|  |
| --- |
| 2020년 1학기 |
| 프로그래밍 언어 레포트 |
| #14주차 학습활동 |

|  |
| --- |
| 송 지민  사회학과 201721899 |

**[과제#1] 은행계좌를 나타내는 구조체**

* **코드 및 실행화면**

#include <stdio.h> // 표준입출력 헤더파일 포함

struct account { // 구조체 선언(정의)

/\* C의 자료형은 크게 기본 자료형과 파생 자료형으로 나눌 수 있다.

기본 자료형은 int, double, char와 같은 기본적 자료형이며 파생 자료형은 배열, 구조체, 공용체, 포인터 등이다.

구조체는 파생 자료형 중에서도 가장 일반적이고 많이 사용되는 자료형이다.

배열에는 동일한 종류의 데이터만 저장할 수 있지만 구조체를 활용하면 서로 다른 자료형을 하나로 묶을 수 있다.

account는 구조체의 이름이며 struct는 구조체 정의 시 사용하는 키워드이다.

아래는 구조체의 멤버가 위치하며 구조체 선언도 하나의 문장에 해당하기 때문에 중괄호 뒤에 세미콜론을 반드시 사용해야 한다.\*/

// 구조체 멤버

int number; // 계좌번호

// long long 으로 선언하였을 때 아래 실행화면과 같은 오류가 발생



char name[20]; // 고객이름

int balance; // 잔액

};

int main(void) // 반환값이 정수형이며 매개변수가 없는 메인 함수의 선언부

{

struct account s = { 1, "홍길동", 10000 }; // 구조체 변수 선언 및 초기화

/\* 구조체 정의와 구조체 변수 선언은 다르다. 구조체 선언은 구조체 안에 어떤 변수들이 들어간다는 것만 말해주는 것이다.

즉 구조체 선언은 구조체의 형태만 정의한 것이고 아직 구조체를 이용하여 변수를 만들지 않았으므로 위와 같이 구조체 변수를 반드시 선언해야 한다.

하나의 문장에서 여러 개의 구조체 변수를 동시에 선언할 수도 있으며 구조체 정의와 구조체 변수 선언을 동시에 할 수도 있다.\*/

printf("{ %d, %s, %d }", s.number, s.name, s.balance); // 출력하기

/\* . 기호는 구조체에서 멤버를 참조할 때 사용하는 연산자이다. 구조체변수.구조체멤버와 같은 형태로 사용한다.

포인터를 사용하여 p(포인터, s의 주소를 저장) -> number와 같이 나타낼 수도 있다.\*/

return 0; // 0 반환하여 끝내기

}



**[과제#2] 구조체를 이용한 이메일 출력**

* **코드 및 실행화면**

#include <stdio.h> // 표준입출력 헤더파일 포함

#include <string.h> // 문자열 처리 라이브러리 포함

// 문자열을 구조체 변수에 복사하기 위해 선언

struct email { // 구조체 정의

/\* 구조체는 서로 다른 자료형의 변수들을 묶어서 새로운 자료형을 만드는 것이다.

구조체와 함수를 결합한 것이 객체지향 언어의 class라고 할 수 있다.

email은 구조체의 이름(태그)이며 아래는 구조체의 멤버이다.

구조체 선언은 변수 선언이 아니며 구조체 안에 어떤 변수들이 들어간다는 것만 정의한 것이다. 구조체를 정의하는 것은 붕어빵 틀을 정의하는 것과 같다.

소스 파일의 첫 부분에서 구조체를 정의할 경우 이 구조체 정의를 전체 소스 파일에서 사용할 수 있다.

만약 구조체를 특정 함수 안에서 정의하면 그 함수에서만 구조체 정의를 사용하여 변수를 만들 수 있다.\*/

// 구조체 멤버 선언

char title[100]; // 제목

char sender[100]; // 수신자

char receiver[100]; // 발신자

char content[100]; // 내용

char date[100]; // 날짜

int pri; // 우선순위

};

print(struct email e) // 메일의 항목을 출력하는 함수의 선언부

/\* 구조체 변수를 매개변수로 받는 함수이며, 구조체를 함수의 인수로 넘기는 방법은 다른 기본 자료형을 넘길 때와 크게 다르지 않다.

구조체의 복사본이 함수로 전달되므로 구조체의 크기가 크다면 그만큼 시간과 메모리가 소요된다.

