dongguk UNIVERSITY

심화프로그래밍

실습강의 14주차

실습강의 소개

- **실습 진행 방법**
  - 간단한 이론 복습 및 해당주차 실습과제 설명
  - 실습 후 보고서와 소스코드를 압축하여 **수요일 자정(23:59)까지**

    꼭!! 이클래스 제출(**이메일 제출 불가, 반드시 이클래스를 통해 제출**)
  - 실습과제 제출기한 엄수(제출기한 이후로는 0점 처리)
- **Q & A**
  - 이클래스 및 실습조교 이메일을 통해 질의응답
  - **이메일 제목 : [심화프로그래밍_홍길동] *본인 과목명과 성명 꼭 작성!!**
  - **실습조교 메일 주소 : 0hae@dgu.ac.kr**

실습 보고서 작성 방법 [1/2]

- **실습 보고서**
  - **문제 분석: 실습 문제에 대한 요구 사항 파악, 해결 방법 등 기술**
  - **프로그램 설계 및 알고리즘**
    - 해결 방법에 따라 프로그램 설계 및 알고리즘 등 기술
    - e.g.) 문제 해결 과정 및 핵심 알고리즘 기술
  - **소스코드 및 주석**
    - 소스코드와 그에 해당하는 주석 첨부
    - 각각의 함수가 수행하는 작업, 매개변수, 반환 값 등을 명시
    - 소스코드 전체 첨부**(소스코드 화면 캡처X, 소스코드는 복사/붙여넣기로 첨부)**
  - **결과 및 결과 분석**
    - 결과 화면을 캡쳐 하여 첨부, 해당 결과가 도출된 이유와 타당성 분석
  - **소감**
    - 실습 문제를 통해 습득할 수 있었던 지식, 느낀 점 등을 기술

실습 보고서 작성 방법 [2/2]

- **제출 방법**
  - 보고서, 소스코드를 1개의 파일로 압축하여 e-class "과제" 메뉴를 통해 제출
    - **"이름학번실습주차.zip" 형태로 제출(e.g. :김동국19919876실습14.zip)**
    - 파일명에 공백, 특수 문자 등 사용 금지
- **유의 사항**
  - 보고서의 표지에는 학과, 학번, 이름, 담당 교수님, 제출일자 반드시 작성
  - 정해진 기한내 제출
    - 기한 넘기면 **0점** 처리
    - 이클래스가 과제 제출 마지막 날 오류로 동작하지 않을 수 있으므로, 최소 1~2일전에 제출
    - **과제 제출 당일 이클래스 오류로 인한 미제출은 불인정**
  - 소스코드, 보고서를 자신이 작성하지 않은 경우 **실습 전체 점수 0점 처리**
  - **Visual Studio 2019 또는 Sharstra 웹 IDE 기반 학습 프로그램 사용하여 실습 진행**

텍스트 파일 & 바이너리 파일

- 파일(File)
  - 저장 매체에 저장되는 정보
  - 바이트나 블록 단위로 입출력됨

- 텍스트 파일
  - 글자 혹은 문자로만 구성되는 문서 파일
  - 각 글자마다 고유한 바이너리 코드(이진 코드-ASCII/UNI code)로 구성
  - 텍스트 파일에는 문자 코드만 저장됨
  - e.g.) txt, html, xml, c/c++/java source file

- 바이너리 파일
  - 사진 이미지, 오디오, 그래픽 이미지, 컴파일된 코드 등 문자로 표현되지 않는 바이너리 정보들로 구성된 파일
  - 각 파일을 만든 응용프로그램만이 해석할 수 있음
  - e.g.) jpeg, bmp, mp3, hwp, doc, ppt, obj, exe

C++ 파일 입출력

- 파일 입출력 라이브러리의 클래스
  - ifstream(읽기) / ofstream(쓰기) / fstream(읽기/쓰기)
- 파일 입출력 스트림은 파일을 프로그램과 연결한다
  - ifstream, ofstream은 각각의 객체를 통해 프로그램과 파일을 연결하여 파일을 읽고 쓰도록 한다
- 헤더 파일과 namespace
  - ifstream, ofstream, fstream 클래스의 객체를 생성하기 이해 <fstream>헤더 파일을 추가하고 std 이름 공간을 사용하도록 한다.
- 파일 입출력 모드
  - 텍스트 I/O
    - 파일에 있는 바이트를 문자로만 해석하고, 문자들만 기록하는 입출력 방식
  - 바이너리 I/O
    - 바이트 단위로 바이너리 데이터를 입출력하는 방식(텍스트/바이너리 파일 상관 없음)

