

C++_예외처리, 파일입출력

Visual Studio 2022

예외 처리(Exception Handling)


● 예외 처리

예외란 프로그램 실행 중에 발생할 수 있는 오류를 말한다.

예외 처리를 하면 프로그램이 강제로 종료되지 않고, 문제를 감지하고 적절히 대처할 수 있다.

▪ 예외 처리 기본 구조

```
try {  
    정상적인 처리 내용  
    예외 발생 경우 throw 전달인수;  
}  
catch(throw에서 전달받은 인수){  
    예외 발생시 수행할 내용  
}
```



예외 처리(Exception Handling)

- 예외 처리 예제

나누기 연산에서 0으로 나누었을때(분모가 0인 경우) 예외 처리

```
int n1, n2;  
int quotient, remainder;  
  
cout << "첫번째 수 입력: ";  
cin >> n1;  
  
cout << "두번째 수 입력: ";  
cin >> n2;
```

```
첫 번째 수 입력: 10  
두 번째 수 입력: 0  
10은 0으로 나눌 수 없습니다.
```

try~catch 구문

● 예외 처리 예제

```
int n1, n2;
int quotient, remainder; //몫, 나머지

try {
    cout << "첫번째 수 입력: ";
    cin >> n1;

    cout << "두번째 수 입력: ";
    cin >> n2;

    if (n2 == 0)
        throw n1; //예외 발생 - catch()의 인자로 보냄
    quotient = n1 / n2; //몫
    remainder = n1 % n2; //나머지

    cout << "몫: " << quotient << endl;
    cout << "나머지: " << remainder << endl;
}
catch (int e_n) {
    cout << n1 << "은 0으로 나눌 수 없습니다.\n";
}
```

함수에서 예외 블록 호출하기

- 문자열을 정수로 변환하는 프로그램

문자열을 정수로 변환하는 `stringToInt` 함수를 구현하고,
변환 중에 숫자가 아닌 문자가 발견되면 예외를 발생시키는 프로그램

```
char s[] = "apple";  
cout << strlen(s) << endl;  
  
int x = '0';  
int y = '1';  
  
cout << x << ", " << y << endl; //48, 49  
cout << y - x << endl; //1
```

함수에서 예외 블록 호출하기

- 문자열을 정수로 변환하는 프로그램

```
//문자열을 정수로 변환하는 함수
int stringToInt(const char x[]) {
    int sum = 0;
    int len = strlen(x);

    for (int i = 0; i < len; i++) {
        if (x[i] > '0' && x[i] <= '9')
            sum = sum * 10 + (x[i] - '0');
        else
            throw x; //예외 발생(비정상 문자)
    }
    return sum;
}
```

try~catch 구문

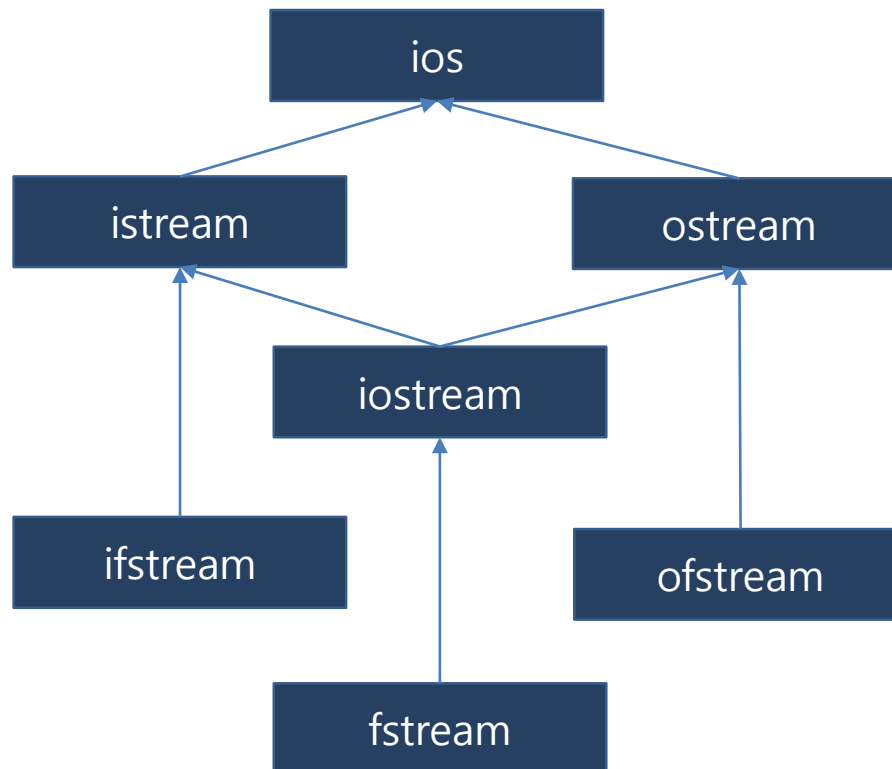
- 문자열을 정수로 변환하는 프로그램

```
int n;  
  
try {  
    //n = stringToInt("12");  
    n = stringToInt("12a");  
  
    cout << "\"12\"는 정수 " << n << "으로 변환됨\n";  
}  
catch (const char* str) {  
    cout << str << " 처리에서 예외 발생!" << endl;  
}
```

