제12장

스트림 및 파일 입/출력

학습 목표

- 입/출력 스트림
  - 파일 입/출력
  - 문자 입/출력
  - 스트림 입/출력 도구들
  - 파일 이름을 통한 입력
  - 출력 형식, 플래그 설정
- 계층적 스트림 구조
  - 상속 개념 엿보기
- 파일의 임의 접근

스트림과 스트림 클래스 사용법

- 스트림: 문자들의 흐름
  - 입력 스트림
    - 컴퓨터 키보드/파일 를 통하여 데이터가 입력될 수 있음
  - 출력 스트림
    - 데이터가 프로그램으로부터 밖으로 향함

- 이미 스트림 클래스의 기능을 사용하였음
  - cin: 키보드를 연결시키는 입력 스트림 오브젝트
  - cout: 화면을 연결시키는 출력 스트림 오브젝트
- 이외에도 다른 종류의 스트림을 정의할 수 있음
  - 데이터를 파일로 향하게 하거나 또는 파일에서 밖으로 향하게 할 수 있음
  - cin, cout 객체의 기능과 유사함

```
49  ostream& operator <<(ostream& outputStream, const Money& amount)
50  {
51      int absDollars = abs(amount.dollars);
52      int absCents = abs(amount.cents);
53      if (amount.dollars < 0 || amount.cents < 0)
54          //dollars ==0과 cents ==0에 대한 계산을 위해
55          outputStream << "$-";
56      else
57          outputStream <
58      outputStream << abs
59      if (absCents >= 10)
60          outputStream <
61      else
62          outputStream <
63      return outputStrea
64  }
```

main 함수에서 cout이 outputStream에 대해 들어오게되었다.

```
66  //iostream과 cstdlib 사용:
67  istream& operator >>(istream& inputStream, Money& amount
68  {
69      char dollarSign;
70      inputStream >> dollarSign; //희망을 가지고
71      if (dollarSign != '$')
72      {
73          cout << "No dollar sign in Money input.\n";
74          exit(1);
75      }
76      double amountAsDouble;
77      inputStream >> amountAsDouble;
78      amount.dollars = amount.dollarsPart(amountAsDouble);
79      amount.cents = amount.centsPart(amountAsDouble);
80      return inputStream;
81  }
```

main 함수에
inputStre

뒤에 연산자가 아니
뒤에 함수에 대해

참조 리턴

cin, cout 과 유사한 스트림의 입출력 기능

- 다음과 같은 사항을 고려하면:
  - 파일로부터 입력을 수행하는 inStream 스트림을 정의한 프로그램을 다음과 같이 고려하면:

    ```
    int theNumber;
    inStream >> theNumber;
    ```

    - 여기서 스트림으로부터 데이터를 읽어서 theNumber 에 저장함
  - 파일에 데이터를 저장하는 outStream 스트림을 다음과 같이 고려하면:

    ```
    outStream << "theNumber is " << theNumber;
    ```

    - 스트림에 데이터가 향하고 이 값은 파일에 저장됨

파일

- 텍스트 파일을 사용함
- 파일로부터 데이터를 읽기
  - 데이터를 입력하는 프로그램을 작성하는 경우
- 파일에 데이터를 저장하기
  - 데이터를 출력하는 프로그램을 작성하는 경우
- 파일의 처음부터 끝 부분까지 데이터를 읽기 수행
  - 이외 다른 방법도 있지만 여기서는 텍스트 파일을 읽는 방법으로 간단하게 설명함

파일 연결하기와 파일 입출력 라이브러리

- 파일연결하기: 우선 파일을 스트림 객체에 연결해야 됨
  - 입력의 경우: 파일 → ifstream 오브젝트
  - 출력의 경우: 파일 → ofstream 오브젝트
  - ifstream 및 ofstream 클래스
    - <fstream> 라이브러리에서 정의됨
    - std 네임스페이스를 명시해야 됨

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

스트림 선언

- 스트림을 사용하려면 다른 클래스와 같이 사전에 정의되어야
  - 클래스 변수:

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

- 파일에 반드시 " 연결" 되어야 함:

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

  - open 멤버 함수 호출. 파일 경로를 명시할 수 있음

- 스트림이 선언된 다음
  → 사용하면 됨!