<<,>>연산자 이용한 파일 입출력

- 파일 쓰기를 위한 스트림 객체 생성

```
ofstream fout; // 파일 출력 스트림 객체 fout 생성
```

- 파일 열기
  - 파일 입/출력 스트림 객체의 open() 멤버 함수를 호출하여 파일을 열고 스트림을 연결

```
fout.open("song.txt"); // song.txt 파일 열기
```

- 파일 열기 성공 검사
  - 파일 열기 실패를 처리하기 위해 fout 스트림의 operator!() 연산자 함수를 실행한다. 열기가 실패한 경우 true 리턴

```
if(!fout) { ... } // fout 스트림의 파일 열기가 실패한 경우
// if(!fout.is_open()) { ... }
```

- << 연산자를 이용한 파일 쓰기
  - << 연산자는 문자만 저장
- 파일 닫기

```
fout.close();
```

파일 모드

- 파일을 열 때 어떤 파일 입출력을 수행할 것인지 알리는 것
- 파일 모드 상수

파일 모드	설명
ios::in	파일 읽기 모드
ios::out	파일 쓰기 모드
ios::ate	(at end) 쓰기 위해 파일을 염. 파일 포인터를 파일 끝에 둔다. 파일 포인터를 옮겨 파일 내의 임의의 위치에 쓸 수 있다.
ios::app	파일 쓰기시에만 적용. 자동으로 파일 포인터가 파일 끝으로 옮겨져, 항상 파일의 끝에 쓰기가 이루어짐
ios::binary	바이너리 I/O 로 파일을 연다. (디폴트는 텍스트I/O)

텍스트 I/O

- 파일 열기
  - ios::binar로 지정하는 것이 아니면 디폴트로 텍스트I/O로 입출력이 이루어짐
- get() 과 put()
  - get() 은 파일에서 문자 한 개를 읽는다
  - put() 은 문자 한 개를 파일에 기록한다
- 문자열-getline()
  - 문자열 단위로 텍스트 파일을 읽는 두가지 방법
    - istream의 getline(char* line, int n)-char 타입의 문자열
    - getline(ifstream& fin, string& line)-string 문자열 객체(<string>헤더파일의 전역 함수)

바이너리 I/O

- 파일 열기
  - 이미지 사진 파일 등 바이너리 파일에서도 텍스트 파일에서 사용한 것과 동일하게 get(), put() 함수를 이용하여 문자 단위로 바이너리 파일을 다룰 수 있음

- 블록 단위 입출력
  - read() 와 write()로 블록 단위 파일 입출력할 수 있음
  - read()는 한 블록을 읽는 함수, 실제 익은 바이트 수는 gcount()로 알 수 있음

스트림 상태 검사

- 스트림 상태를 나타내는 비트
  - eofbit – 파일의 끝을 만났을 때 1로 세팅
  - failbit – 입력의 오류나 쓰기가 금지된 곳에 쓰기를 시행하는 등 전반적인 I/O 실패 시 1로 세팅
  - badfit - 유효하지 않은 입출력 명령이 주어졌을 때 1로 세팅

- 스트림 상태 검사하는 멤버 함수
  - eof()
  - fail()
  - bad()
  - good()
  - clear()

임의 접근

- 임의 접근이란?
  - 파일 내에 원하는 위치로 옮겨 다니면서 입출력하는 방식

- C++ 파일 포인터(2가지)
  - get pointer
    - 파일 내의 읽기 지점을 가리키는 파일 포인터
    - ios::in 으로 열려진 fstream 객체: get, read, getline, >>
  - put pointer
    - 파일 내의 쓰기 지점을 가리키는 파일 포인터
    - ios::out 으로 열려진 fstream 객체: put, write, <<

예외 처리

- 실행 오류
  - 예상치 못한 입력 값이나 논리적 실수로 실행 중 발생한 오류
- 오류 처리
  - 실행 중 발생한 오류로 프로그램이 비정상 종료하는 코드를 수정하여 오류 처리

