입출력 시스템

202004103 정원석, 202004904 이종하

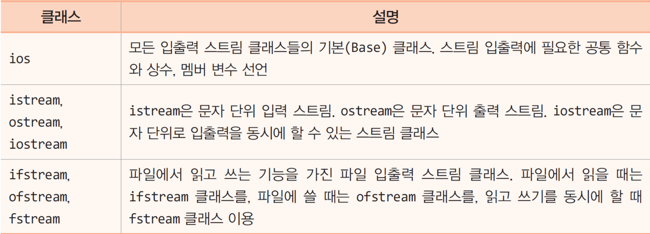
1 )입출력 기초  
스트림(stream) – 데이터의 흐름, 혹은 데이터를 전송하는 소프트웨어 모듈이다.  
스트림의 종류에는 입력 스트림과 출력 스트림으로 나뉜다. 입력 스트림은 데이터를 프로그램으로 전달하는 스트림이고 출력 스트림은 프로그램에서 출력되는 데이터를 외부로 전달하는 스트림이다.

C++ 입출력 스트림은 버퍼를 가진다.  
키 입력 스트림의 버퍼의 목적은 입력장치로부터 입력된 데이터를 프로그램으로 전달하기 전에 잠시 저장하고 키 입력 도중에 수정이 가능하다.

스크린 출력 스트림 버퍼의 목적은 프로그램에 출력된 데이터를 출력 장치로 보내기 전에 잠시 저장하고 출력 장치를 반복적으로 사용하는 비효율성을 개선한다. 버퍼가 꽉 차거나 강체 출력 명령 시에 출력 장치에 출력된다.

입출력 방식은 2가지가 있는데 스트림 입출력 방식과 저 수준 입출력 방식이다.  
스트림 입출력 방식은 스트림 버퍼를 이용한 입출력 방식이고 입력된 키를 버퍼에 저장하고 출력되는 데이터는 일차적으로 스트림 버퍼에 저장한다.  
저 수준 입출력 방식은 키가 입력되는 즉시 프로그램에 키 값을 전달하고 프로그램이 출력하는 즉시 출력 장치에 출력하고 컴파일러마다 다른 라이브러리나 API를 지원한다.  
C++ 표준은 스트림 입출력 방식만 지원하여 모든 표준 C++ 컴파일러에 의해 컴파일 되고 높은 호환성을 가진다.

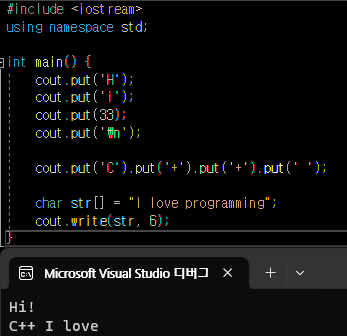
C++ 입출력 라이브러리는 문자를 한 바이트의 char로 처리하고 cin으로 문자를 읽을 때 한글 문자 단위로는 읽을 수 없다. 그 이유는 영어나 기호는 1바이트의 문자 코드이지만 한글 문자는 2바이트의 문자 코드이기 때문이다.





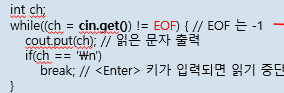
cin > istream 타입의 스트림 객체로서 키보드 장치와 연결됨  
cout > ostream 타입의 스트림 객체로서 스크린 장치와 연결됨  
cerr > ostream 타입의 스트림 객체로서 스크린 장치와 연결되고 오류 메세지를 출력할 목적을 가지고 스트림 내부 버퍼를 거치지 않고 출력된다.  
clog > ostream 타입의 스트림 객체로서 스크린 장치와 연결되고 오류 메시지를 출력할 목적은 cerr과 동일하나 스트림 내부에 버퍼를 거쳐 출력한다는 차이점이 있다.

