객체지향프로그래밍 정리노트(Week8)

202004029 김정호, 202004040 노성민

Q1. 11-1 ostream 멤버 함수를 이용한 문자 출력

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

intmain() {
	// "Hi!"를 출력하고 다음 줄로 넘어간다.
	cout.put('H');
	cout.put('i');
	cout.put(33);
	cout.put('\n');

	// "C++ "을 출력한다.
	cout.put('C').put('+').put('+').put(' ');

	char str[]="I love programming";
	cout.write(str, 6); //  str배열의 6 개의 문자 "I love"를 스트림에출력
}
```

김정호: 이 예제에서는 ostream 멤버 함수인 put(),write() 함수를 이용하는 예제인 것 같아.

노성민: 여기서 메서드를 연결해서 사용할 수도 있는 것을 확인할 수 있어.

Q2. 11-2 get()과 get(char&)을 이용한 한 줄의 문자 읽기

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

void get1() {
	cout<< "cin.get()로 <Enter> 키까지 입력 받고 출력합니다>>";
	int ch; // EOF와의 비교를 위해 int타입으로 선언
	while((ch= cin.get()) != EOF) { // 문자 읽기. EOF 는 -1
	cout.put(ch); // 읽은 문자 출력
	if(ch== '\n')
	break; // <Enter> 키가 입력되면 읽기 중단
	}
}

void get2() {
```

```
        cout<< "cin.get(char&)로 <Enter> 키까지 입력 받고 출력합니다>>";
        char ch;
        while(true) {
        cin.get(ch); // 문자 읽기
        if(cin.eof()) break; // EOF를 만나면 읽기 종료
        cout.put(ch); // ch의 문자 출력
        if(ch== '\n')
        break; // <Enter> 키가 입력되면 읽기 중단
        }
}

int main() {
        get1(); // cin.get()을 이용하는 사례
        get2(); // cin.get(char&)을 이용하는 사례
}
```

```
cin.get()로 <Enter> 키까지 입력 받고 출력합니다>>Do you love C++?
Do you love C++?
cin.get(char&)로 <Enter> 키까지 입력 받고 출력합니다>>Yes, I do.
Yes, I do.
```

김정호: 츌력된 값과 main 함수를 보고 get1,get2를 만들어보자.
우선 함수의 선언된 타입이 없는 걸 보아 void 타입으로 void get1(){ } 로 만들고 cin.get을 이용한 사례로 입력을 받으니 코드를 작성해보면 우선 cout < < " cin.get()로 <Enter> 키까지 입력받고 출력 합니다"; 하고 cin.get()이 EOF가 되기 전까지 반복하면 되니까 입력받을 값을 저장할 변수 int ch;를 만들고 while((ch = cin.get() != EOF){} 하고 cout.put(ch) 로 출력 후 enter가 눌리면 종료 if(ch == '\n') break;를 작성해주면 그러면 최종적으로 파란색처럼 작성할 수 있어. 이번엔 너가 get2 만들어보자.

노성민: 음 이번에는 cin.get(char&)를 이용해야 하니까 여기서도 get1과 똑같이 cin.get(char&)로 <Enter> 키까지 입력받고 출력 합니다"; 하고 여기서는 char ch; 로 선언해주고 이번에는 반환값이 int가 아닌 현재 입력 스트림 객체의 참조 값이니까 while(true)로 무한루프에서 cin.get(ch)로 입력된 값을 리턴해주고 그 cin.eof()가 참이면 종료하고 cout.put(ch)로 출력하고 똑같이 Enter 입력되면 break;문을 걸어서 종료하면돼. 그러면 빨간색처럼 작성될 거야.

Q3. 11-4 getline()으로 한 줄 단위로 문장 읽기

istream의 getline()을 이용하여 빈칸을 포함하는 한 줄을 읽고 다시 그대로 출력하는 프로그램을 작성하라.

