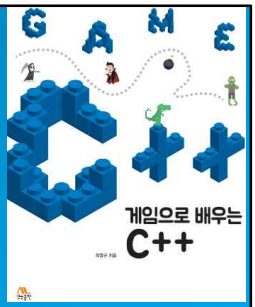
게임으로 배우는 C++



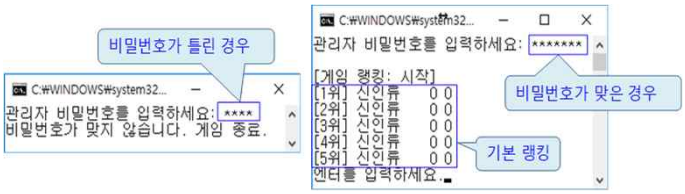
CHAPTER 12

예외처리와 형 변환

몬스터 월드7 (새로운 경쟁의 시작)

- Monster World 7
  - 몬스터 세상에 랭킹 추가
  - 랭킹 파일을 읽는데 파일이 존재하지 않는 경우

  → C++ 예외 처리 기법을 적용
    - 관리자가 비밀번호를 입력하는 방식



2

12장 학습 목표

- 예외 처리의 개념을 이해한다.
- 기존의 예외 처리 방법들을 이해한다.
- C++의 예외처리 방법을 이해한다.
- 예외의 전달을 이해하고 활용할 수 있는 능력을 기른다.
- const 지시자를 이해하고 활용할 수 있다.
- 네 가지 형 변환 방법을 이해한다.

3

12.1 예외 처리란?

- 예외 처리란?
- 일반적인 예외 처리 방법들

4

예외 처리란?

- **예외(exception)** : 심각하지 않은 오류
  - 예외 처리를 통해 프로그램 계속 진행 가능

일반적인 예외 처리 방법들

- 예: 유리수 클래스의 "나누기 0" 예외

```
class Rational {
    int top, bottom;      // 유리수의 분자, 분모(0이 아니어야 함)
public:
    double real() {return (double) top / bottom;}
    ...
};
```

- 예외가 발생하지 않을 것이라고 가정
- 예외 메시지 출력

```
if( bottom == 0)
    cout << "예외발생: 분모가 0이 되었습니다!" << endl;
```

- 단정(assertion) 검사와 실행 중지

```
double real() {
    assert( bottom != 0 );
    return (double)top / bottom;
}
```

일반적인 예외 처리 방법

- 특수한 값이나 예외 코드를 반환

```
double real() {
        if( bottom == 0)
                return 0.0;
        return (double)top / bottom;
}
```

- 오류 확인 변수(error flag)의 사용

```
bool bError = false;    // 전역 변수
. . .
        double real() {
                if( bottom == 0)
                        bError = true;
                return (double)top / bottom;
        }
```

- 예외 처리 함수 지정

12.2 C++의 예외 처리 방법

- C++ 예외 처리 키워드
- C++ 예외 처리
- 여러 개의 예외를 던지는 예

C++의 예외 처리 방법

- try, catch, throw라는 키워드를 이용
- 기본 형식

```
try {
        ...                 // 예외가 발생할 수 있는 코드
        if (예외_조건)
                throw 예외;
}
catch {
        ...                 // 예외를 처리하는 코드
}
```

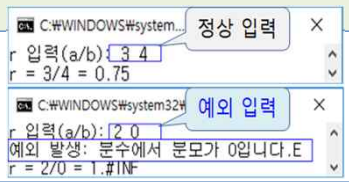
9

C++ 예외 처리

- try – throw – catch

```
double real() const {
    try {
        if (bottom == 0) throw('E');
    }
    catch (char c) { cout << "예외 발생: 분수에서 분모가 0입니다.\n"; }

    return (double)top / bottom;
}
```



- 모든 예외 처리

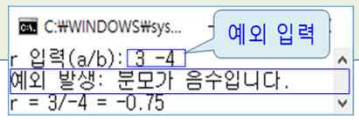
```
    catch (...) { cout << "예외 발생: 모든 예외를 처리합니다.\n"; }
```

10

여러 개의 예외를 던지는 예

- 분모가 음수인 경우 예외 발생 추가

```
class Rational {
    ...             // 프로그램 11.5의 Rational 클래스 모든 멤버 추가
    double real() {
        try {
                if (bottom == 0) throw('E');
                if (bottom < 0) throw(bottom);
        }
        catch (char c) { cout << "예외 발생: 분수에서 분모가 0입니다.\n"; }
        catch (int bot) { cout << "예외 발생: 분모가 음수입니다.\n"; }
        return (double)top / bottom;
    }
};
```



12.3 예외 클래스를 만들어 사용하기

- 예외 클래스 사용하기
- 언제 예외 처리 기법을 사용할까?

