Chapter 12. Stream and File I/O

박 종 혁 교수
UCS Lab
(http://www.parkjonghyuk.net)

Tel: 970-6702
Email: jhpark1@snut.ac.kr

Learning Objectives

- I/O Streams
  - File I/O
  - Character I/O
- Stream I/O 도구
  - 입력 : 파일 이름
  - 서식 출력, 플래그 설정
- Stream 계층
  - 상속 미리보기
- 파일 Random Access

개요

- Streams
  - 특수 개체
  - 프로그램 입력 및 출력을 제공
- File I/O
  - 상속을 사용
  - File I/O 는 매우 유용하게 사용

Streams

- 문자의 흐름
- 입력 스트림
  - 프로그램 흐름
    - 키보드에서 입력 가능
    - 파일에서 입력 가능
- 출력 스트림
  - 프로그램 흐름
    - 화면으로 출력 가능
    - 파일로 출력 가능

Streams 사용

- 이미 스트림을 사용중
  - cin
    - 키보드를 통한 스트림 입력
  - cout
    - 스크린을 통한 스트림 출력
- 다른 스트림 정의
  - 파일
  - cin, cout 과 유사한 사용

cin, cout과 유사한 stream 사용

- 고려:
  - 주어진 프로그램이 일부 파일에서 inStream 정의:
    int theNumber;
    inStream >> theNumber;
    - theNumber 에 할당된 스트림에서 값을 읽음
  - 프로그램에서 정의한 outStream으로 파일에 쓰기
    outStream << "theNumber is " << theNumber;
    - Stream에 저장된 값을 파일에 쓰기

Files

- 텍스트 파일 사용
- 파일에서 읽기
  - 프로그램이 입력을 받을 경우
- 파일에 쓰기
  - 프로그램이 출력을 내보낼 경우
- 파일의 처음부터 시작해서 끝까지
  - 사용가능한 다른 방법
  - 여기서는 간단한 텍스트 파일 액세스를 다룸

File Connection

- 첫째로 스트림 객체와 파일을 연결
- For input:
  - File → ifstream object
- For output:
  - File → ofstream object
- Classes ifstream and ofstream
  - 라이브러리로 정의된 <fstream>
  - Std namespace

File I/O Libraries

- 프로그램의 파일 입출력을 모두 허용하려면:

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

OR

```
#include <fstream>
using std::ifstream;
using std::ofstream;
```

Streams 선언

- 스트림은 다른 클래스 변수처럼 선언:

```
ifstream inStream;
ofstream outStream;
```

- 파일과 연결:

```
inStream.open("infile.txt");
```

  - "opening the file“
  - *open* 멤버 함수 사용
  - 전체 경로명 지정가능

Streams 사용

- 한번 선언 → 일반적으로 사용!

```
int oneNumber, anotherNumber;
inStream >> oneNumber >> anotherNumber;
```

- 유사한 출력 스트림:

```
ofstream outStream;
outStream.open("outfile.txt");
outStream      << "oneNumber = " << oneNumber
               << " anotherNumber = "
               << anotherNumber;
```

  - 출력 파일에 항목을 보냄

File Names

- 프로그램 및 파일
- 프로그램에서 두개의 파일이름을 갖음
  - 외부 파일 이름
    - 물리적 파일 이름
    - Ex)"infile.txt"
    - 때로는 실제 파일이름으로 간주
    - 열기 위해 프로그램에서 한번 사용
  - 스트림 이름
    - 논리적 파일 이름
    - 프로그램은 모든 파일 작업에 대해 이 이름을 사용

Closing Files

- 다 사용한 파일은 닫아야함
  - 프로그램에서 입력과 출력이 완료되면 파일에서 스트림 연결을 끊음
  - In action:

```
inStream.close();
outStream.close();
```

- 프로그램이 끝날 때 파일이 자동으로 종료

File Flush

- 출력은 종종 버퍼를 사용
  - 파일에 기록하기 전에 일시적으로 저장
- 필요에 의해 강제적으로 쓰기 가능: outStream.flush();
  - *flush* 멤버함수는 모든 스트림 출력에 사용
  - 모든 버퍼 출력은 물리적으로 작성
- 파일을 닫으면 자동으로 flush() 호출

File Example: **Display 12.1** Simple File Input/Output (1 of 2)

Display 12.1 Simple File Input/Output

```
1  //Reads three numbers from the file infile.txt, sums the numbers,
2  //and writes the sum to the file outfile.txt.
3  #include <fstream>
4  using std::ifstream;
5  using std::ofstream;
6  using std::endl;

7  int main( )
8  {
9      ifstream inStream;
10     ofstream outStream;

11     inStream.open("infile.txt");
12     outStream.open("outfile.txt");

13     int first, second, third;
14     inStream >> first >> second >> third;
15     outStream << "The sum of the first 3\n"
16               << "numbers in infile.txt\n"
17               << "is " << (first + second + third)
18               << endl;
```

A better version of this program is given in Display 12.3.