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

intmain() {
        char line[80];
        cout<< "cin.getline() 함수로 라인을 읽습니다." << endl;
        cout<< "exit를 입력하면 루프가 끝납니다." << endl;
```

```
	int no = 1; // 라인 번호
	while(true) {
		cout<< "라인 " << no << " >> ";
		cin.getline(line, 80);// 79개까지의 문자 읽음
		if(strcmp(line, "exit") == 0)
			break;
		cout<< "echo --> ";;
		cout<< line << endl; // 읽은 라인을 화면에 출력
		no++; // 라인 번호 증가
	}
}
```

김정호: 여기서 getline을 이용하여 스트림 버퍼에서 delim 문자를 제거했어. 이를 통해 get을 이용해서 무한루프가 걸릴 수 있는 환경을 제거해주었어.

노성민: get 함수를 쓸 때 어떤 경우에서 무한루프가 생기는거야?

김정호: get은 버퍼에 '\n'이 남아있기 때문에 문자를 다시 읽을 때 '\n'부터 읽기 시작해서 아무것도 읽지 않은 채로 리턴하기 때문에 ignore(1)을 이용해서 버퍼에 남은 '\n' 을 제거해 줘야 해.

Q4. 11-5 setf(), unsetf()를 사용한 포맷 출력

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

intmain() {
	cout<< 30 << endl; // 10진수로 출력

	cout.unsetf(ios::dec); // 10진수 해제
	cout.setf(ios::hex); // 16진수로 설정
	cout<< 30 << endl;

	cout.setf(ios::showbase); // 16진수로 설정
	cout<< 30 << endl;

	cout.setf(ios::uppercase); // 16진수의 A~F는 대문자로 출력
	cout<< 30 << endl;

	cout.setf(ios::dec| ios::showpoint); // 10진수 표현과 동시에
	// 소숫점이하 나머지는 0으로 출력
	cout<< 23.5 << endl;

	cout.setf(ios::scientific); // 실수를 과학산술용 표현으로 출력
	cout<< 23.5 << endl;

	cout.setf(ios::showpos); // 양수인 경우 + 부호도 함께 출력
```

```
        cout<< 23.5;
}
```

노성민: 이 예제를 통해 포맷 플래그에 대해 알 수 있는 것 같아. 근데 포맷 플래그를 뭐라고 이해하면 될까?

김정호: 음 먼저 포맷 플래그는 모든 입출력 스트림에서 공통으로 사용되는데 ios 클래스에 정수형 상수로 정의되어 있어. 우리는 이를 활용해서 포맷 변수를 적절히 설정하여 입출력 포맷을 제어하면 되는 것 같아.

Q5. 예제 11-7 매개 변수 없는 조작자 사용

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

int main(){
        cout << hex << showbase << 30 << endl;
        cout << dec << showpos << 100 << endl;
        cout << true << ' ' << false << endl;
        cout << boolalpha << true << ' ' << false << endl;
}
```

김정호: 이 예제는 단순히 조작자 사용을 보여주는 예시인 것 같아. 조작자가 뭔지 이야기해보자.

노성민: 조작자는 입출력 포맷을 지정하는 방법으로 알고 있고 << , >> 연산자와 함께 사용된대.

김정호: 맞아. 추가적으로 이 예제에서 어떤 조작자가 사용됐는지 알고 넘어가자.

노성민: 여기서 hex는 16진수 표현 showbase는 16진수일 때 0x 8진수 일 때 0으로 숫자 앞에 추가해주는 역할을 하고 endl는 버퍼에 있는 데이터를 모두 출력 후 한 줄 띄도록 포맷하는 조작자야.

김정호: dec은 10진수 표현, showpos는 +, boolalpha를 이용해 ture, false가 문자열로 출력되게 했어.

Q6. 예제 11-9 Point 객체를 스트림에 출력하는 << 연산자 작성

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

class Point { // 한 점을 표현하는 클래스
        intx, y; // private 멤버
public:
        Point(intx=0, inty=0) {
        this->x = x;
        this->y = y;
        }
```