예외 클래스 사용하기

```
struct NegBotException {
    int top, bottom;
    NegBotException(int t = 1, int b = 0) : top(t), bottom(b) {}
};
...
class Rational {
    ...
    double real() {
        try{
            if (bottom < 0)
                throw( NegBotException(top, bottom) );
        }
        catch (NegBotException e) {
            cout << "예외 발생: 분모가 음수입니다. "
                 << -e.top<< '/' << -e.bottom<< "이 바람직합니다.\n";
        }
        return (double)top / bottom;
    }
};
```


언제 예외 처리 기법을 사용할까?

- 예외 상황을 그 함수 안에서 처리할 수 있다면?
  - 예외 처리 기법을 사용하지 않는 것이 좋음.
  - 그 함수 안에서 직접 처리.

```
double real() {
        if (bottom == 0)
                cout << "예외 발생: 분수에서 분모가 0입니다.\n";
        else if (bottom < 0)
                cout << "예외 발생: 분모가 음수입니다.\n";
        return (double)top / bottom;
}
```

- 함수의 호출자마다 예외를 처리하는 방법이 다르다면?
  - 반드시 예외 처리를 사용하는 것이 좋음
  - 이때 예외의 전달을 사용
  - **프로그램 코드와 오류 처리 코드를 분리**

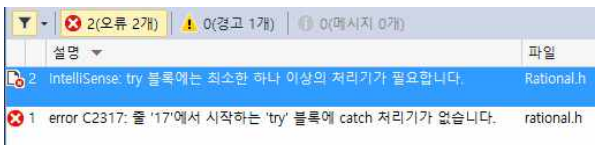
12.4 예외의 전달

- 예외의 전달
- 예외 전달의 예

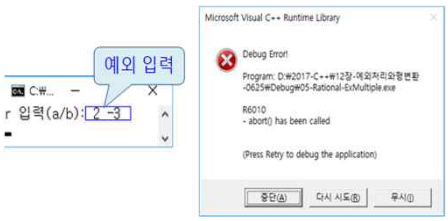
15

예외의 전달

- 만약 예외를 던지기만 하고 처리하지 않으면?
  - try 블록 다음에는 최소한 하나 이상의 catch 블록이 있어야 함
  - 없으면 오류



- 예외를 전달하고 마지막까지 처리하지 않으면
  - 실행 오류



16

예외 전달의 예

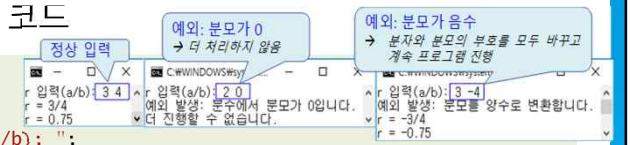
- 예외를 전달하는 코드

```
struct NegBotException {...}

class Rational
{
        ...
        double real() {
                if (bottom == 0) throw('E');
                return (double)top / bottom;
        }
        friend ostream& operator<<(ostream& os, const Rational& f) {...}
        friend istream& operator>>(istream& is, Rational& f) {
                is >> f.top >> f.bottom;          // 분자 / 분모 읽기
                if (f.bottom == 0) throw('E');
                if (f.bottom < 0) throw(NegBotException(f.top, f.bottom));
                return is;
        }
};
```


예외 전달의 예

- 예외를 처리하는 코드




```
void main()
{
        Rational r;
        cout << "r 입력(a/b): ";
        try {
                cin >> r;
        }
        catch (char c) {
                cout << "예외 발생: 분수에서 분모가 0입니다.\n";
                cout << "더 진행할 수 없습니다.\n";
                exit(0);
        }
        catch (NegBotException e) {
                r = Rational(-e.top, -e.bottom);
                cout << "예외 발생: 분모를 양수로 변환합니다.\n";
        }
        cout << "r = " << r << endl;
        cout << "r = " << r.real() << endl;
}
```