따라서 구조체의 크기가 클 경우 구조체의 포인터를 인수로 보내는 것이 좋다.

구조체 포인터를 받아 함수에서 간접 멤버 연산자(->)를 사용할 경우 잘못하면 원본 데이터가 훼손될 수 있다.

이를 방지하는 방법은 원본을 읽기만 하는 경우에 const를 쓰는 것이다.\*/

{

printf("제목 : %s\n", e.title); // 메일 제목(문자열)을 출력

printf("수신자 : %s\n", e.sender); // 수신자(문자열)을 출력

printf("발신자 : %s\n", e.receiver); // 발신자(문자열)을 출력

printf("내용 : %s\n", e.content); // 내용(문자열)을 출력

printf("날짜 : %s\n", e.date); // 날짜(문자열)을 출력

printf("우선순위 : %d\n", e.pri); // 우선순위(정수)를 출력

} /\* 구조체 변수를 인수로 받아 구조체 멤버의 값을 출력하는 함수이다. 멤버 연산자(.)를 활용하여 출력이 이루어진다.\*/

int main(void) // 메인 함수의 선언부

{

struct email e; // 구조체 변수 선언

/\* 구조체를 정의할 때 태그 이름은 생략하여도 무방하지만 이 경우 구조체의 이름이 없으므로 이 구조체의 변수를 더 이상 생성할 수 없다는 맹점이 있다.

만약 위에서 구조체 정의 시 이름을 생략했다면 필요한 모든 구조체 변수를 지금과 같이 후에 선언할 수 없으므로 구조체 정의와 함께 구조체 변수를 선언해야 한다.

구조체 변수를 구조체와 같이 정의하는 방법은 struct {} 구조체 변수 이름; 과 같다.\*/

strcpy\_s(e.title, 100, "안부 메일"); // 구조체 변수의 멤버 title에 문자열을 복사

strcpy\_s(e.sender, 100, "chulsoo@hankuk.ac.kr"); // 구조체 변수의 멤버 sender에 문자열을 복사

strcpy\_s(e.receiver, 100, "hsh@hankuk.ac.kr"); // 구조체 변수의 멤버 receiver에 문자열을 복사

strcpy\_s(e.content, 100, "안녕하십니까? 별일 없으신지요?"); // 구조체 변수의 멤버 content에 문자열을 복사

strcpy\_s(e.date, 100, "2010/9/1"); // 구조체 변수의 멤버 date에 문자열을 복사

/\* 구조체 멤버가 문자열이라면 위와 같이 strcpy()함수를 이용해 문자열을 복사해야 하는데, 비주얼 스튜디오의 경우 strcpy() 대신 strcpy\_s()를 사용해야 한다.

또한 이 때 인수는 문자열을 복사하여 저장할 문자열, 복사될 문자열 외에 문자열의 크기를 지정해주어야 하는 것으로 보인다. 인수를 문자열 두 개만 주면 오류가 나므로 주의해야 한다.\*/