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

- 데이터를 출력 파일에 전송함

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

파일 닫기

- 파일은 사용 후 반드시 닫아야 됨
  - 입력 또는 출력 작업이 완료된 후에 파일을 닫아야 하는 프로그램을 작성하는 경우
  - 스트림을 파일로부터 연결을 해제해야 됨
  - 예:

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

    - Close 함수에 인자를 사용하지 않음에 유의
- 프로그램이 종료될 때 파일은 자동적으로 닫힘

파일의 데이터 비우기 (Flush)

- 출력 데이터를 버퍼에 " 임시로 저장 " 하는 경우들이 많음
  - 데이터를 파일에 저장하기 전에 임시로 버퍼에 저장하는 경우를 말함
  - "groups“ 형태로 쓰기 작업
- 강제적으로 쓰기 작업이 실행되는 경우들이 많음:

  outStream.flush();
  - 출력 스트림의 flush 멤버 함수를 사용함
  - 인위적인 쓰기 작업을 통하여 버퍼에 데이터가 저장됨
  - 파일이 닫히면 자동적으로 flush() 함수가 실행됨

파일 입/출력 예제: 디스플레이 12.1 간단한 파일 입/출력 (1 of 2)

디스플레이 12.1 간단한 파일 입/출력

```
1   //infile.txt 파일로부터 3개의 숫자를 읽은 뒤에 합을 계산하고
2   //그 합을 outfile.txt 파일에 기록한다.
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;
```

보다 개선된 버전이 디스플레이 12.3에서 다시 소개된다.

파일 입/출력 예제:
디스플레이 12.1 간단한 파일 입/출력 (1 of 2)

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

21          return 0;
22      }
```

Sample Dialogue

출력 화면에 아무런 내용이 표시되지 않으며
키보드를 통한 입력이 없는 경우

Infile.txt
(프로그램 실행 결과에 영향을 받지 않음)

```
1
2
3
4
```

Outfile.txt
(프로그램 실행 후)

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

기존 파일에 데이터 첨부하기

- 파일은 일반적으로 덮어쓰기로 작성
  - 파일을 여는 과정은 보통 데이터가 저장되어 있지 않은 빈 파일로부터 시작하며, 데이터가 저장된 파일을 열면 데이터가 모두 사라짐
- 기존 파일 끝에 데이터를 추가하고 싶으면:

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

  - 파일이 존재하지 않으면 → 새로운 파일이 만들어 짐
  - 파일이 존재하면 → 데이터가 이 파일에 첨부됨
  - 두 번째 인자는 class ios 클래스에서 정의된 상수임
    - <iostream> 라이브러리에, std 네임스페이스를 명시해야 됨

파일 열기 작업의 또 다른 방법

- 파일을 선언하면서 동시에 파일 이름을 명시할 수 있음
- 즉 생성자 함수에 인자 형태로 이름을 사용하면 됨
- ifstream inStream;

  inStream.open("infile.txt");

  위의 코드는 다음의 코드와 동일함:

  ifstream inStream("infile.txt");

파일 열기 작업 성공여부 판별

- 다음의 경우들 파일 열기 작업이 실패할 수 있음I
  - 입력 파일이 존재하지 않거나
  - 출력 파일에 쓰기 작업이 허락되지 않거나
  - 또는 예상치 않은 결과가 나오는 경우
- fail() 멤버 함수를 사용하여 확인함
  - 스트림 작업의 성공 여부를 확인하기 위해서 fail() 함수를 사용함

예: inStream.open("stuff.txt");

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

파일을 통한 문자 입/출력

- cin 및 cout 객체를 이용하여 문자 입/출력은 파일 입/출력 과정과 동일함!
- 다음과 같은 멤버 함수들을 사용함:
  - get (https://modoocode.com/191)
  - getline(https://modoocode.com/236, https://modoocode.com/149)
  - put(http://www.cplusplus.com/reference/ostream/ostream/put/)
  - putback(http://www.cplusplus.com/reference/istream/istream/putback/)
  - peek(https://modoocode.com/194)
  - ignore(https://modoocode.com/193)

파일의 끝 부분 확인하기

- 방법1: eof() 멤버 함수 사용
  - 파일 끝부분이 나올 때까지 문자를 읽음
    - eof() 멤버 함수의 리턴 값은 bool 형임

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

- 방법 2: 읽기 작업의 성공/실패를 확인
  - 읽기 작업을 수행하는 경우 bool 값을 리턴 한다는 사실을 활용함!
  - (inStream >> next)
    - 읽기 작업이 성공하면 true 값이 리턴 됨
    - 파일의 끝 부분을 지나쳐서 데이터를 읽으면 false 값이 리턴 됨
  - 예:

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

도구: 파일 이름을 통한 입력

- 스트림을 이용한 읽기
  - open() 함수의 인자는 string 형
  - 인자는 문자상수 (지금까지 사용한 방식) 또는 변수임

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

- argv 활용

