11.1 C++ 입출력 기초

스트림 개념 스트림 흐르는 시냇물

연속적인 데이터의 흐름 혹은 데이터를 전송하는 소프트웨어 모듈

입력스트림 입력된 데이터를 순서대로 프로그램에 전달하는 객체

C++ 표준 입력 스트림객체 cin

출력스트림 프로그램에서 출력한 데이터를 목적 장치로 순서대로 보내는 객체

C++ 표준 출력 스트림객체 cout

스트림 스트림 버퍼 입력스트림버퍼 입력된 데이터를 프로그램에 전달하기 전에 일시 저장하는 공간

출력스트림버퍼 출력 장치로 보내기 전에 데이터를 일시 저장하는 공간

키 입력 스트림 버퍼 cin입력 스트림 <backspace>키 = 제어키

<enter>키 입력 = 프로그램 전달

스크린 출력 스트림 버퍼 cout출력 스트림 ‘\n’이 도착하거나 버퍼가 꽉 찰 때 스크린에 출력

cout.flush() 출력 버퍼에 있는 내용을 모두 장치에 출력

버퍼의 필요성 운영체제API를 이용해 입출력 장치, 프로그램 사이애서 데이터 전송

운영체제 API를 많이 호출하면 호출할수록 효율 하락

효율 상승을 위해 버퍼가 필요함

C++표준은 스트림 입출력만 지원 저수준 입출력 방식 버퍼를 가지지 않는 저수준 입출력 방식

키가 즉시 프로그램에 전달

C++ 컴파일러마다 특별한 저수준 입출력 방식이 사용됨

C++에선 스트림 입출력만 다룸

C++ 입출력 라이브러리 템플릿을 이용해 입출력 라이브러리 작성

다국어 사용을 위해 여러 바이트로 구성되는 문자 사용 가능

2003년 이전 문자를 한 바이트로만 다룸 🡪 문자가 2바이트인 한글문자 입력 불가

라이브러리 ios, istream, ostream, iostream, ifstream, ofstream, fstream

현재 템플릿을 통한 일반화 🡪 문자가 2바이트인 한글문자 입력 가능

라이브러리 basic\_ios, basic\_istream, basic\_ostream, basic\_iostrea,…

ios 모든 입출력 스트림 클래스의 기본 클래스

istream 문자 단위 입력 스트림

ostream 문자 단위 출력 스트림

iostream 문자 단위 입출력 스트림

ifstream 파일에서 읽을 때

ostream 파일에 쓸 때

fstream 파일에 읽고쓸 때

C++ 표준 입출력 스트림 객체 cin 키보드 장치와 연결된 istream 타입의 표준 입력 스트림 객체

cout 스크린 장치와 연결된 ostream 타입의 표준 출력 스트림 객체

cerr, clog cerr와 clog객체 는 둘다 표준 오류 출력 스트림 객체

cerr 버퍼 거침

clog 버퍼 안 거침

Quiz 1. 3

2. 3

11.2 ostream의 멤버함수를 이용한 문자 출력

<<연산자 외 화면출력을 위한 멤버함수 제공

put() 문자단위로 출력하는 함수

형식 cout.put(‘문자’)

리턴 ostream&

ex) cout.put('a');

cout.put(33);

cout.put('c').put('+').put('+');

write() char배열에 들어있는 문자들을 출력하는 함수

형식 cout.write(배열이름, 출력하는문자갯수

ex) char str[] = "i love programming";

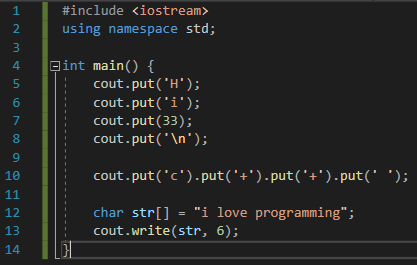
cout.write(str, 6);

flush() 출력버퍼에 있는 문자를 강제로 출력

형식 cout.flush()

ex) cout.put('a');