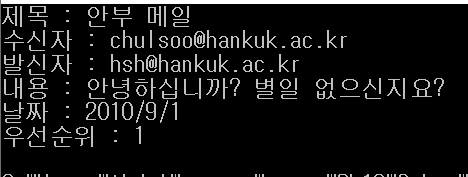
e.pri = 1; // 구조체 변수의 멤버 pri에 1을 저장

print(e); // 구조체 변수를 함수에 인수로 주어 호출

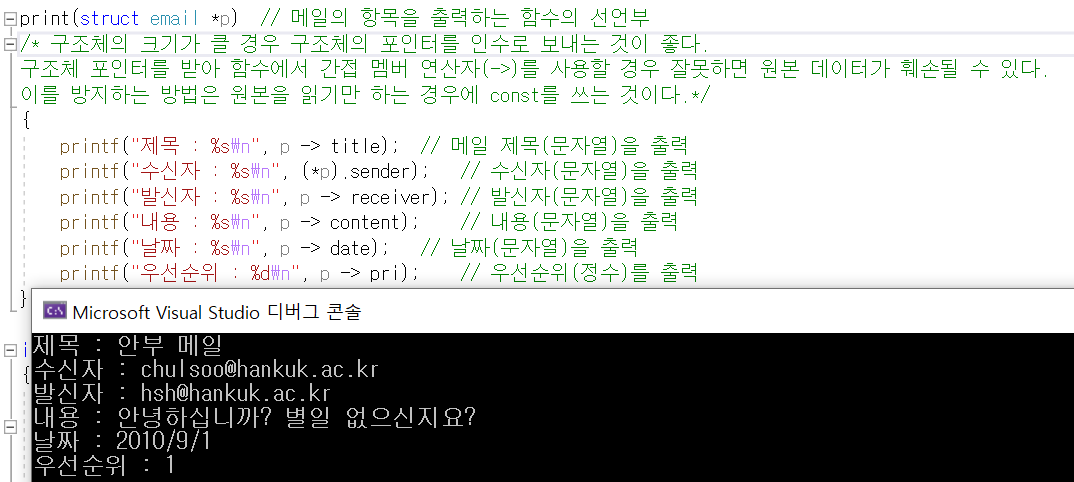
/\* 앞서 살펴본 것처럼 구조체를 인수로 주어 함수를 호출하는 방법은 일반 변수 사용 시와 크게 다르지 않다.\*/

return 0; // 0을 반환하여 메인 함수 끝내기

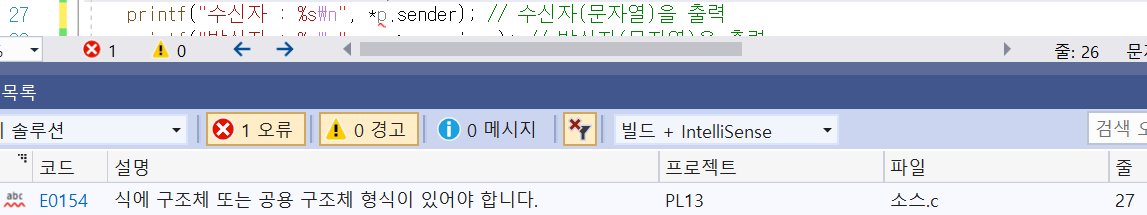
}



* **함수의 인수로 구조체 포인터 주기( print(e)을 print(&e)로 )**

****

간접 멤버 연산자를 사용하든, 포인터 연산자를 사용하든 상관없이 똑같이 출력됨을 확인하였습니다. 다만 이 때 괄호없이 \*p.sender 와 같은 형태로 사용하면 이는 구조체 p의 멤버 sender가 가리키는 것이란 의미이므로 sender가 포인터가 아니기 때문에 오류가 발생합니다.