- 예외 처리
  - 예외 발생 시 대처하는 코드
  - C++ 에서는 try-throw-catch 구조로 예외를 처리
  - 예외를 탐지하고 예외 처리를 지시하는 throw문을 가진 코드들을 try{} 블록으로 묶는다
  - 하나의 try{} 블록에는 반드시 1개 이상의 catch{} 블록이 연결되어야 하며 throw 문에 의해 던져진 예외를 처리할 catch{} 블록이 없다면 강제 종료

```
try {
    // 예외 발생 가능한 코드
    if (condition)
        throw MyException(); // 예외를 발생시킴
}
catch (MyException& e) {
    // 예외 처리 코드
    std::cerr << "Caught an exception: " << e.what() << std::endl;
}
```


C++ 코드와 C 코드의 링킹

- C++ 코드와 C 코드를 함께 사용해야 할 때 C++는 C코드를 포함하고 링크하는 것이 가능

```
// C 코드 (mylibrary.h)
#ifdef __cplusplus
extern "C" {
#endif

void myCFunction(); // C 함수 선언

#ifdef __cplusplus
}
#endif

// C++ 코드
#include "mylibrary.h"

void myCFunction() {
    // C 함수의 구현
}

int main() {
    myCFunction(); // C 함수 호출
    return 0;
}
```


연습문제(12장)

1. 다음에서 텍스트 파일이 아닌 것은?

   ① test.hwp ② test.cpp ③ test.htm ④ test.c

2. 다음에서 텍스트 파일은?

   ① test.doc ② test.jpg ③ test.au ④ iostream

3. 파일 입출력을 하기 위해 필요한 헤더 파일은?

4. 텍스트 I/O와 바이너리 I/O에 대해 설명한 것 중 옳은 것은?

   ① 텍스트 I/O 방식으로 텍스트 파일과 바이너리 파일을 모두 읽을 수 있다.

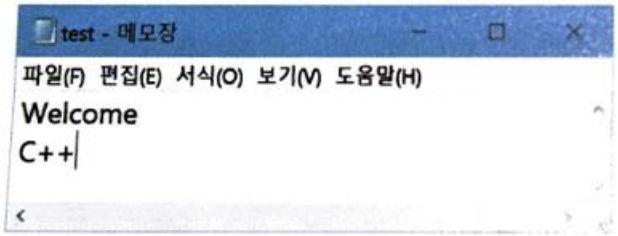
   ② ifstream이나 ofstream의 디폴트 방식은 바이너리 I/O이다.

   ③ 텍스트 I/O와 바이너리 I/O는 파일의 끝을 알아내는 방법에 차이가 있다.

   ④ 텍스트 I/O와 바이너리 I/O는 '\n'을 입출력하는데 차이가 있다.

연습문제(12장)

5. 메모장으로 Welcome 입력 후 <Enter> 키를 입력하고, C++ 입력 후 <Enter> 키 없이 test.txt 파일에 저장하였다.



(1) test.txt 파일의 바이트 수는 얼마인가? 속성 창으로 보라.

(2) ASCII 표를 참고하여 test.txt에 저장된 바이트를 16진수로 말하라.

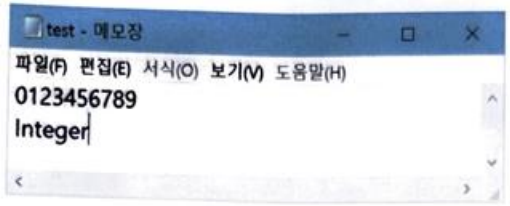
(3) 아래 코드는 파일의 문자 개수를 세는 코드이다. 실행하였을 때 출력되는 count 값은 얼마인가?

```
ifstream fin("test.txt");
int ch, count=0;
while((ch = fin.get()) != EOF)
   count++;
cout << count;
```

(4) 위 코드에서 ifstream fin("test.txt");를 ifstream fin("test.txt", ios::binary);로 변경하면 출력되는 count 값은 얼마인가?

연습문제(12장)

6. 다음과 같이 메모장으로 test.txt 파일을 작성하였다. 0123456789 뒤에 <Enter> 키가 있지만, Integer 뒤에는 <Enter> 키가 없다.



(1) test.txt 파일의 바이트 수는 얼마인가?

(2) ASCII 표를 참고하여 test.txt에 저장된 바이트를 16진수로 말하라.