12.5 응용: MonsterWorld 7

- Monster World 7: 새로운 경쟁의 시작
- 구현: 예외 발생부

Monster World 7: 새로운 경쟁의 시작

- 랭킹 추가 ➔ 게이머와 게이머의 경쟁
  - 랭킹 저장 : 파일
- 만약 랭킹 파일이 열리지 않으면 → 예외 발생
  - 관리자 "암호" 입력
    - **암호가 맞으면 → 새로운 랭킹 파일 생성 및 게임 진행**
    - 암호가 틀리면 → 게임 종료
- 구현이나 수정이 필요한 코드
  - FileException 클래스: 예외 클래스
  - RankingBoard 클래스: 6.6절 참고
    - 랭킹은 아이템을 많이 먹을수록 높아야 함.
    - 파일을 읽고 저장하는 함수에서 예외 발생 코드 포함
  - main()함수
    - 예외 처리 코드 포함

구현: 예외 발생부

```
struct FileException {
    string filename;
    bool bRead;
    FileException(string name, bool b) : filename(name), bRead(b) {}
};
class RankingBoard {
    ...
    void load(string filename) {
        ...
        if (!is)
          throw(FileException(filename,true));
        ...
    }
    void store(string filename) {
        ...
        if (!os)
            throw(FileException(filename, false));
        ...
    }
    void print(string title = "게임 랭킹") {...}
    int add(int nItem, double ipm) {...}
};
```


구현: 예외 처리부

```
void main() {
    RankingBoard rank;
    try { rank.load("MonsterWorld.rnk"); }
    catch (FileException e) {
        char str[80];
        string passwd, correct = "123456";
        cout << "관리자 비밀번호를 입력하세요: ";
        for (int i = 0;; i++) {
            str[i] = getch(); putchar('*');
            ...
        }
        if (passwd != correct) {
            cout << "비밀번호가 맞지 않습니다. 게임 종료.\n\n"; exit(0);
        }
        // 비밀 번호가 맞으면 기본 랭킹으로 게임을 계속 진행함.
    }
    rank.print("[게임 랭킹: 시작]");
    ...
    printf("------게임 종료-------------------\n");
    rank.add(human->nItem, human->nItem/human->total );
    rank.print("[게임 랭킹: 종료]");
    rank.store("MonsterWorld.rnk");
}
```

실행 결과

12.6 const 지시자

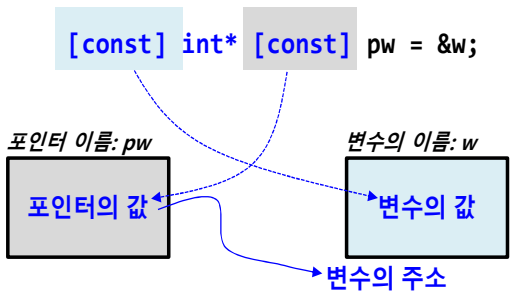
- 변수와 const
- 포인터와 const
- 함수와 const

변수와 const

```
void main() {
    const double phi = 3.14;
    //phi = 3.5;          // Error: 상수 phi 값을 변경할 수 없음
    int x = 10, y = 20, z = 30, w = 40;
    int* px = &x;
    int* const py = &y;   // py 자체를 변경 불가
    const int* pz = &z;   // pz가 가리키는 주소의 값을 변경 불가
    const int* const pw = &w; // pw와 가리키는 주소의 값을 모두 변경 불가
    cout << " x=" << x << " y=" << y << " z=" << z << " w=" << w << endl;
    *px = 100;
    *py = 100;
    //*pz = 100;          // Error: pz가 가리키는 주소의 값을 변경할 수 없음
    //*pw = 100;          // Error: pw가 가리키는 주소의 값을 변경할 수 없음
    cout << " x=" << x << " y=" << y << " z=" << z << " w=" << w << endl;
    cout <<" px="<< px <<" py="<< py <<" pz="<< pz <<" pw="<< pw << endl;
    px = &y;
    pz = &w;
    //py = &x;            // Error: py의 값(주소)를 변경할 수 없음
    //pw = &z;            // Error: pw의 값(주소)를 변경할 수 없음
    ...
}
```

```
C:₩WINDOWS₩system32₩cmd.exe
x=10 y=20 z=30 w=40
x=100 y=100 z=30 w=40
px=00D6FEB4 py=00D6FEA8 pz=00D6FE9C pw=00D6FE90
px=00D6FEA8 py=00D6FEA8 pz=00D6FE90 pw=00D6FE90
```