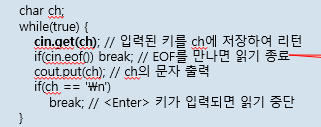
예제 11 – 1) ostream 멤버 함수를 이용한 문자 출력



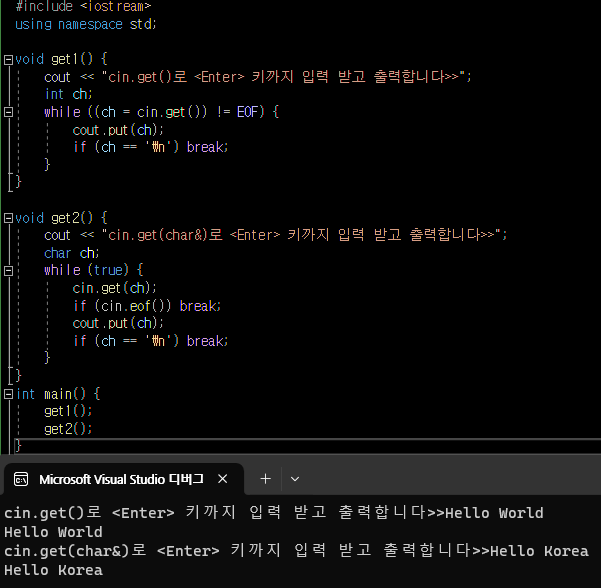
ostream의 멤버 함수로는 put(char ch) , write(char\* str, int n), flush()가 있다.

Istream) 멤버 함수 – 문자 입력, get() 함수

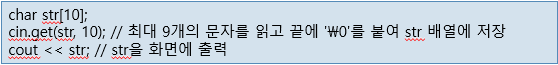
int get() – 입력 스트림에서 문자를 읽어 리턴, 오류나 EOF를 만나면 -1(EOF) 리턴한다.  
  
istream& get(char\* ch) – 입력 스트림에서 문자를 읽어 ch에 저장하고 현재 입력 스트림 객체(\*this)의 참조 리턴한다. EOF를 만나면, 스트림 내부의 오류 플래그를 세팅한다.



예제 11 – 2) get()과 get(char&)을 이용한 한 줄의 문자 읽기

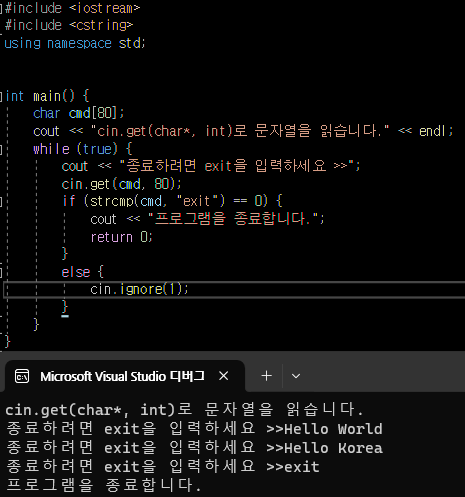


문자열 입력)  
istream& get(char\* s, int n) 입력 스트림으로부터 n-1 개의 문자를 읽어 배열 s에 저장하고 마지막에 ‘\0’ 문자 삽입. 입력 도중 ‘\n’을 만나면 ‘\0’을 삽입하고 리턴한다.



get()이 읽는 도중 <Enter>키(‘\n’)을 만나면 읽기를 중단하고 리턴한다. 이때 <Enter>키(‘\n’)는 스트림 버퍼에 남아 있어 다시 get()으로 문자열 읽기를 시도하면 입력 스트림에 s남은 ‘\n’키를 만나게 되어 리턴한다. 그래서 프로그램이 무한 루프에 빠질 수 있다.  
이때 해결하기 위해서는 cin.get()이나 cin.ignore(1);를 통해 ‘\n’를 스트림에서 읽어 버려야 한다.

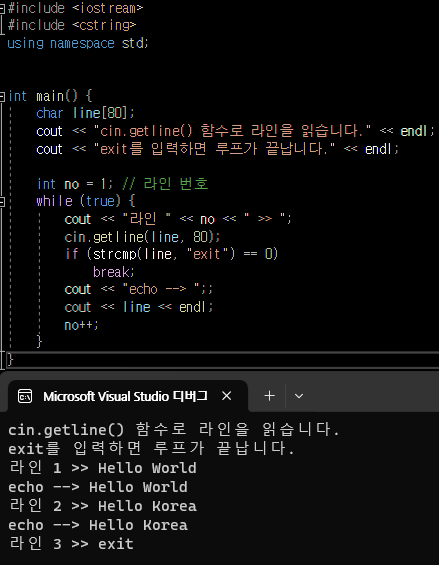
예시)