파일 입출력

➤ 파일 입출력의 필요성

프로그램 실행 중에 메모리에 저장된 데이터는 프로그램이 종료되면 사라진다. 데이터를 프로그램이 종료된 후에도 계속해서 사용하려면 파일에 저장하고 필요할때 파일을 읽어서 데이터를 사용할 수 있다.



파일 입출력

● 파일 쓰기/ 읽기

1. fstream 헤더 파일을 include 한다.
2. ofstream or ifstream을 사용하여 쓰고, 읽는다.

함수(연산자)	기능 설명
f1 << str << endl;	연산자(<<)로 파일에 쓴다
getline(f1, str)	getline() 함수로 파일을 읽음

파일 입출력

- 파일 쓰기

```
#include <fstream>

ofstream f1("data.txt"); //출력 객체 생성(파일 열기)

if(!f1) // 파일이 없을때 종료

    return 1;

f1 << 내용 << endl; //파일에 쓰기

f1.close() //파일 닫기
```

파일 입출력

- 파일 읽기

```
#include <fstream>

ifstream f1("data.txt"); //입력 객체 생성(파일열기)

if(!f1)

    return 1;

while(!f1.eof()){ //파일의 끝까지 읽기 }

f1.close() //파일 닫기
```

파일 입출력 스트림

● 파일 쓰기 예제

```
#include <iostream>
#include <fstream> //ofstream 사용
using namespace std;

int main()
{
    ofstream f1("data.txt"); // 파일 쓰기 객체 생성
    int x = 1, y = 2;

    if (!f1) {
        cerr << "파일을 열 수 없습니다.\n";
        return 1; // 오류 코드 반환
    }
}
```

파일 입출력 스트림

● 파일 쓰기 예제

```
// 파일에 쓰기  
f1 << x << " " << y << endl;  
f1 << "Good Job!";  
  
f1.close(); // 파일 닫기  
  
return 0;  
}
```

data.txt

파일	편집	보기
----	----	----

1 2		
Good Job!		

파일 입출력 스트림

● 파일 읽기 예제

```
#include <iostream>
#include <fstream> //ifstream 사용
#include <string>

using namespace std;

int main()
{
    ifstream f1("data1.txt"); //f1 객체 생성
    if (f1.fail()) {
        cerr << "파일을 찾을 수 없습니다.\n";
        return 1;
    }
}
```

파일 입출력 스트림

● 파일 읽기 예제

```
string str; //읽은 내용 저장할 변수
/*while (!f1.eof()) { //end of file
    getline(f1, str);
    cout << str << endl;
}*/

while (getline(f1, str)) {
    cout << str << endl;
}

f1.close(); //파일 닫기

return 0;
}
```

```
1 2
Good Job!
```

