하여 what() 함수
    // 를 작성하시오.
};
```

● 실습 문제 6 – math.h

```
double math( char oper, double data1, double data2 )
{
    double result ;
    switch(oper)
    {
        case '+' :
            result = data1 + data2 ;
            break ;

        case '-' :
            result = data1 - data2 ;
            break ;

        case '*' :
            result = data1 * data2 ;
            break ;

        case '/' :
            if( data2 == 0 )
                throw DivbyZero() ;
            if( data2 < 0 )
                throw DivbyNeg() ;
            result = data1 / data2 ;
            break ;

        default:
            throw BadOperator() ;
            break ;
    }
    return result ;
}
```

● 실습 문제 6 – Lab06.cpp

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

#include "Error.h"
#include "math.h"
#define FLUSH while( cin.get() != '\n')

int main()
{
    cout << "Begin Error class
demonstration\n" ;

    cout << "Enter the first data : " ;
    double data1 ;
    cin >> data1 ;
    cout << "Enter the second data : " ;
    double data2 ;
    cin >> data2 ;
    cout << "Enter the operator : " ;
    char oper ;
    cin >> oper ;
    FLUSH ;
```

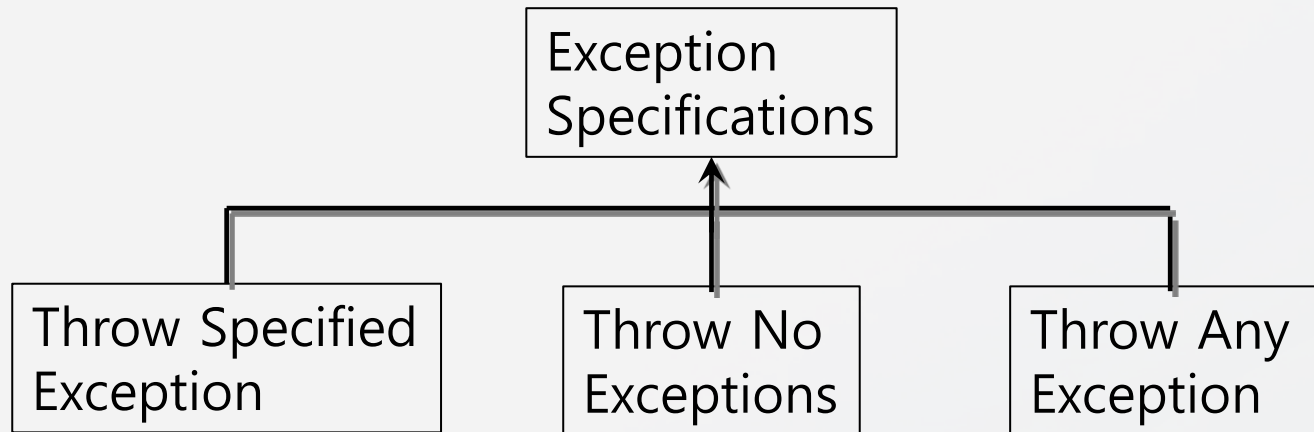
```
try
{
    double result = math( oper, data1, data2 ) ;
    cout << "result : " << result << endl ;
}
catch( exception & error )
{
    cout << error.what() ;
    exit(100) ;
}
cout << "Normal end of demonstration\n" ;
return 0 ;
}
```

● 실습 문제 7

목표 : 예외 명세에 대하여 알아보고, unexpected 시스템 함수에 대하여 이해한다.

문제 : 예외 명세중 unexpected 함수가 호출되었을 때, unexpected 함수의 기본값을 변경하여 예외처리 함수의 이름으로 정의하는 방법에 대하여 알아본다.

다음 프로그램의 소스를 실행하여, 출력결과를 살펴보고 출력 결과에 대한 이유를 설명하시오.



● 실습 문제 7

```
#include <iostream>
#include <exception>
using namespace std;
void myunexpected () {
    cerr << "myunexpected called\n";
    throw 0;    // throws int (in exception-specification)
}
void myunexpected2 () {
    cerr << "myunexpected2 called\n";
    exit(100); // exit the program
}
void myfunction () throw (int) {
    throw 'x'; // throws char (not in exception-specification)
}
int main (void) {
    // 주의: MSVC는 C++의 exception specification을 표준대로 구현하지 않아 textbook의 설명대로 동작하
    // 지 않음
    set_unexpected( myunexpected ); // &myunexpected can also be used instead of myunexpected
    try {
        myfunction();
    }
    catch (int) { cerr << "caught int\n"; }
    catch (...) { cerr << "caught other exception (non-compliant compiler?)\n"; }
    unexpected_handler oldHand = set_unexpected( myunexpected2 ); // &myunexpected can also be used
    cout << "oldHand: " << oldHand << " myunexpected: " << myunexpected << " &myunexpected: "
    << &myunexpected << endl;

    return 0;
}
```