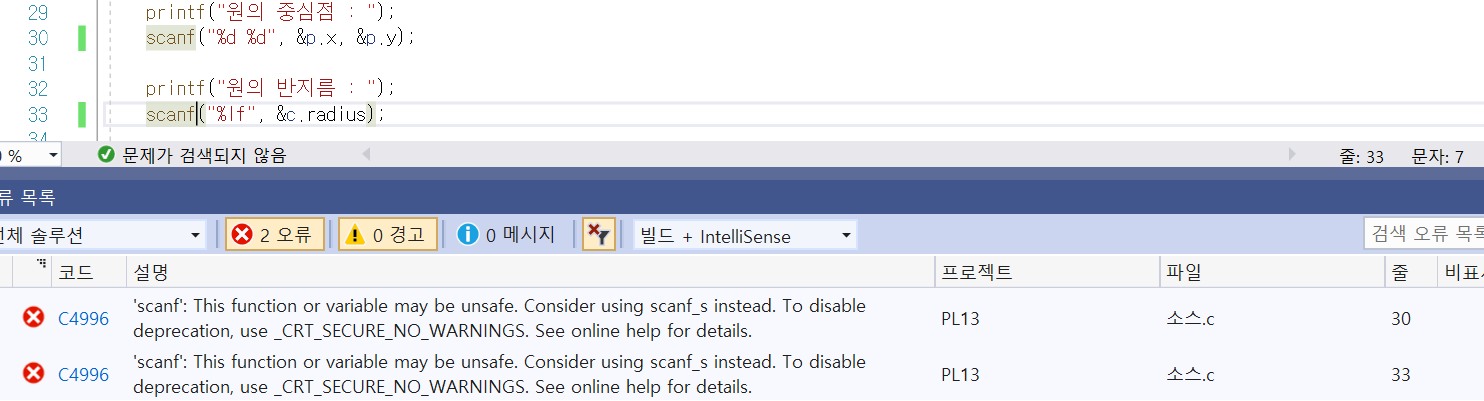
**[과제#3] circle과 point 구조체를 이용한 프로그램**

* **코드 및 실행화면**

**#define \_CRT\_SECURE\_NO\_WARNINGS**

**/\* 비주얼 스튜디오에서는 scanf를 사용하면 오류가 발생한다.**

**오류를 수정하기 위해 비주얼 스튜디오는 두 가지 방법을 제시하는데 그 중 하나가 scanf함수 대신 scanf\_s를 쓰는 것이고 다른 하나는 바로 위의 방법이다.**

****

**scanf함수를 쓴 상태에서 전처리 없이 실행을 하면 오류가 발생하지만(캡처화면) 위와 같이 코드를 작성하면 scanf함수를 사용해도 오류가 발생하지 않는다.\*/**

**#include <stdio.h> // 표준입출력 헤더파일 포함**

**struct point { // 원의 중심 구조체 정의**

**int x, y; // 구조체 멤버**

**/\* 구조체는 서로 다른 자료형을 하나로 묶을 수 있으나 이 구조체에서는 정수형 변수만 포함되었다.**

**구조체는 서로 다른 타입의 자료형을 하나로 묶어 배열처럼 메모리 공간에 연속적으로 나열하는 자료형이라고 할 수 있으며 위의 경우 정수형 멤버 2개를 가지므로 8바이트의 메모리가 할당되었을 것이다.**

**더 정확하게 알기 위해서는 sizeof 연산자를 이용할 수 있다.\*/**

**};**

**struct circle { // 원 구조체 정의**

**// 구조체 멤버**

**struct point center; // 원의 중심**

**/\* 어떠한 자료형이라도 구조체의 멤버가 될 수 있는데 구조체 또한 다른 구조체의 멤버가 될 수 있다.**

**구조체를 다른 구조체 안에 중첩하여 포함시키는 것은 허용되는데 적어도 15번까지는 중첩이 가능하지만 이런 경우 멤버 변수를 참조하려면 많은 수의 마침표를 포함하는 상당히 긴 이름을 임력해야 한다.**

**따라서 대개 3번 이상 중첩시키는 것은 피하는 것이 좋다.**

**이렇게 중첩된 구조체 멤버 변수를 참조할 때는 구조체 태그(이름)이 아니라 구조체 변수와 멤버를 이용하는 것으로 보인다.\*/**

**double radius; // 원의 반지름**

**};**

**double area(struct circle c) { // 반환값이 실수이며 매개변수로 구조체 변수를 갖는 원의 면적 함수**

**return c.radius \* c.radius \* 3.14; // 원의 면적을 계산하여 반환**

**/\* 구조체 변수 c의 멤버 radius의 값을 이용하여 원의 면적을 계산한다.\*/**

**}**

**double perimeter(struct circle c) { // 반환값이 실수이며 매개변수로 구조체 변수를 갖는 원의 둘레 함수**

**return c.radius \* 2 \* 3.14; // 원의 둘레를 계산하여 반환**

**/\* 구조체 변수 c의 멤버 radius의 값을 이용하여 원의 둘레를 계산한다.\*/**

**}**

**int main(void) // 반환값이 정수형이며 매개변수가 없는 메인 함수의 선언부**

**{**

**struct circle c; // 원 구조체 변수 선언**

**struct point p; // 원의 중심 구조체 변수 선언**

**/\* 구조체를 int와 같은 새로운 타입으로 생각하면 이해가 쉽다. int x를 통해 int형 변수 x를 선언한 것처럼 struct circle c를 통해 circle 타입의 변수 c를 선언했다고 생각할 수 있다.\*/**