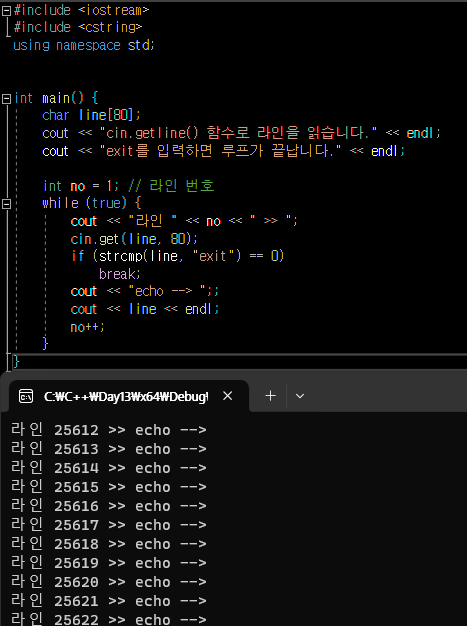
한 줄 읽기)  
istream& get(char\* s. int n, char delim =’\n’) – 입력 스트림으로부터 최대 n-1개의 문자를 읽어 배열 s에 저장하고 마지막에 ‘\n’ 문자를 삽입한다. 입력 도중 delim에 지정된 구분 문자를 만나면 지금까지 읽은 문자를 배열 s에 저장하고 리턴한다.

Istream& getline(char\* s, int n, char delim= ‘\n’) – get()과 동일, 하지만 delim에 지정된 구분 문자를 스트림에서 제거한다.

예제 11-4) Getline의 경우 – 정상적으로 작동된다.



get()의 경우 – 첫번째 버퍼가 ‘\n’로 남아있어 무한 반복된다.



입력 문자 건너 띄기와 문자 개수 알아내기)

istream& ignore(int n=1, int delim = EOF) – 입력 스트림에서 n개 문자 제거, 도중에 delim 문자를 만나면 delim 문자를 제거하고 리턴한다.



Int gcount() – 최근에 입력 스트림에서 읽은 문자 바이트 수(문자의 개수) 리턴한다. <Enter>키도 개수에 포함한다.



포맷 입출력)  
C++에서도 입출력 시 포맷 지정 가능하다.  
방법으로는 포맷 플래그, 포맷 함수, 조작자가 있다.

포맷 플래그)  
입출력 스트림에서 입출력 형식을 지정하기 위한 플래그이다.

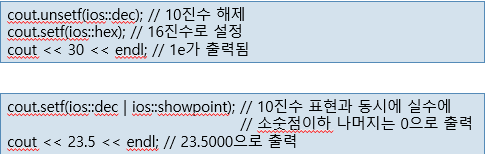




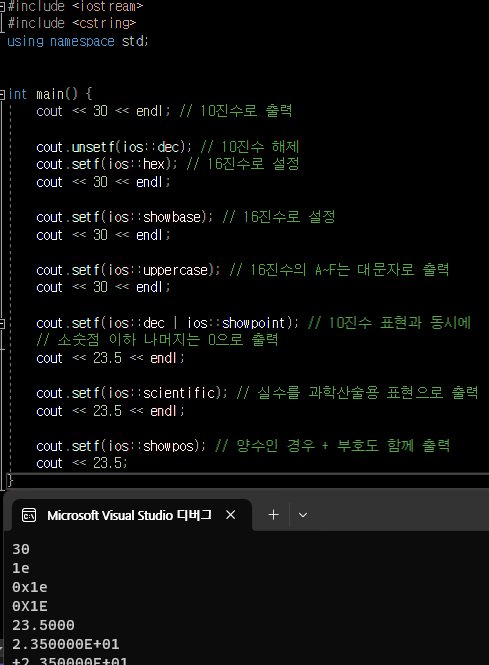
<세팅하는 멤버 함수>

Long setf(Long flags) – flags를 스트림의 포맷 플러그로 설정하고 이전 플래그를 리턴한다.

Long unsetf(Long flags) – flags에 설정된 비트 값에 따라 스트림의 포맷 플래그를 해체하고 이전 플래그를 리턴한다.