(3) 아래 코드는 파일의 문자 개수를 세는 코드이다. 실행하였을 때 출력되는 count 값은 얼마인가?

```
ifstream fin("test.txt");
char ch;
int count=0;
while(true) {
   fin.get(ch);
   if(fin.eof()) break;
   count++;
}
cout << count;
```

(4) 위 코드에서 ifstream fin("test.txt"); 를 ifstream fin("test.txt", ios::binary);로 변경하면 출력되는 count 값은 얼마인가? count 값이 서로 다른 이유는 무엇인가?

연습문제(12장)

7. 다음 두 라인을 최대한 간소화하여 한 라인으로 작성하라.

```
ifstream fin;
fin.open("test.txt", ios::in);
```

8. 다음과 같이 파일을 여는 코드가 있다. 파일 열기가 실패하면 "열기 실패"라고 출력하고 리턴하는 if 문을 작성하라.

```
ofstream fout("song.txt");
```

연습문제(12장)

9. ifstream 타입의 스트림 fin에 대해 다음 코드의 의미를 잘 설명한 것은?

```
if(!fin) {
   // 문장 A
}
```

① 파일 열기가 실패하면 fin이 NULL이 되므로, fin이 NULL인지 검사하며, 열기가 실패하면 문장 A를 실행한다.
② if 문을 if(fin == NULL)로 바꾸어도 동일하다.
③ ifstream 클래스의 ! 연산자가 실행되어 파일 열기가 실패하였으면 true를 리턴한다.
④ if 문을 if(fin.is_open())으로 바꾸어도 동일하다.

10. 다음 코드의 뜻을 정확히 설명한 것은?

```
ifstream fin;
fin.open("test.www", ios::in | ios::binary);
```

① test.www 파일은 텍스트 파일이다.
② test.www 파일이 텍스트 파일인지 바이너리 파일인지 모른다.
③ test.www 파일을 텍스트 모드로 읽기 위해 열었다.
④ test.www 파일이 바이너리 파일이므로 바이너리 모드로 열었다.

연습문제(13장)

1. 프로그램 실행 중 오동작이나 결과에 영향을 미치는 실행 오류 발생을 무엇이라고 부르는가?

① 예외 ② 컴파일 오류 ③ 동적 바인딩 ④ 인터럽트

2. 예외 처리와 관련된 C++ 키워드가 아닌 것은?

① try ② throw ③ except ④ catch

3. 하나의 try { } 블록에 연결되는 catch() { } 블록은?

① 있어도 되고 없어도 된다.
② 반드시 1개만 있어야 한다.
③ 여러 개 만들 수 있다.
④ 개발자가 catch() { } 블록을 지정하지 않으면 디폴트 catch() { } 블록이 만들어진다.

연습문제(13장)

4. catch() { } 블록에 대해 틀린 설명은?

① throw 문에서 던진 예외를 처리하는 블록이다.

② 예외 파라미터는 매개 변수 선언과 동일하지만, 매개 변수는 오직 하나만 선언한다.

③ catch() { } 블록은 예외를 처리한 다음 실행을 중단한다.

④ 하나의 catch() { } 블록은 오직 하나의 예외 타입만 처리한다.

5. 다음 코드의 실행 결과는?

```
int m=3;
try {
   throw &m;
}
catch(int* y) {
   *y = 100;
   cout << m;
}
```

연습문제(13장)

6. 다음 코드의 실행 결과는?

```
try {
   throw 3;
}
catch(int x) {
   try {
      cout << x;
      throw "aa";
   }
   catch(const char* p) {
      cout << p;
   }
}
```

연습문제(13장)

7. 다음 프로그램의 실행 결과는?

```
try {
   throw 3;
}
catch(int x) {
   try {
      throw x + 1;
      cout << x;
   }
   catch(int y) {
      cout << y;
   }
}
```

연습문제(13장)

8. 다음 각 문항에 따라 다음 프로그램의 실행 결과는?

```
int n, m;
try {
   if(n == 0)
      throw "0을 다루지 않음";
   if(m % n == 0)
      throw 0;
   cout << m % n;
}
catch(int x) {
   cout << x;
}
catch(const char* s) {
   cout << s;
}
```

(1) n = 0; m = 5; 의 경우

(2) n = 5; m = 10; 의 경우

(3) n = 6; m = 10; 의 경우

연습문제(13장)

9. 다음 함수가 있다.

```
void printDouble(int m) {
   try {
      if(m < 0) throw m;
      else m = m*2;
   }
   catch(int y) {
      cout << "음수는 다루지 않음";
   }
   cout << m << endl;
}
```

다음과 같이 호출할 때 실행 결과는?