**double re\_1, re\_2; // 계산한 원의 면적, 둘레 값을 받아 저장하기 위한 변수**

**printf("원의 중심점 : "); // 원의 중심점 입력 안내**

**scanf("%d %d", &p.x, &p.y); // 원의 중심점 입력받기**

**/\* 구조체 멤버에 값을 입력하여 저장할 때는 기본 자료형을 입력받을 때와 마찬가지로 주소 연산자를 사용한다.**

**이 때 참조를 위해 (.)연산자를 사용하는 것을 주의한다.\*/**

**printf("원의 반지름 : "); // 원의 반지름 입력 안내**

**scanf("%lf", &c.radius); // 원의 반지름 입력받기**

**/\* 위와 마찬가지의 방법으로 (.)연산자를 이용해 구조체 변수 c의 멤버 radius에 사용자로부터 값을 입력받아 저장한다.**

**이 때 반지름의 자료형을 실수형으로 선언한 것을 고려해 형식지정자는 %lf를 사용하였다.\*/**

**re\_1 = area(c); // 면적 계산 함수에 구조체 변수 c를 인수로 주어 호출하고 반환값을 저장**

**re\_2 = perimeter(c); //둘레 함수에 구조체 변수 c를 인수로 주어 호출, 반환값 저장**

**/\* 함수에 구조체 변수를 인수로 주는 방법은 기본 자료형을 변수로 주는 것과 크게 다르지 않다.**

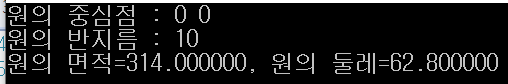
**이 때도 일반적인 경우와 마찬가지로 복사본을 전달하기 때문에 함수를 호출하면 새로운 메모리 공간이 할당된다.**

**따라서 구조체의 크기가 클 경우 비효율적이므로 포인터를 이용하여 원본을 전달하는 것이 좋다.\*/**

**printf("원의 면적=%lf, 원의 둘레=%lf\n", re\_1, re\_2); // 원의 면적과 둘레를 출력**

**return 0; // 0을 반환하여 메인 함수 끝내기**

**}**

****

**[응용] 싱싱마트 주문배달 프로그램**

* **코드 및 실행화면**

// 싱싱마트 주문배달 프로그램

#define \_CRT\_SECURE\_NO\_WARNINGS // scanf함수를 이용하기 위함

#include <stdio.h> // 표준입출력 헤더파일 포함

#define TOMATO 1.5 // 토마토의 가격 기호상수로 선언

#define EGG 0.8 // 달걀의 가격 기호상수로 선언

#define BANANA 2.5 // 바나나의 가격 기호상수로 선언

struct James { // 구조체 정의

// 구조체 멤버

int tomato; // 토마토 개수

int egg; // 달걀 개수

int banana; // 바나나 개수

double cost;// 총금액

int will; // 주문의향

char address[100]; // 주소

char phone[20]; // 전화번호

};

int main(void) // 메인함수의 선언부

{

struct James s; // 구조체 변수 선언

printf("========James의 장바구니===========\n"); // 장바구니 출력

printf("토마토를 몇 개 구매하시겠습니까? "); // 토마토 구매 안내

scanf("%d", &s.tomato); // 토마토 개수 입력받기

printf("달걀을 몇 개 구매하시겠습니까? "); // 달걀 구매 안내

scanf("%d", &s.egg); // 달걀 개수 입력받기

printf("바나나를 몇 개 구매하시겠습니까? "); // 바나나 구매 안내

scanf("%d", &s.banana); // 바나나 개수 입력받기

s.cost = s.tomato \* TOMATO + s.egg \* EGG + s.banana \* BANANA;

// 총금액 처리하기

printf("\n"); // 줄바꿈

printf("총금액 : %.2lf\n", s.cost); // 총금액 출력하기

printf("===================================\n"); // 장바구니 닫기

printf("\n주문하시겠습니까?\n"); // 주문의향 질문하기

printf("주문을 원하시면 1번을, \n취소를 원하시면 2번을 입력해주세요. ");

scanf("%d", &s.will); // 주문의향 입력받기

printf("\n"); // 줄바꿈

while(s.will !=1 && s.will !=2) { // 사용자가 주문이나 주문취소를 선택하지 않는 동안 반복

printf("잘못 입력하셨습니다. 다시 입력해주세요. "); // 오류 안내

scanf("%d", &s.will); // 다시 입력받기

printf("\n"); // 줄바꿈

}

if(s.will == 1) { // 사용자가 주문의향이 있는 경우

printf("핸드폰 번호를 입력해주세요. "); // 전화번호 입력안내

scanf("%s", &s.phone); // 전화번호 입력받기

printf("배송지를 입력해주세요. "); // 배송지 입력안내

getchar(); // 배송지를 입력받지 않는 오류가 발생하여 버퍼 비우기

gets\_s(s.address, 100); // 배송지 입력받기

printf("\n주문이 완료되었습니다.\n싱싱마트 주문배달을 이용해주셔서 감사합니다.\n\n");

