솔루션: 프로젝트(cpp,cs) 파일들의 집합체

# : 전처리 기호

C++에는 함수앞에 public나 private가 붙지 않는다

시작 함수인 int main()(= awake())은 리턴값을 생략해도 가능하다

C++에서 문자열은 +로 연결할 수 없다??(cout << “hello” << “World”;

실수형 변수에 값을 대입할 때 값에 f를 안붙여도 된다?

포인터 선언 int\* (자료형 뒤에 \*)

포인터 대입 int\* data = &value (대입하는 변수 앞에 &)

(C++에서?) float의 비교 연산은 위험하다 수식계산은 맞지만   
데이터 상의 수로는 같지 않을수 있기 때문  
-> num0 = 1.0 num1 = 0.02f \* 5.0f num0 != num1

Float은 합연산과 차연산을 구분 하지 못하는 경우가 있다  
따라서 정밀한 값을 요구하는 경우에는 부동소수점을 사용하지 말고 고정 소수점을 사용할 것

c언어에서는 함수 내부에서 함수 외부의 변수에 값을 조정하기 위해선   
&를 변수 앞에 붙여줘야 함

scanf는 정수를 리턴한다 리턴값은 scanf 함수 내부에서 제대로 입력받아 값을 대입한   
변수의 개수 이다.

scanf는 입력받는 값에 문자열등을 넣어 형식을 지정할 수 있으며 콘솔에서 값을 입력할 때  
해당 형식과 같은 문자열등을 넣어야 변수에 값을 제대로 지정할 수 있다  
  
long float과 float은 다른곳에선 같다고 볼 수 있지만 scanf에서는 다르기 때문에 확실히 구분을 하여 사용하여야 한다

getchar()함수는 문자에 남아 있는 버퍼값 문자 하나를 가져간다

cout은 “<<” 연산자를 오버로딩해서 뒤의 값을 어떻게 출력할 지 정하는 것

**연산자 오버로딩?**

cout << ‘a’ << endl (cout으로 흐른다 라고 생각하면 편하다 함)

cout.setf (출력 포맷 지정함수 지정해제함수는 cout.unsetf)  
ios\_base::showpos (출력하는 대상이 수 이고 양수라면 +표시)

‘=’(할당 연산자)는 왼쪽 변수의 오른쪽 결과를 할당시킬 뿐만 아니라 할당시킨 값 자체를  
리턴시키기도 한다

할당과 초기화는 다르니 구분하도록 하자

std::size(배열) or sizeof(배열명) / sizeof(배열자료형) 으로 배열의 사이즈를 구할 수 있다

C++에서는 배열에 배열로 할당이 되지 않는다 (arr1 = arr2 같은 방식이 되지 않음)  
(배열 변수자체는 값이 아닌 주소값을 가지고 있기 때문?

배열복사: memcpy(할당받을배열, 할당할배열, 사이즈); (각 배열앞에 &를 붙여도 됨)  
or  
std::copy(할당할 배열 시작 주소, 할당할 배열 끝 주소, 할당 받을 배열) (배열 앞에 &를 붙이면 x)

C++의 문자열은 문자열의 시작 주소부터 문자들을 읽어들여 null문자(혹은 (숫자)0 )가   
나올때까지의 연속된 주소들이다  
  
C++ 문자열 char str = “abc’ 이 있을 때 str[1] = 0 을 할당하고  
cout << str; 을 실행하면 a만 출력된다 (str의 2번째 값에 0이 들어가서 문자열을 a까지만 인식)  
  
cout.precision(n) 소수 n번째 자리까지 표기(가능하다면)  
  
문자열을 초기화 한 후 맨 뒤 값에 강제로 문자를 추가한 후 해당 문자열을 출력하면  
기존의 문자와 추가한 문자까지는 정상출력하나 이후 우연히 null 문자를 만나기 전까지 쓰레기  
값을 출력한다.  
  
c++의 배열의 변수들은 죄다 주소값? 인듯 하니 변수끼리 직접 비교를 하면 안된다  
char str1; char str2의 내용만을 순수하게 비교하고 싶다면  
strcmp(str1,str2);를 사용해야 한다 (왼쪽 문자열이 사전상에서 먼저 나온다면 -1, 같으면 0, 왼쪽 문자열이 사전 상에서 나중에 나온다면 1)  
  
문자열 할당은 strcpy(그러나 메모리 침범이 발생할 수 있으므로 주의해서 사용)  
(4996에러 처리 필요)  
문자열 연결은 strcat(str1,str2) (str1 바로 뒤 주소값에 str2를 연결)  
문자 to int,float 형변환 => atoi, atof  
공백을 무시하고 한줄 전체를 입력 받을땐 cin.getline(문자열변수, 수량)  
-> 실제 받을 글자보다 하나 더 커야함(공백 문자 때문)