(1) printDouble(5);

(2) printDouble(-3);

연습문제(13장)

10. 다음 코드를 실행하면 어떻게 되는가?

```
int n = 20;
throw n;
try {
   n = n/2;
}
catch(int x) {
   cout << n;
}
```

실습과제(12장)

1. C:₩windows₩system.ini를 c:₩temp₩system.txt로 복사하는 동안 10%를 진행할 때 마다 '.'과 바이트 크기를 다음과 같이 출력하는 프로그램을 작성하라.

```
복사 시작...
.21B 10%
.21B 20%
.21B 30%
.21B 40%
.21B 50%
.21B 60%
.21B 70%
.21B 80%
.21B 90%
.21B 100%
219B 복사 완료
```

실습과제(12장)

2. 단어가 들어 있는 words.txt 파일을 읽어 단어 별로 vector<string>에 저장하고 사용자가 입력한 문자열로 시작되는 모든 단어를 출력하는 프로그램을 작성하라. "exit"을 입력하면 프로그램을 종료하라.

```
... words.txt 파일 로딩 완료
검색을 시작합니다. 단어를 입력해 주세요.
단어>> love
love
lovebird
lovelorn
단어>> fat
fat
fatal
fate
fateful
father
fathom
fatigue
fatten
fatty
fatuous
단어>> fathor
발견할 수 없음
단어>> exit
```

실습과제(13장)

3. 다음 코드에서 getFileSize() 함수는 매개 변수에 NULL이 넘어오면 -1을, 파일을 열수 없으면 -2를 리턴하고, 정상적인 경우 파일 크기를 리턴한다.

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

int getFileSize (const char* file) {
    if(file == NULL) return -1; // file이 NULL 포인터이면 -1 리턴
    ifstream fin(file);
    if(!fin) return -2; // 열기가 실패하면 -2 리턴
    fin.seekg(0, ios::end);
    int length = fin.tellg();
    return length;
 }

int main() {
    int n = getFileSize("c: ₩₩windows₩₩system.ini");
    cout <<"파일 크기 = "<< n << endl; // 파일 크기 = 219가 출력됨
    int m = getFileSize(NULL);
    cout <<"파일 크기 = "<< m << endl; // 파일크기 = -1이 출력됨
}
```

실습과제(13장)

3. (앞 슬라이드 계속) 위 프로그램을 수정하여 try-throw-catch 블록으로 예외 처리하고 프로그램을 완성하라. 프로그램을 실행하면 다음과 같다.

```
파일 크기 = 219
예외 발생 : 파일명이 NULL 입니다.
```

실습과제(13장)

4. try-catch 블록을 사용하면, 프로그램 내 오류 검사 if 문으로 인해 반복되고 길어지는 코드를 간소화할 수 있다. 다음 함수 copy()는 int [] 배열을 복사하여 복사본 배열의 포인터를 리턴한다. 복사가 여의치 않는 경우 참조 매개변수인 failCode에 적절한 오류 코드를 삽입하고 NULL을 리턴한다. copy() 함수의 원형을 int* copy(int* src, int size);로 고치고 copy()와 main() 모두 try-catch 블록을 이용하여 수정하라. 코드가 튼튼하고 단순해진다.

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

int* copy (int* src, int size, int& failCode) {
    int* p = NULL;
    if(size < 0) {
        failCode = -1; // too small
        return NULL;
    }
    else if(size > 100) {
        failCode = -2; // too big
        return NULL;
    }
    p = new int [size]; //메모리 할당
    if(p == NULL) {
```

(뒤 슬라이드 계속)

실습과제(13장)

```
        failCode = -3; // memory short
        return NULL;
    }
    else if(src == NULL) {
        failCode = -4; // NULL source
        delete [] p;
        return NULL;
    }
    else { // 정상적으로 배열을 복사하는 부분
        for (int n=0; n<size; n++) p[n] = src[n];
        failCode = 0;
        return p;
    }

int main() {
    int x[] = {1,2,3};
    int ret;
    int *p = copy(x, 3, ret);
    for (int i=0; i<3; i++) cout << p[i] << ' ';
    cout << endl;
    delete [] p;
}
```

```
1  2  3
```