cout.flush(); //스트림버퍼에 있는 문자 'a'를 강제출력

예제11-1 

11.3 istream의 멤버함수를 이용한 문자 입력

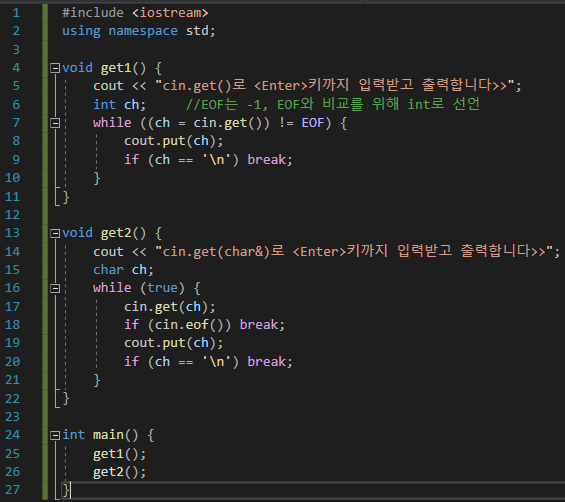
>>연산자 공백문자를 읽을 수 없음

istream 멤버함수 공백문자를 읽을 수 있음

문자 입력 int get() 1개의 문자(공백문자 포함)를 읽어 리턴

istream& get(char& ch) 입력스트림에서 문자를 읽어 ch에 저장하고 리턴

공백문자를 읽어 리턴 가능

예제11-2 

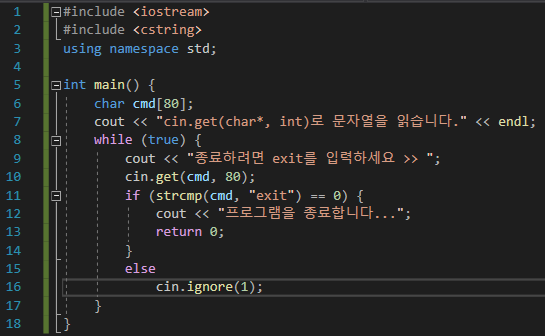
문자열 입력 istream& get(char\* s, int n) n-1개의 문자를 읽어 배열 s에 저장, 마지막에 ‘\0’문자 삽입

입력도중 \n을 만날 때 \n문자를 만나면 읽기 중단 후 리턴 🡪 \n이 입력스트림에 남아있음

무한루프

cin.ignore(1) cin버퍼에서 문자 1개 삭제

입력도중 EOF나 오류가 발생할 때 읽기도중 EOF를 만나거나, 오류를 발생하면 읽기 중단

예제11-3 

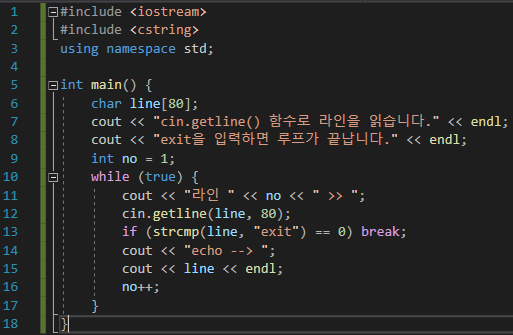
한 줄 읽기 istream& get(char\* s, int n, char delim=’\n’) n-1개의 문자를 읽어, 배열s에 저장

delim에 지정된 구분문자를 만나면 지금까지 만난 문자를 s에 저장하고 리턴

구분문자를 스트림버퍼에서 제거하지 않음 🡪 동일한 구분문자로 get()을 사용 시, 무한루프에 빠질 수 있음

istream& getline(char\* s, int n, char delim=’\n’) get()과 동일 but 구분문자를 스트림버퍼에서 제거

get()보단 getline() 사용 권장

예제11-4 

입력스트림의 문자 건너 띄기 istream& ignore(int n=1, int delim=EOF) 입력스트림에서 n개의 문자 제거, 도중에 delim을 만나면 delim문자를 제거하고 리턴