```
int main(int argc, char *argv[])
{
    ifstream inStream;
    if (argc != 2) return 1;
    inStream.open(argv[1]);
```

스트림 함수를 이용한 출력 형식 설정

- 1 장의 내용에서 다루었던 다음과 같은 " 마술 공식 "

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

  - 숫자를 출력하는 형식은 " 화폐" 를 표현하는 방식을 사용할 수 있음 (예:12.52)
  - 임의의 출력 스트림을 사용할 수 있음
    - 파일 스트림은 cout 객체가 동일한 멤버 함수들을 사용함

- 다음의 코드를 고려하면:

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

  - 멤버 함수가 표현하는 정밀도(x)
    - 여기서 "x" 는 소수점 이하의 개수를 말함
  - setf() 멤버 함수
    - 플래그 설정을 통한 출력문 형식 결정

추가 출력 멤버 함수와 플래그

- 멤버 함수의 width (x)

```
outStream.width(5);
```

  - width를 사용하여 출력 항목이 차지하는 공간의 크기를 설정함
    - width 함수 설정 이후에 오는 코드에만 적용 됨

- setf() 멤버 함수를 다시 한번 살펴보면:
  - 출력 플래그의 상태를 설정. 모든 출력 스트림은 setf() 멤버 함수를 보유
  - ios 클래스는 몇몇 개의 중요한 플래그 값을 가지고 있음
  - <iostream> 라이브러리에 포함되어 있고, std 네임스페이스를 명시해야 됨
  - 흔히 사용되는 플래그 상수들:
    - outStream.setf(ios::fixed); // 고정 소수점 표기법 설정 (십진법)
    - outStream.setf(ios::showPoint) //항상 소수점을 포함시킴
    - outStream.setf(ios::right); //오른쪽 열로 맞춤
  - 한번의 함수 호출을 통하여 여러 개의 플래그 값을 동시에 설정 가능
    - outStream.setf(ios::fixed | ios::showpoint |ios::right);

조작기

- 조작기의 정의: " 기존의 함수와는 다른 방법으로 호출됨"
  - 인자를 가질 수 있음
  - 삽입 연산자 다음에 위치함
  - 멤버함수와 동일한 일을 수행함!
    - 다름 방법으로 동일한 일을 수행함
  - 조작기와 멤버함수를 같이 사용함
- setw() 및 setprecision() 조작기는 라이브러리에 정의됨
  - <iomanip>를 포함시키고, std 네임스페이스를 명시해야 됨

조작기 사용 예제: setw(), setprecision()

- setw() 조작기:

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

  - 실행 결과:

```
Start  10  20    30
```

  - 주의: setw() 조작기가 선언된 다음 위치부터 출력 형식이 작동
    - 출력 형식 설정에 활용하기 전에 먼저 setw() 함수를 포함시켜야 함

- setprecision() 조작기 사용 예:

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

  - 실행 결과:

```
$10.30  $20.50
```

플래그 값 저장

- 한번 설정된 플래그 값은 " 변하지 않음"
- setprecision and setf 조작기를 통하여 설정된 플래그는 저장될 수 있고 나중에 값을 읽을 수 있음
  - precision() 조작기는 인자없이 호출되면 현재 설정된 값을 리턴 함
  - flags() 멤버 함수도 비슷한 기능을 가지고 있음

```
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);
}
```

- Function to save & restore "typical" settings
  - Call: outputStuff(myStream);

저장된 플래그 값 읽기

- 디폴트로 설정된 플래그 값 읽기:

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

- 플래그의 디폴트 값은 변경될 수 있음!
- 플래그의 디폴트 값은 프로그래머가 사용하는 컴파일러의 구성에 따라 달라짐
  - setf 함수를 사용하지 않고 설정된 값 또는 precision 설정 값을 원래대로 복원시키지 않음

계층적 스트림 구조

- 클래스 상속 관계
  - " 파생 관계"
    - 한 클래스가 다른 클래스로부터 파생됨
    - 파생된 클래스에 함수를 " 추가함"
  - 예:
  - 입력 파일 스트림 클래스는 모든 입력 스트림 클래스에서 파생됨
    - 그리고 입력 파일 스트림 클래스에 open 및 close 멤버 함수를 추가함
    - 예: ifstream 클래스는 istream 클래스에서 파생됨
  - 스트림 클래스와 연관하여:
    - ifstream 오브젝트는 istream 오브젝트임
    - 매개변수 형식의 istream 오브젝트 사용
      - 이런 경우 여러 오브젝트들을 매개변수로 사용 가능함!

Stringstream 클래스

- stringstream 클래스는 상속의 개념을 사용한 또 다른 예
  - Iostream 클래스에서 파생됨
  - 스트림을 사용하여 string 읽기 또는 쓰기가 가능함. 이러한 기능은 cin을 활용한 스트림 작업과 유사함
    - 동일한 방법을 공유하거나 아니면 상속이 가능함
- String을 다른 유형의 데이터로 변환하거나 또는 다른 유형의 데이터를 string으로 변환하는데 유용함
  - 외부 자료로부터 string 값을 읽거나 특정한 필드를 검출하고자 할 때 이 클래스를 사용하면 편리함

Stringstream 클래스 사용법

- Strinstream을 사용하기 위해서는

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

- Stringstream 형의 오브젝트 생성