File Example:
Display 12.1 Simple File Input/Output (1 of 2)

```
19          inStream.close( );
20          outStream.close( );

21          return 0;
22      }
```

SAMPLE DIALOGUE

There is no output to the screen and no input from the keyboard.

infile.txt
(Not changed by program)

```
1
2
3
4
```

outfile.txt
(After program is run)

```
The sum of the first 3
numbers in infile.txt
is 6
```

Appending to a File

- 표준 open 은 빈 파일에서 시작
  - 파일이 존재 → 내용 분실
- 추가를 위한 open:

```
ofstream outStream;
outStream.open("important.txt", ios::app);
```

  - 파일이 없음 → 파일 생성
  - 파일이 존재 → 파일의 끝에 추가
  - 2번째 인자인 ios는 <iostream> 라이브러리에서 지정

File Opens을 위한 대체 구문

- 선언에서 파일 이름을 지정 가능
  - 생성자에 인수로 전달
- ifstream inStream;
  inStream.open("infile.txt");

  EQUIVALENT TO:

  ifstream inStream("infile.txt");

Checking File Open Success

- 파일 오픈은 실패할수 있음
  - 파일이 존재하지 않는 경우
  - 파일에 대한 쓰기 권한이 없는 경우
  - Unexpected results
- 멤버함수 fail()
  -
```
inStream.open("stuff.txt");
if (inStream.fail())
{
  cout << "File open failed.\n";
  exit(1);
}
```

문자 I/O with Files

- 모든 cin 과 cout 문자 I/O 는 파일과 같음
- 멤버 함수:
  - get, getline
  - put, putback,
  - peek, ignore

Checking End of File

- 파일의 끝까지 loop
  - 전형적인 방법
- 파일의 끝을 검사하는 두가지 방법
  - eof() 멤버함수

```
inStream.get(next);
while (!inStream.eof())
{
    cout << next;
    inStream.get(next);
}
```

    - 파일이 끝날 때까지 문자를 읽음
    - eof() 멤버함수는 BOOL 반환

End of File Check with Read

- 두번째 방법
  - 읽기 작업에서 BOOL 값 반환!
    (inStream >> next)
    - 성공적으로 읽으면 true 반환
    - 파일의 끝을 지나 읽기를 시도하면 false 반환
  - In action:

```
double next, sum = 0;
while (inStream >> next)
  sum = sum + next;
cout << "the sum is " << sum << endl;
```

Tools: 파일 이름 입력받기

- 스트림 오픈 작업
  - open() 에 사용할 string 변수 지정

```
char fileName[16];
ifstream inStream;
cout << "Enter file name: ";
cin >> fileName;
inStream.open(fileName);
```

  - 더 많은 유연성 제공

Formatting Output with Stream Functions

- 1장의 "magic formula":

```
cout.setf(ios::fixed);
cout.setf(ios::showpoint);
cout.precision(2);
```

- 모든 출력 스트림에 사용 가능
  - 파일 스트림은 cout 객체와 동일한 멤버함수를 갖음

Output Member Functions

- 고려:

```
outStream.setf(ios::fixed);
outStream.setf(ios::showpoint);
outStream.precision(2);
```

- precision(x) 멤버 함수
  - 소수점 뒤에 "x" 자리로 작성된 십진수
- setf() 멤버 함수
  - 출력 플래그의 다양한 설정 허용

More Output Member Functions

- 고려:

  outStream.width(5);

- width(x) 멤버 함수
  - 출력할 너비를 "x“값으로 지정
  - 다음 출력될 값에만 영향

Flags

- setf() 멤버함수
  - 출력 플래그의 상태 설정
- 모든 출력 스트림은 setf() 를 멤버로 갖음
  - 플래그는 <iostream> 라이브러리에 저장

setf() Examples

- 일반적인 사용:
  - outStream.setf(ios::fixed);
    - 십진수 고정 소수정
  - outStream.setf(ios::showPoint)
    - 항상 소수점 포함
  - outStream.setf(ios::right);
    - 오른쪽 기준 출력
- 프래그의 다중 사용:

```
outStream.setf(ios::fixed | ios::showpoint |
                                    ios::right);
```

조작자

- 조작자 정의:
  "기존과는 다른 새로운 차원의 방법으로 호출하는 함수"
  - 인자를 가질 수도 가지지 않을 수도 있음
  - 삽입연산자 다음에 위치
  - 멤버함수와 동일한 기능 수행!
  - 일반적으로 조작자는 멤버함수와 같이 사용
  - setw() , setprecision()

조작자 Example: setw()

- setw() 조작자:

```
cout        << "Start" << setw(4) << 10
            << setw(4) << 20 << setw(6) << 30;
```

