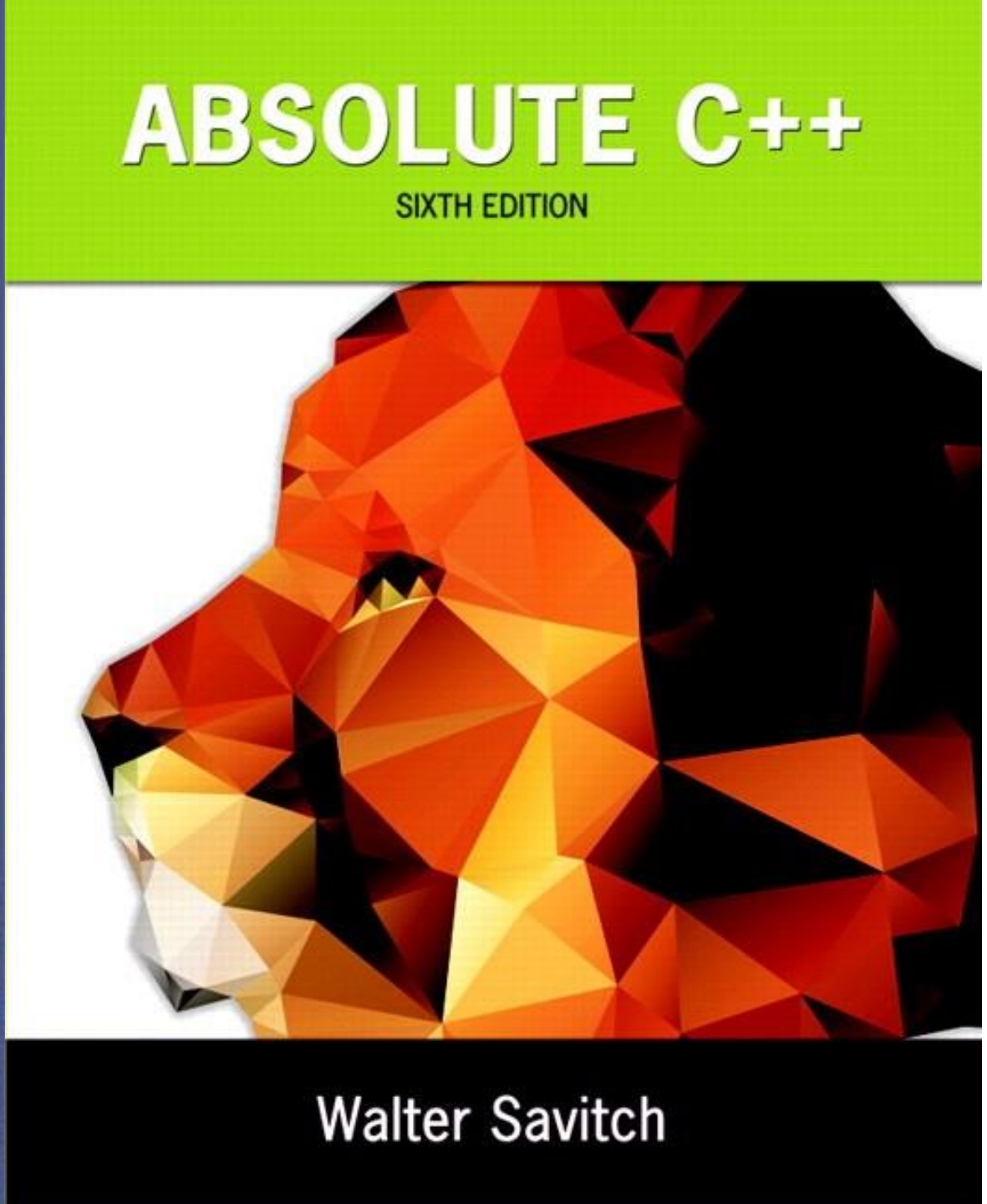

Chapter 12

Streams and File I/O

PEARSON

Introduction

- Streams
  - 프로그램 입력과 출력을 담당하는 특별한 객체들
- File I/O
  - 프로그램과 파일 사이의 입출력은 매우 유용함
  - 상속 매커니즘을 사용함

Streams

- 스트림은 문자들의 흐름을 의미함
- 입력 스트림
  - 키보드(cin) 또는 파일로부터 입력 스트림이 제공됨
  - 예시: int theNumber;
    inStream >> theNumber;
- 출력 스트림
  - 스크린(cout) 또는 파일로 출력 스트림이 제공됨
  - 예시: outStream << "theNumber is " << theNumber;

Files

- 여기서 text file 입출력을 다룸
- file 과 stream 객체를 먼저 연결해야 함
  - 입력 File → ifstream object
  - 출력 File → ofstream object
- Classes ifstream and ofstream
  - std 네임스페이스의 <fstream> 라이브러리에 정의됨

File I/O Libraries

- 라이브러리 사용법:

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

또는

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

Declaring Streams

- 스트림 객체를 다음과 같이 선언한다:

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

- 그리고나서 파일에 연결한다:

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

  - open 함수로 파일을 연다.
  - 연결하려는 파일의 완전한 경로이름을 명시해야 한다.