파일 입출력 스트림

● 파일에 구구단 쓰기

파일	편집	보기
$2 \times 1 = 2$		
$2 \times 2 = 4$		
$2 \times 3 = 6$		
$2 \times 4 = 8$		
$2 \times 5 = 10$		
$2 \times 6 = 12$		
$2 \times 7 = 14$		
$2 \times 8 = 16$		
$2 \times 9 = 18$		
$3 \times 1 = 3$		
$3 \times 2 = 6$		
$3 \times 3 = 9$		
$3 \times 4 = 12$		
$3 \times 5 = 15$		
$3 \times 6 = 18$		
$3 \times 7 = 21$		
$3 \times 8 = 24$		
$3 \times 9 = 27$		

$8 \times 1 = 8$
 $8 \times 2 = 16$
 $8 \times 3 = 24$
 $8 \times 4 = 32$
 $8 \times 5 = 40$
 $8 \times 6 = 48$
 $8 \times 7 = 56$
 $8 \times 8 = 64$
 $8 \times 9 = 72$

$9 \times 1 = 9$
 $9 \times 2 = 18$
 $9 \times 3 = 27$
 $9 \times 4 = 36$
 $9 \times 5 = 45$
 $9 \times 6 = 54$
 $9 \times 7 = 63$
 $9 \times 8 = 72$
 $9 \times 9 = 81$

파일 입출력 스트림

● 파일에 구구단 쓰기

```
ofstream f1("gugudan.txt"); // 파일 쓰기 객체 생성

if (!f1) {
    cerr << "파일을 열 수 없습니다.\n";
    return 1; // 오류 코드 반환
}

// 구구단 쓰기
for (int i = 2; i <= 9; i++) {
    for (int j = 1; j <= 9; j++) {
        f1 << i << " x " << j << " = " << (i * j) << endl;
    }
    f1 << endl;
}

f1.close(); // 파일 닫기
```

파일 입출력 스트림

● 구구단 읽기

```
ifstream f1("gugudan.txt"); // 파일 객체 생성
string str;

if (!f1) {
    cerr << "파일을 열 수 없습니다.\n";
    return 1; // 오류 코드 반환
}

// 구구단 읽기
while (getline(f1, str)) {
    cout << str << endl;
}

f1.close(); // 파일 닫기
```

```
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18

3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
3 x 9 = 27
```

파일 쓰기 추가 모드

- 파일 쓰기 추가 모드

파일 쓰기 추가 모드 – `ios::app` 인자 명시함

모드 구분	기능 설명
<code>ios::binary</code>	<code>ofstream fin("data.txt", <code>ios::binary</code>);</code>

파일에 성적 저장하기

- 키보드 입력으로 성적 저장하기

```
int sid; //학번
string name, dept; //이름, 학과

//키보드 입력
cout << "학번 입력(숫자)>> ";
cin >> sid;
cin.ignore(); //버퍼에 남은 개행('\n') 제거

cout << "이름 입력>> ";
//cin >> name;
getline(cin, name); //공백 포함 입력

cout << "학과 입력>> ";
getline(cin, dept);
```

파일에 성적 저장하기

● 키보드 입력으로 성적 저장하기

```
//파일 열기(추가 모드 - append)
ofstream fout("student.txt", ios::app);
if (!fout) {
    cerr << "파일을 열 수 없습니다.\n";
    return 1;
}

//파일 쓰기
fout << "학번: " << sid << endl;
fout << "이름: " << name << endl;
fout << "학과: " << dept << endl;

fout.close();

cout << "파일 쓰기 완료!!\n";
```

학번: 2025101
이름: 신유진
학과: 컴퓨터 과학과
학번: 2025102
이름: 이상영
학과: 전자공학과

성적 리스트 만들기

● 성적 리스트 만들기

```
1번째 학생의 이름 : 이정후  
영어점수 입력 : 87  
수학점수 입력 : 88  
2번째 학생의 이름 : 최민정  
영어점수 입력 : 91  
수학점수 입력 : 86  
3번째 학생의 이름 : 신유빈  
영어점수 입력 : 87  
수학점수 입력 : 76  
데이터가 성공적으로 저장되었습니다!
```

```
이정후 87 88 87.5  
최민정 91 86 88.5  
신유빈 87 76 81.5
```