  - 결과:

```
Start  10  20    30
```

- setw() 는 오직 다음에만 영향

조작자 setprecision()

- setprecision() 조작자:

```
cout.setf(ios::fixed | ios::showpoint);
cout    << "$" << setprecision(2) << 10.3 << "  "
        << "$" << 20.5 << endl;
```

- 결과:

$10.30 $20.50

플래그 값 저장

- 플래그 설정을 변경할 때까지 유지
- Precision, setf 프래그는 현재 설정을 저장하거나 원래 설정으로 복원가능
  - precision() 함수는 인수없이 호출되면 현재 설정 반환
  - flags() 멤버 함수도 유사한 기능 제공

플래그 값 저장 Example

- void outputStuff(ofstream& outStream)
  {
  int precisionSetting = outStream.precision();
  long flagSettings = outStream.flags();
  outStream.setf(ios::fixed | ios::showpoint);
  outStream.precision(2);
  outStream.precision(precisionSetting);
  outStream.flags(flagSettings);
  }

setf 기본값 복원 설정

- 기본값으로 복원 설정 가능:

  cout.setf(0, ios::floatfield);
- 디폴트 값은 플래그 값이 변경되기 직전의 설정값을 의미 하지 않음
- 디폴트 값은 컴파일러 구성에 따라 다름
- precision 설정 값을 복원 시키지 않음
  - Only setf settings

계층적 Stream 구조

- 클래스에서의 상속
  - 파생 클래스
    - 다른 클래스에서 얻은 하나의 클래스
    - 추가적인 기능 보유
  - Example:
  - 입력 파일 스트림 클래스는 모든 입력 스트림 클래스에서 파생
    - Open, close 멤버 함수 추가
  - i.e.: ifstream is derived from istream

상속의 실제 예

- 컨버터블은 자동차에서 파생
  - 모든 컨버터블은 자동차이다.
  - 컨버터블은 일반 자동차와는 다른 특성을 지닌다.

스트림 클래스 상속

- 클래스 D가 클래스B로부터 파생→
  - D형의 모든 객체는 B형 객체

- 스트림:
  - ifstream 객체는 istream 객체
  - istream 을 매개변수로 사용
    - 더 많은 객체에 연결 가능!

스트림 클래스 상속 Example

```
void twoSumVersion1(ifstream& sourceFile)//ifstream with an 'f'
{
    int n1, n2;
    sourceFile >> n1 >> n2;
    cout << n1 << " + " << n2 << " = " << (n1 + n2) << endl;
}
```

and

```
void twoSumVersion2(istream& sourceFile)//istream without an 'f'
{
    int n1, n2;
    sourceFile >> n1 >> n2;
    cout << n1 << " + " << n2 << " = " << (n1 + n2) << endl;
}
```

스트림 클래스 상속
Example Calls

- 이전 기능을 고려:
- twoSumVersion1(fileIn); // Legal!
- twoSumVersion1(cin); // ILLEGAL!
  - cin 는 ifstream형이 아님!
- twoSumVersion2(fileIn); // Legal!
- twoSumVersion2(cin); // Legal!
  - istream 인자는 둘다 사용 가능

파일 임의 접근

- 순차 접근
  - 일반적인 사용
- 임의 접근
  - 빠른 접근
  - 특별한 데이터 베이스 소프트웨어에 사용
  - 파일 임의의 부분에 접근
  - fstream 객체 사용
    - input and output

임의 접근 Tools

- Istream, ostream과 동일
  - 두번째 인자 필요
  - fstream rwStream;
    rwStream.open("stuff", ios::in | ios:: out);
    - 두 번째 인자 값은 입력,출력,입출력 모드인지 여부 알림
- 파일에 대한 이동
  - rwStream.seekp(1000);
    - 파일의 1000번째 바이트 위치
  - rwStream.seekg(1000);
    - 1000번째 위치의 포인터 값

Random Access Sizes

- 파일에 대한 이동 → 사이즈를 알아야 함
  - sizeof() 연산은 개체의 바이트 수 결정:
    sizeof(s) //Where s is string s = "Hello"
    sizeof(10)
    sizeof(double)
    sizeof(myObject)
  - 100번째의 MyStruct형의 레코드 부분의 포이터 값:

    rwStream.seekp(100*sizeof(MyStruct) – 1);

요약 1

- open 연산으로 스트림과 파일을 연결
- fail() 멤버함수로 파일 open 성공 유무 확인
- 스트림 멤부 함수 포맷 출력
  - e.g., width, setf, precision
  - cout (screen) , files 동일
- 함수는 스트림형의 매개변수 보유 가능
  - 반드시 call-by-reference

요약 2

- istream ("f“ 철자 없음) cin, ifstream 매개변수 사용가능
- ostream ("f” 철자 없음) cout,ofstream 매개변수 사용 가능
- Eof 멤버 함수
  - 입력 파일의 끝을 검사하는데 사용

Q&A