Streams Usage

- instream 객체가 파일에 연결되면 cin 처럼 사용함

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

- outstream 객체가 파일에 연결되면 cout 처럼 사용함

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

File Names

- 파일은 두 가지 이름을 갖는다.
  - 외부 파일 이름
    - "infile.txt"과 같은 물리적인 파일 이름
    - 진짜 파일 이름
    - open 할 때 딱 한 번 사용됨
  - 스트림 이름
    - stream 객체로 대변되는 논리적인 파일 이름
    - 프로그램 입장에서는 이 논리적인 이름을 모든 입출력에 사용함

Closing Files

- 파일 입출력이 끝나면 파일을 close 해야 함
  - 스트림 객체와 파일을 분리함
  - 예시:

```
inStream.close(); // 인자 없음
outStream.close(); // 인자 없음
```

- 파일 close 없이 프로그램이 종료하면, 파일은 자동적으로 close 됨

File Flush

- 출력은 종종 버퍼링된다 ("buffered")
  - 프로그램이 출력하는 데이터는 바이트 단위로 기록되지 않고 바이트 그룹 단위로 기록된다.
  - 바이트 그룹은 임시적으로 프로그램 안에 버퍼링된다.
- 바이트 그룹이 완성되지 않아도, 파일에 기록하는 것을 종종 강제할 필요가 생긴다:
  - outStream.flush(): buffered 상태의 모든 출력이 파일에 물리적으로 기록된다.
- 파일을 close 하면, flush() 가 자동으로 호출된다.

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

- 일반적으로 파일 조작은 빈 파일로 시작한다.
  - 파일 내용이 존재하면, 그 내용은 소실된다.
- append 를 위해서 파일 열기:

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

  - 파일이 존재하지 않으면 → 그 파일을 생성한다.
  - 파일이 존재하면 → 파일 끝에 새로운 내용을 추가한다.
  - 두번째 인자는 “ios” 클래스가 정의한 상수이다.
    - std 네임스페이스의 <iostream> 라이브러리

Alternative Syntax for File Opens

- 스트림 객체 생성자를 이용한 파일 열기
- 예: ifstream inStream;
  inStream.open("infile.txt");

  다음 한 줄과 동등하다:

  ifstream inStream("infile.txt");

Checking File Open Success

- 파일 열기는 실패할 수 있다.
  - 파일이 존재하지 않을 때
  - 출력용 파일을 열고자 하는데, 쓰기 권한이 없을 때
  - 기타
- 멤버 함수 fail()
  - 스트림 연산 직후에 호출하여 성공 여부를 확인한다.
  - 예시:

```
inStream.open("stuff.txt");
if (inStream.fail())
{
  cout << "File open failed.\n";
  exit(1);
}
```

Character I/O with Files

- cin 과 cout 에 사용되던 모든 문자 입출력은 파일 입출력에 동일하게 적용된다.
- 멤버 함수들
  - istream: get, getline, peek, ignore, ...
    - https://cplusplus.com/reference/istream/
  - ostream: put, putback, ...
    - https://cplusplus.com/reference/ostream/

Checking End of File

- 파일 처리시 파일 끝에 도달할 때까지 파일 처리하는 루프를 이용함
- 파일 끝에 도달했는지 테스트하는 두 가지 방법
  - 방법1: 멤버 함수 eof() 를 사용하기

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

    - eof() 함수는 bool 값을 리턴함

End of File Check with Read

– 방법2: read 연산이 리턴하는 bool 값 이용

- 예: (inStream >> next)
  - read 가 성공하면 true 리턴함
  - 파일 끝을 넘어 read 시도하면 false 리턴함
- 예시:

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

Formatting Output with Stream Functions

- 출력 포맷팅: 모든 출력 스트림에 적용 가능

  cout.setf(ios::fixed); // 소수점 자리수 고정
  cout.setf(ios::showpoint); // 소수점 출력
  cout.precision(2); // (소수점 이하) 두 자리 지정
  → 출력 예시 123.52, 1.00, 9.80, ...

More Output Member Functions

- 멤버 함수 precision(x)
  - 처음 “x” digits 를 출력함
- 멤버 함수 setf()
  - 다중 출력 플래그들을 허용함
  - 예: outStream.setf(ios::fixed | ios::showpoint);
- 멤버 함수 width(x)
  - 다음 출력이 “x”개의 칸을 차지할 것을 요구함
    - x = 10 이고 다음 출력 문자열의 길이가 6이면, 4개 빈칸이 삽입됨
  - 다음 출력에만 영향을 미치고, 그 다음 출력에는 영향을 미치지 않음

Manipulators

- 전통적이지 않은 방식으로 호출되는 함수들
  - 인자들을 가질 수 있음
  - << 연산자 다음에 놓임
  - 멤버 함수들과 동일한 동작을 (다른 방식으로) 수행함
- setw() 과 setprecision():
  - std 네임스페이스의 <iomanip> 라이브러리에 정의됨

Manipulator Example: setw()

- setw() manipulator:

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

  - Results in:

```
Start  10  20    30
```

- setw() 역시 다음 출력에만 영향을 미침

Manipulator setprecision()

- setprecision() manipulator:

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

- Results in:
  $10.30 $20.50

Saving Flag Settings

- 플래그 세팅은 다음 세팅을 만날 때까지 유지된다.
- Precision 과 setf 플래그들을 저장되고 복구될 수 있다.
  - precision() 함수를 인자 없이 호출하면 현재 세팅을 리턴한다.
  - flags() 함수를 인자 없이 호출하면 현재 세팅을 리턴한다.