예제 11-5)

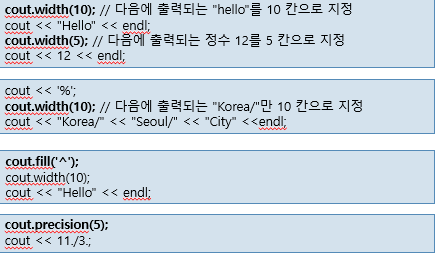


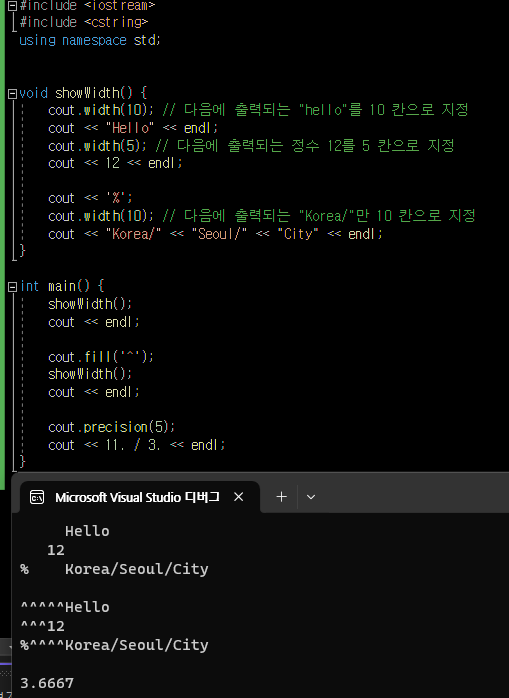
포맷 함수 활용)

Int width(int minWidth) – 출력되는 필드의 최소 너비를 midWidth로 설정하고 이전에 설정된 너비 값 리턴한다.

Char fill(char cFill) – 필드의 빈칸을 cFill 문자로 채우도록 지정하고 이전 문자 값 리턴한다.

Int precision(int np) – 출력되는 수의 유효 숫자 자리수를 np개로 설정하고 정수 부분과 소수점 이하의 수의 자리를 모두 포함하고 소수점은 제외한다.

  
예제 11-6)



조작자) - manipulator, 스트림 조작자(stream manipulator)

조작자는 함수로 C++ 표준라이브러리에 구현된 조작자는 입출력 포맷 지정 목적으로 사용하고 개발자 만의 조작자를 작성 가능하다. 그리고 매개 변수 없는 조작자와 매겨 변수를 가진 조작자로 구분한다.