포인터와 const



함수와 const

```
class Rational {
    ...
    double real() const {
        // (1) top = 2; bottom += 10;
        return (double)top / bottom;
    }
    friend ostream& operator<<(ostream& os, const Rational& f) {
        // (2) f.bottom = 10;
        os << f.top << "/" << f.bottom;
        return os;
    }
    friend istream& operator>>(istream& is, Rational& f) /* (3) const */ {
        is >> f.top >> f.bottom;
        return is;
    }
    Rational operator+(const Rational& f) const {
        // (4) top = 10; f.bottom = 20;
        return Rational(top*f.bottom + f.top*bottom, bottom*f.bottom);
    }
};
```


12.7 형 변환

- 형 변환
- reinterpret_cast<>
- const_cast<>
- static_cast<>
- dynamic_cast<>

형 변환

- C언어의 형 변환

```
int iValue = 10;
double dValue = iValue;   // 자동 형 변환
int* pi = &iValue;
*pi = (int) dValue;       // 강제 형 변환
float* pf = (float*)pi;   // 강제 형 변환 (조심해서 사용)
```

- C++의 네 가지 형 변환 연산자

형 변환 연산자	설명
reinterpret_cast<>	무조건적인 형 변환. C언어의 형 변환과 유사함
const_cast<>	상수형 포인터에서 const를 제거함
static_cast<>	컴파일시 형 변환
dynamic_cast<>	실행 시간의 형 변환

29

reinterpret_cast< >

- 강력함. 조심해서 사용

```
int iValue = 10;
int* pi = &iValue;
float* pf = reinterpret_cast<float*>pi;
pi = reinterpret_cast<int*> iValue;
iValue = reinterpret_cast<int> pf;
```

30

const_cast< >

- const 속성을 제거하기 위해 사용

```
void sub1(const char* s) { ... }
void sub2(char* s) { ... }

const char *s1 = "hello ";
char s2[10] = "world !";

sub1(s1);         // OK: 형이 정확히 일치
sub1(s2);         // OK: 자동 변환: char* -> const char*
sub2(s1);         // Error: const char* -> char* 변환 안됨 !
sub2(s2);         // OK: 형이 정확히 일치


sub2(const_cast<char*>s1);        // OK: const속성 제거
```


static_cast< >

- 컴파일 시간에 논리적으로 가능한 변환인지를 검사
  - 기본 자료형 사이의 변환 허용
  - 상속 관계에 있는 클래스들의 포인터 변환 허용

```
Line* pl = new Line;
Shape* ps1 = pl;                          // OK. 묵시적 변환.
Shape* ps2 = static_cast<Shape*> pl;      // OK. 명시적 변환.
```

```
Shape* ps = new Shape;
Line* pl1 = ps1;                  // Error: 묵시적 변환 안 됨.
Line* pl2 = static_cast<Line*> ps;// OK: 명시적 변환.
```

  - 다른 변환은 허용 않음

```
Shape* ps = new Shape;
Point* pp = static_cast<Point*> ps;       // Error: 상속관계 아님.
ps = static_cast<Shape*> pp;              // Error: 상속관계 아님.
```

dynamic_cast< >

- 상속 관계에 있는 클래스 포인터를 변환
- 실행 시간에 가능한지를 검사하여
- 예: Shape과 Line에 가상 함수가 없다면

```
Line* pl = new Line;
Shape* ps = dynamic_cast<Shape*> pl;       // OK. 명시적 변환.
Shape* ps1 = new Shape;
Line* pl2 = dynamic_cast<Line*> ps1;       // 에러: 하향 형변환.
Line* pl3 = dynamic_cast<Line*> ps;        // 에러: 가상함수 없음.
```

- 예: Shape과 Line에 가상 함수가 있다면

```
Line* pl3 = dynamic_cast<Line*> ps;        // OK: 실행시간 검사.
```

33

12장 요약문제, 연습문제, 실습문제

34

감사합니다!

35