Saving Flag Settings Example

- 예시:

```
void outputStuff(ofstream& outStream)
{
        int precisionSetting = outStream.precision(); // 세팅 저장
        long flagSettings = outStream.flags(); // 세팅 저장
        outStream.setf(ios::fixed | ios::showpoint); // 세팅 변경
        outStream.precision(2); // 세팅 변경
        outStream.precision(precisionSetting); // 세팅 복구
        outStream.flags(flagSettings); // 세팅 복구
}
```

Restoring Default setf Settings

- setf 에 한해 디폴트 세팅으로 복구하는 방법:

```
cout.setf(0, ios::floatfield);
```

- 디폴트 세팅은 c++ 구현마다 다름

Stream Hierarchies

- 상속 관계
  - 입력 파일 스트림 클래스 (ifstream) 는 입력 스트림 클래스(istream)로부터 파생되었다.
  - 출력 파일 스트림 클래스 (ofstream) 는 출력 스트림 클래스(ostream)로부터 파생되었다.

Stream Class Inheritance 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;
}
```

Stream Class Inheritance Example Calls

- 인자를 ifstream 으로 한정한 경우:
  - twoSumVersion1(fileIn); // Legal!
  - twoSumVersion1(cin); // ILLEGAL!
  - Because cin is not of type ifstream!
- 인자를 istream 으로 한정한 경우:
  - twoSumVersion2(fileIn); // Legal!
  - twoSumVersion2(cin); // Legal!

stringstream

- stringstream 클래스는 상속의 또다른 예
  - iostream 클래스를 상속함
  - 문자열에 대해서 스트림 연산들을 수행하는 것을 가능하게 함
  - 문자열을 또 다른 데이터 타입으로 변환하거나 반대의 동작을 수행할 때 유용함

Using stringstream

- To use

```
#include <sstream>
using std::stringstream;
```

- Create an object of type stringstream

```
stringstream ss;
```

- To clear and initialize to blank

```
ss.clear( );
ss.str("");
```

- To create a string from other variables

```
ss << c << " " << num;        // c is a char, num is an int
```

Using stringstream

- To extract variables from a string

```
ss << "x 10";
ss >> c >> num;
// c is set to 'x' and num is set to 10
```

stringstream Demo (1 of 3)

```
//Demonstration of the stringstream class.  This program takes
//a string with a name followed by scores.  It uses a
//stringstream to extract the name as a string, the scores
//as integers, then calculates the average score.  The name
//and average are placed into a new string.

#include <iostream>
#include <string>
#include <sstream>

using namespace std;

int main( )
{
        stringstream ss;
        string scores = "Luigi 70 100 90";
```

stringstream demo (2 of 3)

```
// Clear the stringstream
ss.str("");
ss.clear();

// Put the scores into the stringstream
ss << scores;

// Extract the name and average the scores
string name = "";
int total = 0, count = 0, average = 0;
int score;
ss >> name;                    // Read the name
while (ss >> score)  // Read until the end of the string
{
        count++;
        total += score;
}
```

stringstream demo (3 of 3)

```
        if (count > 0)
        {
                average = total / count;
        }

        // Clear the stringstream
        ss.clear();
        ss.str("");
        // Put in the name and average
        ss << "Name: " << name << " Average: " << average;

        // Output as a string
        cout << ss.str() << endl;

        return 0;
}
```

Random Access to Files

- 순차 접근
  - 가장 많이 사용됨
- 랜덤 접근
  - Database 레코드들을 빠르게 접근하고자 할 때 사용
  - 파일 어느 부분도 랜덤하게 접근할 수 있음
  - fstream 객체를 사용함

Random Access Tools

- istream/ostream 과 같은 방식으로 열기
  - fstream rwStream;
    rwStream.open("stuff", ios::in | ios:: out);
    - 읽기 권한과 쓰기 권한을 동시에 갖도록 열기
- 읽기 또는 쓰기를 수행할 파일 내부 위치를 옮기기
  - rwStream.seekp(1000);
    - put 포인터를 1000th byte 에 위치시킴
  - rwStream.seekg(1000);
    - get 포인터를 1000th byte 에 위치시킴

Random Access Sizes

- 파일 내부 위치를 이동하려면, 레코드 크기를 정확히 알아야 함
  - sizeof() 연산자로 객체의 크기를 확인해야 함:
    sizeof(s) //Where s is string s = " Hello "
    sizeof(10)
    sizeof(double)
    sizeof(myObject)
  - put 포인터를 100th record 에 위치시키기:
    rwStream.seekp(100*sizeof(myObject) – 1);