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

// << 연산자 함수
ostream& operator << (ostream& stream, Point a) {
        stream << "(" << a.x<< "," << a.y<< ")";
        return stream;
}

int main() {
        Point p(3,4); // Point 객체 생성
        cout<< p << endl; // Point 객체 화면 출력

        Point q(1,100), r(2,200); // Point 객체 생성
        cout<< q << r < endl; // Point 객체들 연속하여 화면 출력
}
```

김정호: 우선 main 함수를 살펴보면 우리가 만든 Point 클래스의 객체를 출력할 수 있는 << 연산자가 C++ 입 출력시스템에는 없기 때문에 우리가 이를 만들어 줘야 해. 먼저 ostream 클래스 내에서는 모든 연산자들이 ostream&를 리턴해. 우리가 복사본을 출력하면 원하는대로 출력이 안되는 것은 앞선 파트에서 이해했잖아. 이러한 이유 때문에 참조를 리턴해 주어야해. 그리고 객체의 값을 출력하기 위해서는 Point 클래스의 멤버 변수에 접근해야 하는데 << 연산자가 외부에서 호출되었기 때문에 함수를 외부에 작성을 해야하는데 이를 해결하기 위해서 friend 함수를 이용해서 해결하면 돼.

Q7. 11-11 Point 객체를 입력 받는 >> 연산자 작성

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

class Point { // 한 점을 표현하는 클래스
        intx, y; // private 멤버
public:
        Point(intx=0, inty=0) {
        this->x = x;
        this->y = y;
        }
        friend istream& operator >> (istream& ins, Point &a); // friend 선언
        friend ostream& operator << (ostream& stream, Point a); // friend 선언
};
istream& operator >> (istream& ins, Point &a) { // >> 연산자 함수
        cout<< "x 좌표>>";
        ins >> a.x;
        cout<< "y 좌표>>";
        ins >> a.y;
        return ins;
}
```

```
ostream& operator << (ostream& stream, Point a) { // << 연산자 함수
        stream << "(" << a.x<< "," << a.y<< ")";
        return stream;
}

int main() {
        Point p; // Point 객체 생성
        cin>> p;  // >> 연산자 호출하여 x 좌표와 y 좌표를 키보드로 읽어 객체 p 완성
        cout<< p;  // << 연산자 호출하여 객체 p 출력
}
```

김정호: 이 예제는 11-9 예제에서 설명했던 예제와 매우 흡사해. 이번엔 너가 설명해보자.

노성민: 이번에도 Point 클래스의 객체에 대한 >> 연산자가 없으니 우리가 만들어 줘야 해. 이번에 다른 점은 istream을 이용해야 한다는 점이야. 그리고 이번엔 값을 변경해야 하기 때문에 Point &a로 매개변수를 설정해야 해. 이를 제외하고는 ostream을 이용할 때와 큰 차이는 없어.

#Q1. 11-12,13 사용자 정의 조작자 만들기

```
#include<iostream>
#include<string>
using namespace std;

istream& question(istream& ins){
        cout << "거울아 거울아 누가 제일 예쁘니?";
        return ins;
}

ostream& fivestar(ostream& outs){
        return outs << "*****";
}
ostream& rightarrow(ostream& outs){
        return outs << "---->";
}
ostream& beep(ostream& outs){
        return outs << "\a";
}
int main() {
        string answer;
        cout << "벨이 울립니다" << beep << endl;
        cout << "C" << rightarrow << "C++" << rightarrow << "Java" << endl;
        cout << "Visual" << fivestar << "C++" << endl;
        cin >> question >> answer;
        cout << "세상에서 제일 예쁜 사람은 " << answer << "입니다." << endl;
}
```

김정호: 연산자를 만드는 방식처럼 조작자도 리턴 값에만 주의해서 만들어 줄 수 있어.