수 to 문자 => sprint(arr1,”%d(%f),숫자);

구조체 내부에 구조체를 쓰려면 반드시 내부 구조체가 먼저 선언되어야 한다

C++ 구조체는 구조체 끼리의 냅다 할당이 가능(메모리를 통으로 카피)(냅다 비교는 X)

C++ 구조체는 메모리 구성을 할 때 가장 큰 바이트의 배수로 구성하는데  
생성순서에 따라 결과가 다름  
ex) float(4) flaot(4) char{10}(1 \* 10) short{2} -> 4바이트가 제일 큼(1\*10의 배열은 1)  
따라서 4바이트씩을 한칸? 기준으로 메모리를 구성   
4(한칸) + 4(한칸) + 10(3칸중 2칸 꽉채우고 2바이트 남음) + 2(남은 2바이트에 들어감) = 20바이트  
float(4) short(2) float(4) char[10](1 \* 10)  
역시 제일 큰 사이즈는 4바이트라서 4바이트를 한칸으로 구성  
4(한칸) + 2(한칸 한고 2바이트 남음) + 4(남은 2바이트 안에 들어갈 수 없으므로 새로 한칸) +   
10(총 3칸이 필요 하므로 12바이트) = 24바이트

Alighof(구조체): 이 함수는 해당 구조체의 기준 메모리가 몇인지 알려준다  
  
공용체 값은 하나인데 형식이 여러 개일 경우에 사용한다.  
(ex:: 제품코드: 문자열일수도 혹은 정수형 일수도 있다)  
  
C++에서 enum은 따로 변수로 생성하지 않고 냅다 가져다 쓸 수 있는거 같다

C++은 무조건 함수나 변수나 다른곳에서 사용하기 위해선 먼저 선언되어 있어야 한다

Enum을 사용할 때 c#은 이넘.값 이지만 c++은 이넘::값 이다  
  
enum class 열거형 or enum struct 열거형과 같이 범위 지정 열거형의 경우  
사용할 때 냅다 열거타입으로 사용할 수 없고 열거형::열거타입 과 같이 사용해야 한다  
  
범위지정 enum의 경우 냅다 정수 계산이 안됨 (int)형변환을 해줘야 함  
  
enum의 내부 데이터의 값은 기본 int형이지만 enum 이름 : int64\_t 같은 형식으로 더 넓은 범위의 int로 지정해 줄 수 있다  
  
std::array<자료형,길이> arr{}; 으로 선언 및 초기화(선언시 반드시 길이를 지정해야 함)  
  
arr.at(n) 을 사용했을 때 n이 arr의 범위를 벗어난 값인 경우 좀더 명확하게 오류를 알 수 있다.

str::array arr의 자체 함수  
사이즈를 리턴: arr.size()  
처음 요소를 리턴: arr.front()  
마지막 요소를 리턴: arr.back()  
서로 요소를 스왑: arr..swap(arr1)  
냅다 할당, 비교 가능: arr = arr1, if(arr == arr2) // 단 std::array끼리의 사이즈가 같아야 함  
=> 비교는 맨 앞 인자부터 서로 비교하고 모든 인자가 같으면 == 비교가 True를 리턴한다.

for(자료형 이름 : 연결자료형) 으로 해당 연결자료형안의 정보들 전체를 순회 조회 할 수 있다.  
그러나 이렇게 범위 지정으로 접근하면 주소로 접근하지 않고 값으로만 접근한다.  
주소로 접근하기 위해선 for(자료형& 이름 : 연결자료형)으로 접근하면 된다

포인터자료형 변수에 자료형이 지정되는 이유: 포인터 변수는 주소값을 가지는 변수  
포인터변수를 역참조 한다 = 포인터변수가 가진 값(=주소값)에 접근하여 값을 가져온다  
따라서 해당 주소값이 가진 값을 알맞게 처리하고 읽기 위하여 포인터 변수도 자료형을 지정한다.  
  
const int\* pointer = &num의 경우 가지고 있는 주소값을 변경하지 못하는 것이 아닌  
주소를 역참조하여 값을 바꾸는 것이 불가능 한 것  
const int\* = int const\*이다

Int\* const는 역참조한 값을 상수화 하는 것이 아닌 포인터변수가 가진 주소값을 상수화 한다.

const int\* const num 이런식으로 사용할 경우 역참조 값과 가지고 있는 주소값 모두 상수화 한다.