```
stringstream ss;
```

- clear 함수를 실행하고
  여백으로 초기화

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

- 다른 변수들을 이용하여
  string을 만듬
- String으로부터 변수 값을 얻음

```
ss << c << " " << num;
// c는 문자 형, num은 int 형
```

```
ss << "x 10";
ss >> c >> num;
// c는'x' 그리고 num은 10을 저장함
```

stringstream 클래스 활용 예제

```
// stringstream 클래스 사용법의 예.  사람의 이름 및 점수를 string에 저장함
// stringstream 클래스를 사용하여 string에 저장된 이름과 점수를
// 정수 형태로 검출한 다음에, 평균 점수를 계산함.  The name
// 검출된 이름과 평균 점수는 새로운 string에 저장됨.
#include <iostream>
#include <string>
#include <sstream>

using namespace std;

int main( )
{
        stringstream ss;
        string scores = "Luigi 70 100 90";
        // stringstream clear 함수 실행
        ss.str(""); ss.clear();

        // stringstream에 점수 저장
        ss << scores;

        // 이름과 평균 점수를 검출함
        string name = "";
        int total = 0, count = 0, average = 0;
        int score;
        ss >> name;                          // 이름 읽기
        while (ss >> score)  // string 끝 부분까지 읽음
        {
                count++;
                total += score;
        }
        if (count > 0) average = total / count;

        // stringstream clear 함수 실행
        ss.clear(); ss.str("");
        // 이름과 평균 점수 표시
        ss << "Name: " << name << " Average: " << average;

        // string 형태로 출력함
        cout << ss.str() << endl;

        return 0;
}
```

파일 임의 접근

- 순차적 접근 방법
  - 가장 많이 사용되는 방법
- 임의 접근
  - 특정한 레코드에 신속히 접근함
  - 대규모의 데이터베이스 접근에 사용됨
  - 파일의 특정한 부분을 " 임의 " 로 접근할 수 있다는 의미 임
  - fstream 오브젝트를 사용함
    - 파일의 임의 부분을 읽기와 쓰기에 활용

파일 임의 접근을 위한 도구

- istream 또는 ostream 클래스와 유사하게 open 함수 사용
  - 두 번째 인자가 첨가됨
  - fstream rwStream;

    rwStream.open("stuff", ios::in | ios:: out);
    - 읽기 및 쓰기 작업 가능
  - 파일의 측정한 부분의 위치 선정
  - rwStream.seekp(1000);
    - 파일의 1000 번째 바이트 부분에 rwStream을 연결함
  - rwStream.seekg(1000);
    - 파일의 1000 번째 바이트 부분에 포인터를 연결함

임의 접근을 위한 파일의 크기 파악

- 파일의 특정부분에 임의로 접근하려면 → 파일 크기를 알아야
  - sizeof() 바이트 단위로 크기를 알려주는 함수

    오브젝트의 크기 파악을 위해서 필요한 함수임:

```
sizeof(s) //여기서 s는 string이고 "Hello“ 값이 저장됨
sizeof(10)
sizeof(double)
sizeof(myObject)
```

  - 오브젝트의 100 번째 레코드에 포인터를 연결시킴:

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

요약

- 스트림 클래스 함수
  - open 함수 실행을 통하여 파일과 연결됨
  - fail() 함수 실행을 통하여 파일 열기 성공 여부를 판단 할 수 있음
  - eof() 함수 실행을 통해 입력 파일의 끝 부분에 도달하였는지 여부를 확인
  - 출력 형태 설정(예, width, setf, precision)
    - cout 객체를 이용해서 (스크린) 또는 파일에 출력하는 방법과 유사함
- 스트림 유형의 오브젝트는 함수의 매개변수로 사용될 수 있음
  - 이 경우 반드시 call-by-reference 형식으로 해야 됨
- istream ("f" 철자가 없음) / ostream ("f" 철자가 없음)
  - 매개변수에 대응되는 인자로서 cin 스트림 또는 ifstream 형의 입력 파일 스트림 / cout 스트림 또는 ofstream형의 출력스트림 사용 가능
- 스트림 클래스는 상속을 통하여 멤버 함수들을 함께 사용
  - 이런 경우 파생된 클래스는 부모의 클래스에 대하여 “is-a” 관계가 성립