// 주문 완료 안내

printf("================영수증=============\n"); // 영수증 출력

printf("토마토 %d개 : %.2lf달러\n", s.tomato, s.tomato \* TOMATO);

printf("계란 %d개 : %.2lf달러\n", s.egg, s.egg \* EGG);

printf("바나나 %d개 : %.2lf달러\n", s.banana, s.banana \* BANANA);

// 각 물품의 개수와 가격 출력

printf("\n총금액 : %.2lf\n", s.cost);

// 총금액 출력

printf("\n전화번호 : %s\n", s.phone);

printf("배송지 : %s\n", s.address);

// 전화번호와 배송지 출력

printf("===================================\n"); // 영수증 출력 완료

}

else{ // 사용자가 주문의향이 없을 경우

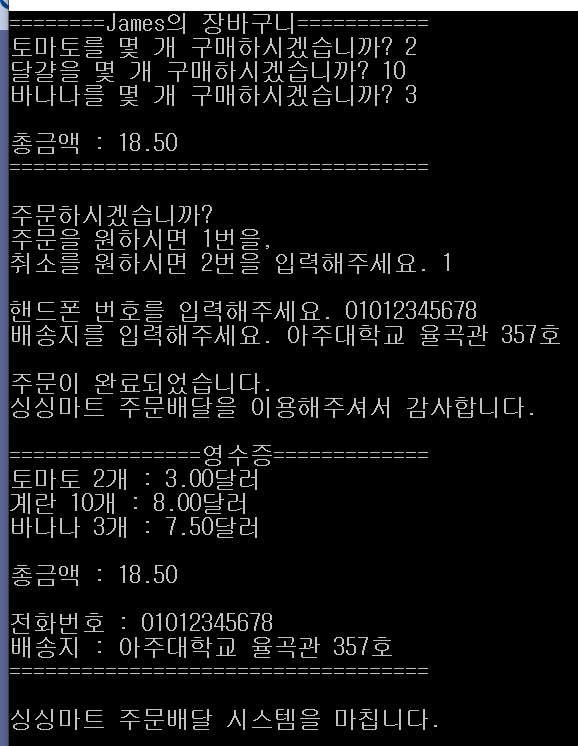
printf("주문이 취소되었습니다.\n"); // 주문 취소 안내

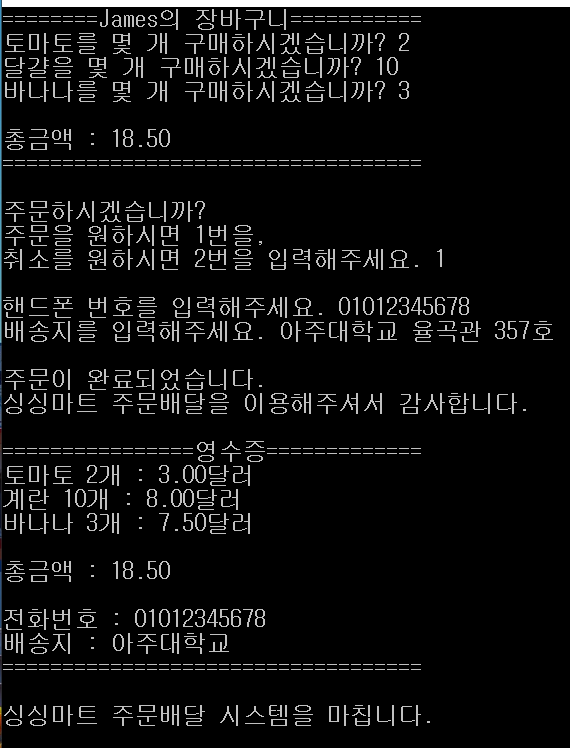
}

printf("\n싱싱마트 주문배달 시스템을 마칩니다.\n"); // 시스템 종료 안내

return 0; // 0 반환하여 끝내기

}

**정상 출력 화면**

**오류 출력화면(주소를 일부만 출력함)**

이를 수정하기 위해 코드를

scanf(“%s”, &s.address);에서 gets\_s(s.address, 100);으로 변경하여 작성하였습니다.