협성대학교 소프트웨어공학과

고급프로그래밍 2 중간고사 정리

C++, 강의 1 ~ 7 주차 내용 정리.

20200670 이시헌

2023-10-23

솔직히 말하자면, 이것을 작성해야 하는 이유는 없다. 이미 다 잘 알고 있는 내용이고 누구나 쉽게 체득했을 내용일 것이다. 그래도 이 문서를 작성하는 이유는 두가지가 있다. 다른 과목과 공부 방식에 차등을 두지 않기 위함이 첫 번째이고, 복습을 통하여 실수의 가능성을 조금이라도 낮추어 보기 위함이 둘째이다.

1. C++ 프로그램의 실행 과정

   .cpp -〉 컴파일(라이브러리를 제외하고 기계어 번역) -〉 .obj -〉 링킹(라이브러리 연결) -〉 .exe

2. C++ 표준 라이브러리

   C++ 표준 라이브러리는 세 종류가 있다.

   ① C 라이브러리 : C 라이브러리는 이름이 C 로 시작하는 헤더로 정리되어 있다. (cmath 등)

   ② C++ 입출력 라이브러리 (iostream, istream, ostream 등)

   ③ C++ STL 라이브러리 (vector 등)

3. 식별자, 예약어, 변수, 상수에 대해서는 C 언어와 거의 같다.

   상수 정의법이 두가지가 있음을 잊지 마라. 항상 define 문의 인자 순서를 헷갈리니, 기록해 두겠다.

   ① #define PI 3.14 //선행처리문이므로 세미콜론 없음.

   ② const int PI = 3.14; //변수를 상수화하므로 세미콜론 있음.

4. 데이터타입

   데이터타입에는 두종류가 있다.

   ① 기본 타입 : 숫자, 문자, 문자열 전용 파생 타입(C 언어와 달리 문자열도 기본타입임)

   ② 파생 타입 : 배열, 클래스, 포인터, 사용자 정의 ……

5. 기타 상식

   A. C++에서도 여전히 char 타입은 아스키코드 기반이다.

   B. C++에서도 이스케이프 시퀀스는 동일하다.

6. 문자열

문자열은 두가지 방식으로 표현이 가능하다.

① C-stirng 방식 : char 타입 배열 + 문자열 끝부분 NULL 문자.

② string 클래스 이용

7. 혹시 모르니 cin.getline()의 사용법을 기억해두라.

A. 공백문자를 포함하여 입력 받을 때 사용한다.

B. istream 라이브러리에 속한다.

```
char a[10];
cin.getline(a, 10);

//cin.getline(저장 위치, 최대 입력 가능 개수 + 1, 만나면 입력을
중단할 제한자 입력 - 미 입력시 엔터(\n)임);
```

8. 정말 혹시 모르니 그냥 getline()의 사용법도 기억하자.

A. string 라이브러리에 속한다. 전역함수이므로 객체 없이 사용 가능.

```
string str;

getline(cin, str);  //문자열 끝까지 cin 을 이용하여 입력.
getline(cin, str, q);  //문자 'q'를 만나면 입력 중단. (q 까지 입력)
```

9. string 의 알아 두면 좋은 메소드

A. .size( ) : 객체가 갖고 있는 문자열의 길이(사이즈) 반환.

B. .find( ) : 입력받은 문자열의 시작 인덱스 반환. 문자열임을 주의!

```
string s1 = "game over";
s1.find("over");
//이때는 인덱스 5 반환. ('o'의 위치 반환)
```

C. .substr( ) : 시작 인덱스부터 끝 인덱스까지 복사해서 리턴.

```
string s = "hello";
cout << s.substr(1,3);
//이 경우 "ell" 리턴.
```

D. .c_str( ) : string 객체를 C-string 문자열로 변환하여 반환.

```
string s = "hello";
cout << s.c_str();  //인자 없이 동작함.
```

10. namespace

특정한 식별자들을 하나의 공간에 모아서 분리시키는 공간. 동일한 이름의 식별자를 여러 객체에서 사용 가능하도록 함. 사용자가 임의로 네임스페이스를 선언하여 식별자를 특정 공간에 포함시킬 수 있음.

스코프 연산자 '::'로 어느 네임스페이스에 속한 식별자를 사용할지 선택 가능.

11. cout 에 대하여

A. boolalpha / noboolalpha 메소드 : cout 이 bool 타입을 어떻게 처리할지 지정하는 메소드임.

```
cout << true;    // 1 출력
cout << boolalpha;
cout << true;    // true 출력
cout << noboolalpha;
cout << true;    // 1 출력
```

B. '〉〉' 연산자

이는 '산술 시프트 연산자'였지만, cout 클래스 내부에서 '스트림 추출 연산자'로 재정의되었다.

입력 스트림에서 값을 받아서 변수에 저장하는 것 까지 정의되어 있다.

12. cin 에 대하여

입력 버퍼를 내장하고 있다.