ex) cin.ignore(10); //입력스트림에서 10개의 문자 제거

cin.ignore(10, '\n'); //입력스트림에서 10개의 문자 제거, 제거도중 \n을 만나면 \n을 제거하고 중단

읽은 문자 개수 알아내기 int gcount() 최근에 읽은 바이트(문자)수 리턴(Enter키도 포함)

ex) char line[80];

cin.getline(line, 80);

int n = cin.gcount(); //6

check time 1. (ch = cin.get()) != EOF

2. 읽지 못함, 한글은 2바이트이기 때문

3. 엔터, 79, \n, 있다

11.4 포맷 입출력

포맷 입출력을 제공하는 3가지 방법 포맷플래그

포맷함수

조작자

포맷플래그 포맷플래그를 저장하는 멤버 변수

모든 입출력 스트림에 공통으로 사용됨

ios클래스에 정수형 상수로 정의되어 있음 하나의 플래그는 하나의 비트로 표현

long setf(long flags) 포맷지정함수

플래그를 스트림의 포맷플래그로 설정하고 이전 플래그 리턴

형식 cout.setf(long 플래그)

리턴 이전 플래그

long unsetf(long flags) 저장된 플래그를 해제하는 함수

형식 cout.unsetf(long flags)

리턴 이전 플래그

ex) cout.unsetf(ios::dec);

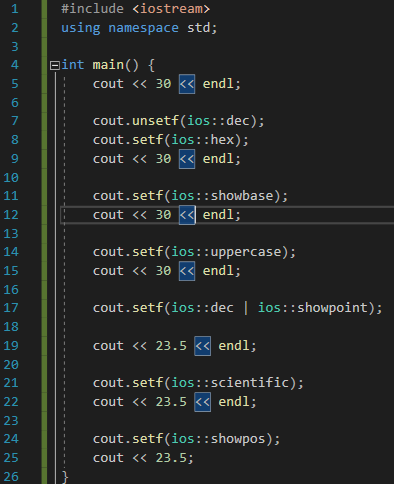
cout.setf(ios::hex);

cout << 30 << endl; //1e

|연산 여러 포맷을 동시에 지정 가능

ex) cout.setf(ios::dec | ios::showpoint);

cout << 23.5 << endl; //23.5000

예제11-5 

포맷 함수 활용 – 너비 설정, 빈칸 채우기, 유효숫자 자리수 지정 width() int width(int minWidth) 출력되는 필드의 최소 너비를 minWidth로 설정

호출 직후 하나의 필드에만 적용

형식 cout.width(int minWidth)

리턴 이전 너비

ex) cout.width(10);

cout << "Korea/" << "Seoul/" << "City" << endl;//korea/만 10칸

fill() char fill(char cfill) 필드의 빈칸을 cfill로 채움

형식 cout.fill(‘cfill’)

리턴 이전에 필드를 채우던 문자

ex) cout.fill('^');

cout.width(10);

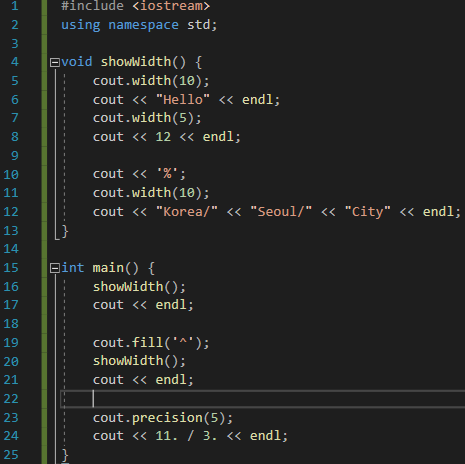
cout << "Hello" << endl; //^^^^^Hello

pricision() int pricision(int np) 유효숫자의 자릿수 변경

형식 cout.pricision(int np)

ex) cout.precision(5);