조작자는 항상 <<나>> 연산자와 함께 사용됨



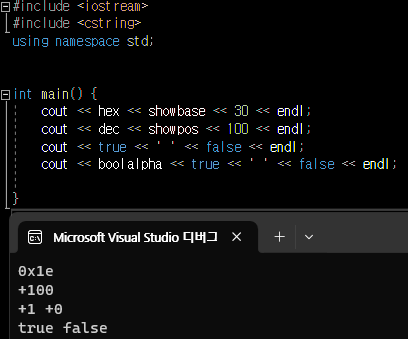
매개 변수가 없는 조작자



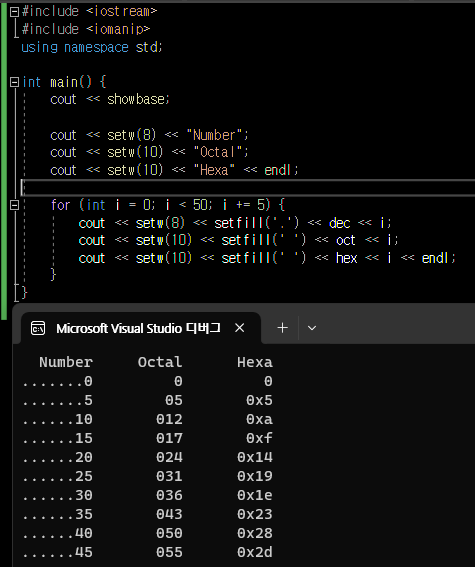
매개 변수를 가진 조작자



예제 11 – 7) 매개 변수가 없는 조작자



예제 11 – 8 ) 매개 변수가 있는 조작자



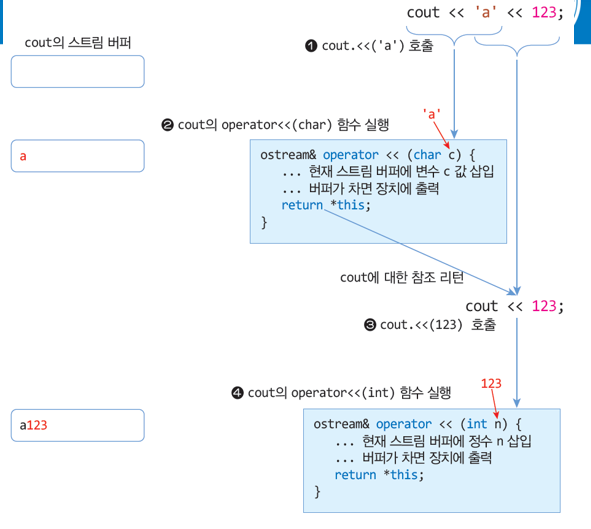
이렇게 조작자를 사용하면 출력하는 포맷을 다양하게 지정할 수 있다.

삽입 연산자(<<) 와 추출 연산자 (>>)

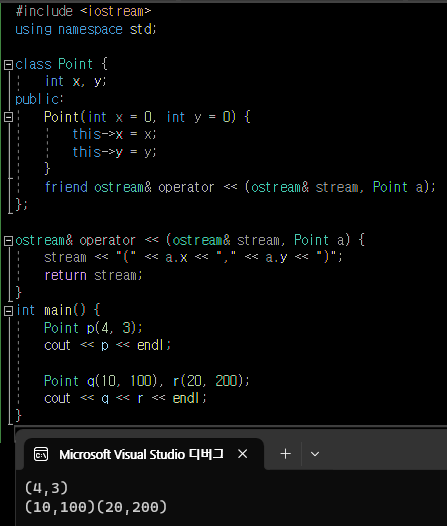
<삽입 연산자>

Insertion operator, 삽입자라고 부르고 <<연산자는 C++의 기본 연산자인 정수 시프트 연산자이다.

(실행과정)

예제 11 – 9)

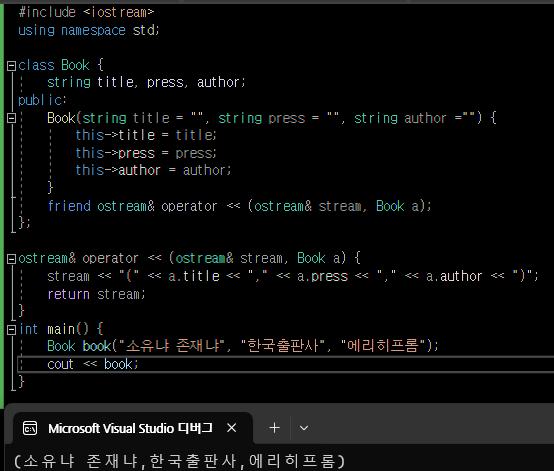




삽입 연산자만으로는 Point를 출력할 수 없어 포인트도 출력 가능한 연산자로 만들었다.

프랜드로 선언하여 Point에 있는 x, y를 접근 가능하게 만들었다.

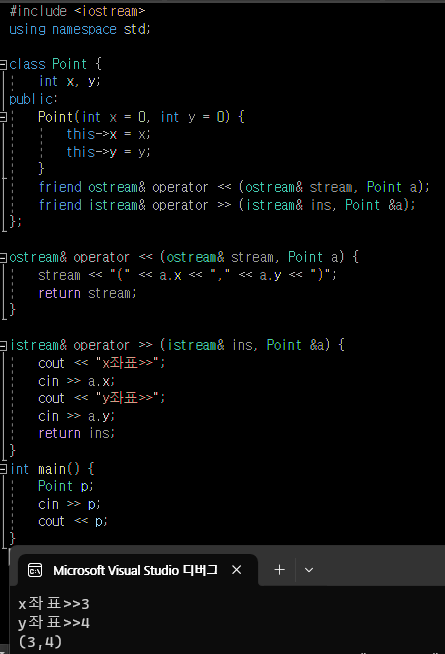
예제 11 -10)



<추출 연산자>

Extraction operator,>>연산자는 C++의 기본 연산자인 정수 시프트 연산자이다.  
과정은 삽입 연산자와 유사함

예제 11-11)



예제 11- 12, 11- 13) 사용자 정의 조작자 만들기