성적 리스트 만들기

- 성적 리스트 만들기

```
class Student {  
private:  
    string name;    //이름  
    int eng;        //영어 점수  
    int math;       //수학 점수  
    double avg;     //평균  
  
public:  
    // 설정자(setter) 메서드들  
    void setName(string name) { this->name = name; }  
    void setEng(int eng) { this->eng = eng; }  
    void setMath(int math) { this->math = math; }  
  
    // 평균 계산 메서드  
    void calculateAvg() {  
        avg = (double)(eng + math) / 2;  
    }  
}
```

성적 리스트 만들기

- 성적 리스트 만들기

```
//접근자(getter) 메서드
string getName() { return name; }
int getEng() { return eng; }
int getMath() { return math; }
double getAvg() { return avg; }
};

int main()
{
    ofstream fout("scorelist.txt");
    Student students[3];

    if (!fout) {
        cerr << "Error: 파일을 열 수 없습니다.\n";
        return 1;
    }
}
```


성적 리스트 만들기

● 성적 리스트 만들기

```
//키보드 입력
for (int i = 0; i < 3; i++) {
    string name;
    int eng, math;

    cout << i + 1 << "번째 학생의 이름: ";
    getline(cin, name);
    students[i].setName(name);

    cout << "영어 점수 입력: ";
    cin >> eng;
    students[i].setEng(eng);

    cout << "수학 점수 입력: ";
    cin >> math;
    students[i].setMath(math);

    cin.ignore(); //개행문자 제거 - getline() 처리

    students[i].calculateAvg(); //평균 계산
}
```

성적 리스트 만들기

- 성적 리스트 만들기

```
//파일에 쓰기
for (int i = 0; i < 3; i++) {
    fout << students[i].getName() << " "
        << students[i].getEng() << " "
        << students[i].getMath() << " "
        << students[i].getAvg() << endl;
}

fout.close();

cout << "데이터가 성공적으로 저장되었습니다.\n";
```

```
이정후 87 88 87.5
최민정 91 86 88.5
신유빈 87 76 81.5
```

바이너리 파일 읽고 쓰기

● 바이너리 파일 모드

바이너리 모드(binary mode)는 파일을 있는 그대로(바이트 단위) 읽고 쓰기 위해 사용하는 모드입니다

모드 구분	기능 설명
ios::binary	<code>ifstream fin("data.bin", ios::binary);</code>

함수의 원형	기능 설명
read(char* str, streamsize _count)	바이너리 파일 읽기
write(char* str, streamsize _count)	바이너리 파일 쓰기

바이너리 파일 읽고 쓰기

- 바이너리 파일 쓰기, 읽기

```
int num = 1234;

// 바이너리 모드로 저장
ofstream fout("data.bin", ios::binary);

//&num은 시작 주소
//(char*) 형변환 - write함수가 문자 배열을 받음
fout.write((char*)&num, sizeof(num));
fout.close();

// 바이너리 파일 읽기
ifstream fin("data.bin", ios::binary);

int readNum;
fin.read((char*)&readNum, sizeof(readNum));
cout << readNum << endl;
fin.close();
```

바이너리 파일 읽고 쓰기

● 이미지 파일 복사하기

```
string source = "cake.jpg";    //원본 파일
string copied = "copycake.jpg"; //복사된 파일

//소스파일 열기(binary 모드)
ifstream fin(source, ios::binary);
if (!fin) {
    cerr << "파일을 열 수 없습니다.\n";
    return 1;
}

//복사파일 열기
ofstream fout(copied, ios::binary);
if (!fout) {
    cerr << "파일을 열 수 없습니다.\n";
    return 1;
}
```



바이너리 파일 읽고 쓰기

● 이미지 파일 복사하기

```
/*int c;
while ((c = fin.get()) != EOF) { //1바이트씩 읽어서
    fout.put(c); //1바이트 씩 쓰기
}*/

//버퍼로 읽기
char buf[1024];
/*while (!fin.eof()) {
    fin.read(buf, 1024); //최대 1024 바이트를 읽어 배열에 저장
    int n = fin.gcount(); //실제 읽은 바이트 수
    fout.write(buf, n); //읽은 바이트 수만큼 쓰기
}*/

while (fin) {
    fin.read(buf, sizeof(buf));
    fout.write(buf, fin.gcount());
}
fin.close();
fout.close();
```