cout << 11. / 3.; //3.667

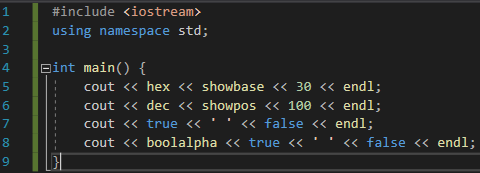
예제11-6 

조작자 <<, >> 연산자와 함께 쓰임

ex) cout << hex << showbase << 30 << endl; //hex 16진수 출력 조작자

//showbase 진수 표기 조작자 16-0x, 8-0

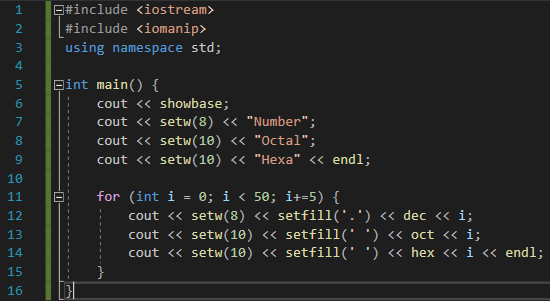
//endl 버퍼에 있는 데이터를 모두 출력하고 한줄 뛰도록 하는 조작자

예제11-7 

매개변수를 가지는 조작자 매개변수는 1개

<iomanip>헤더파일에 정의되어 있음 🡪 #include <iomanip> 필요

ex) cout << setw(10) << setfill('^') << "Hello" << endl; //너비:10, 빈공간:^

예제11-8 

11.5 삽입연산자(<<)와 추출연산자(>>)

삽입연산자의 중복 << 삽입연산자

본래의 삽입연산자 정수를 비트 단위로 시프트하는 C++의 기본 연산자

ostream클래스에 중복 작성된 연산자

삽입연산자의 실행 과정 << 연산자가 연속적으로 실행되는 세부과정

리턴타입이 의미하는 것 ostram& 리턴

구체적인 실행과정 컴파일러의 식변형 ex) cout << ‘a’ <<123 🡪 cout.<<(a)

<<연산자 함수 실행 인수를 스트림버퍼에 저장, 현재 스트림의 참조 리턴

컴파일러의 식변형 ex) cout <<123 🡪 cout.<<(123)

<<연산자 함수 실행 인수를 스트림버퍼에 저장, 현재 스트림의 참조 리턴

사용자 삽입 연산자(<<) 만들기 사용자가 정의한 클래스객체를 <<로 화면에 출력하는 경우

ex) class Point {

int x, y;

public:

Point(int x = 0, int y = 0) { this->x = x; this->y = y; }

friend ostream& operator<<(ostream& stream, Point& a);

};

ostream& operator<<(ostream& stream, Point& a) {

stream << "(" << a.x << "," << a.y << ")";

return stream;

}

int main() {

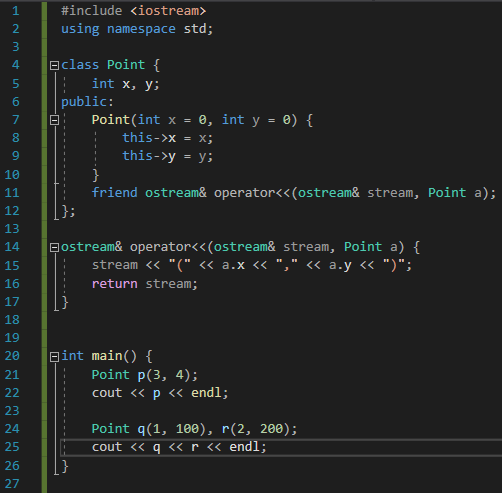
Point p(3, 4);

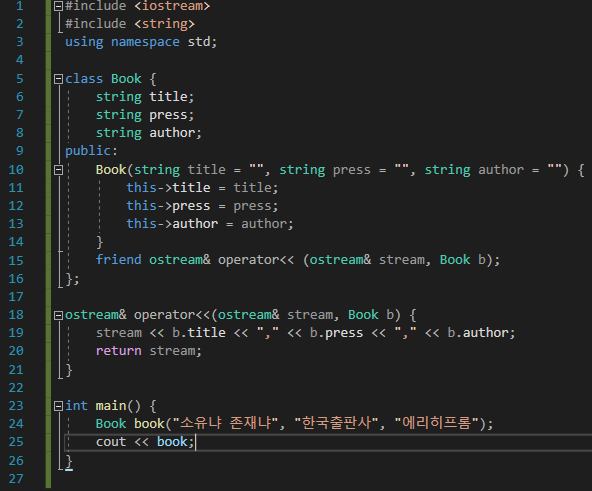
cout << p;