엔터(₩n)키가 입력될 때 까지 입력된 키를 버퍼에 계속 저장한다. 그리고 'backspace'키 입력시 버퍼에 저장된 값 하나를 삭제한다(스택). 단, 공백문자를 만나면 cin 은 값 저장을 중단하므로, 공백문자를 입력 받으려면 getline( )함수를 사용해야 한다.

13. 연산자에 대하여

대체로 C 언어와 유사하다. 다만… 혹시 헷갈릴 수 있는 몇 가지를 적어보면……

① y=(1+x++)+10; //이 경우 x 의 값은 다음 행이 시작될 때 증가함. 후위로 작성한 증감연산자가 항상 다음 행 시작시에 동작한다는 것을 명심하라.

② 관계연산자는 항상 하나만 사용해야 한다. 2 〈 x 〈 5 처럼 작성하면 컴파일 오류 발생! (2 〈 x) && (x 〉 5)로 작성해야 함.

③ 논리연산자를 이용하여 제한된 상황에서 단축 평가 논리 계산 가능. 위의 ② 에서 &&를 사용한 식은, 앞의 2<x 가 거짓인 경우 뒤의 x>5 부분은 실행되지 않는다는 점에서, 이를 단축 계산이라 칭한다. 분기문보다 가독성은 떨어져도 코드의 길이를 단축할 수 있다.

14. 제어문 관련 내용은 작성을 생략하겠다.

15. 배열 내용도 생략하겠다.

16. 포인터에 대하여 색다른 내용이 있어 그것만 기록하겠다.

배열의 모든 값을 두배로 증가시킬 때, 나는 1 번 방법의 코드를 생각했지만, 교수님께서는 2 번 방법의 코드를 작성하셨다. 아마도 이터레이터의 개념에 익숙하신 분이라, 저런 코드를 작성하신 것 같다. 하지만 내 생각엔 이터레이터를 사용하지 않고 포인터로 저런 조작을 할 이유가 없다. 포인터를 사용했다면, 포인터 변수의 값을 불변하는 등대처럼 사용하며, 인덱스를 더하도록 표현하는 것이 직관적이라 생각한다.

```
int* p = ~~~;
for(int i=0; i<10; i++){

    //방법 1 : 포인터 변수의 값을 유지한 채로, 인덱스를 그때그때
더하여 배열 순회.
    *(p+i) = *(p+i)*2;

    //방법 2 : 포인터 변수의 값을 증가시키며 배열 순회.
    *p = *p*2;
    p++;
}
```

17. new / delete 연산자

```
int* pt1 = new int;
int* pt2 = new int[10];

cout << pt2[3];
cout << *(pt + 3);

delete pt1;
delete[] pt2;
```

18. 함수에 대하여

A. 함수 원형의 선언 이유와 방법 생략.

B. 함수에 배열을 넘길 때는 이름과 크기를 따로 넘겨야 함. (5 주차 강의)

C. 함수 호출시에 주소를 넘기려면(&변수명), 함수의 매개변수는 포인터 변수로 선언되어 있어야 함. 배열의 경우는 배열타입도 가능(배열은 포인터니까). 뭐 당연한 말이지만……

D. 참조 타입 변수 : C++에서 새로 생긴 개념. &기호를 포인터 쓰듯이 사용해서 변수를 선언하면, 해당 변수의 이름은 다른 변수의 별명으로 사용 가능. 해당 변수에 새로운 변수를 대입하면, 해당 변수는 새로운 변수의 이름으로도 접근 가능. 말 그대로, 참조타입임. 이때 참조타입 변수는 물리적 메모리 주소를 갖는 객체만을 받을 수 있다. 상수를 대입할 수 없다는 뜻! 조심하라!

```
int a;
int& ra = a;
ra = 1;              //ra는 a의 별명임.
cout << a << endl;  // 1 출력됨.
```

E. 디폴트 매개변수 사용 가능하다. 단, 프로토타입 선언시 프로토타입 부분에만 1 회 선언 가능(디폴트 값 중복지정 안됨)

F. 함수 오버로딩은 매개변수의 타입과 개수에만 가능 여부가 갈린다.

19. 클래스에 대하여

A. 포인터에 클래스 객체를 담은 경우, 해당 포인터의 이름으로 객체 참조시에는 '.'연산자(직접 멤버 연산자)가 아니라 '->'연산자(간접 멤버 연산자)를 사용해야 함.

B. 함수에 객체를 넘길 때, 값에 의한 호출임을 기억하라. 참조에 의한 호출을 원하면, 포인터를 사용하거나, 함수의 매개변수를 참조 타입 변수로 만들어야 한다.

C. 파일을 분리하여 클래스를 선언, 정의한다면… 일반적으로 클래스의 선언부는 헤더파일로 만들고, 구현부는 .cpp 파일로 만든다.

D. 다른 내용은 헷갈릴 여지가 없으므로, 기록하지 않겠다. (*)

이상으로 고급프로그래밍 2 과목의 중간고사 대비가 끝났다.

강의 1 주차부터 7 주차까지의 내용을 정리한 것으로, 5 주차까지의 강의는 C 언어 복습이었고, 6, 7 주차는 클래스 기초에 대한 내용이었다.