}

사용자 정의 삽입 연산자 함수의 원형 외부함수로만 작성

클래스의 private멤버에 접근하는 경우 friend로 선언

예제11-9 

예제11-10 

추출 연산자의 중복 >> 추출연산자

본래의 추출연산자 정수를 비트 단위로 오른쪽 시프트하는 C++의 기본 연산자

istream클래스에 중복 작성된 연산자

사용자 추출 연산자 만들기 사용자가 정의한 클래스객체를 >>로 값을 받는 경우

ex) class Point {

int x, y;

public:

Point(int x = 0, int y = 0) { this->x = x; this->y = y; }

friend istream& operator>>(istream& ins, Point& a);

};

istream& operator>>(istream& ins, Point& a) {

cout << "x 좌표>>";

ins >> a.x;

cout << "y 좌표>>";

ins >> a.y;

return ins;

}

int main() {

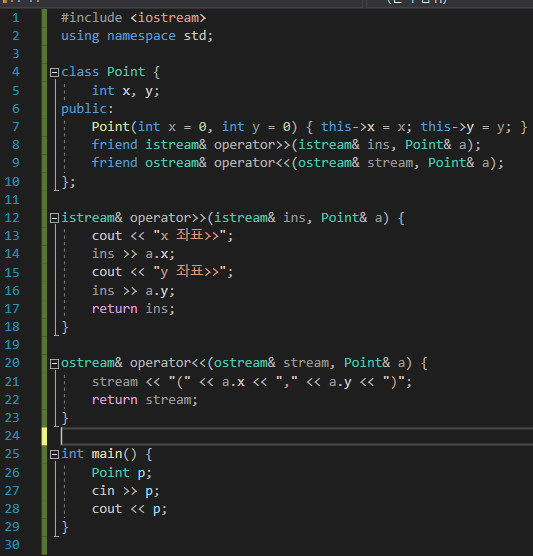
Point p;

cin >> p;

}

사용자 정의 삽입 연산자 함수의 원형 외부함수로만 작성

클래스의 private멤버에 접근하는 경우 friend로 선언

예제11-11 

check time 1.(1) ostream& operator<<(ostream& s, Circle c);

istream& operator>>(istream& s, Circle c);

1.(2) 없음

11.6 사용자 조작자 만들기

조작자의 실행과정 조작자는 <<, >> 연산자와 함께 실행됨

컴파일러의 식변형 ex) cout<<endl; 🡪 cout.(endl);

연산자 함수 호출 ostream& ostream::operator<<(ostream& (\*\_f)(ostream&)) {

(\*\_f)(\*this); //\*this는 cout

//(\*\_f) endl()함수호출문 = endl(cout)

return \*this; //cout의 참조리턴

}

새로운 조작자 만들기 매개변수가 없는 조작자 원형 istream manipulatorFunction(istream& ins);

ostream manipulatorFunction(ostream& outs);

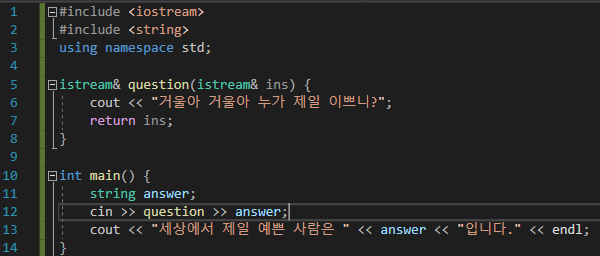
ex) ostream& fivestar(ostream& outs) {

outs << "\*\*\*\*\*";

return outs;

}

예제11-12 

예제11-13 

check time 1. ostream& tab(ostream& outs) {

outs << '\t';

return outs;

}