5 배열과 문자열

5-1. 배열이란?

5-1-1. 배열의 선언

5-1-2. 배열의 메모리

5-1-3. 배열의 사용

5-1-4. 다차원 배열

5-2. 문자열이란?

5-2-1. 문자열과 다른 배열의 공통점, 차이점

5-2-2. 문자열 처리 함수

5-2-2-1. strcpy

5-2-2-2. strlen

5-2-2-3. 그 외의 함수들

5-2-2-4. gets, puts

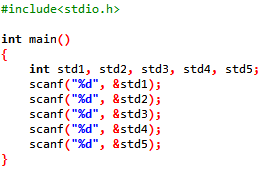
5-2-3. 다차원 문자열 배열

5-1. 배열이란?

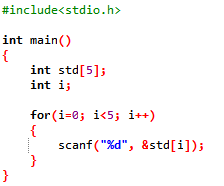
변수가 재료를 담는 상자라면, 배열은 상자의 묶음이라 이해하면 편하다.

(그림. 아이가 여러 개의 상자를 묶어 놓은 그림)

우선 배열이 필요한 이유에 대해서 알아보자. 변수를 사용하는 경우, 5명의 학생의 점수를 입력 받아 저장하고 사용하려면 적어도 다섯 개의 변수가 필요하다.



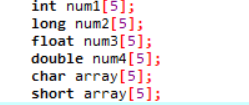
코드를 쓰기 귀찮기는 하지만, 이 정도는 할 만 한 것 같다. 그런데 우리가 실제로 프로그램을 만들 때는 감당하기 힘들만큼 많은 숫자를 다루게 된다. 천 개, 만 개 정도 되는 변수들을 사용하려 하는데 위와 같이 일일이 변수를 선언하는 것은 매우 비효율적인 방법이다. 바로 이럴 때, 배열을 사용하는 것이다.



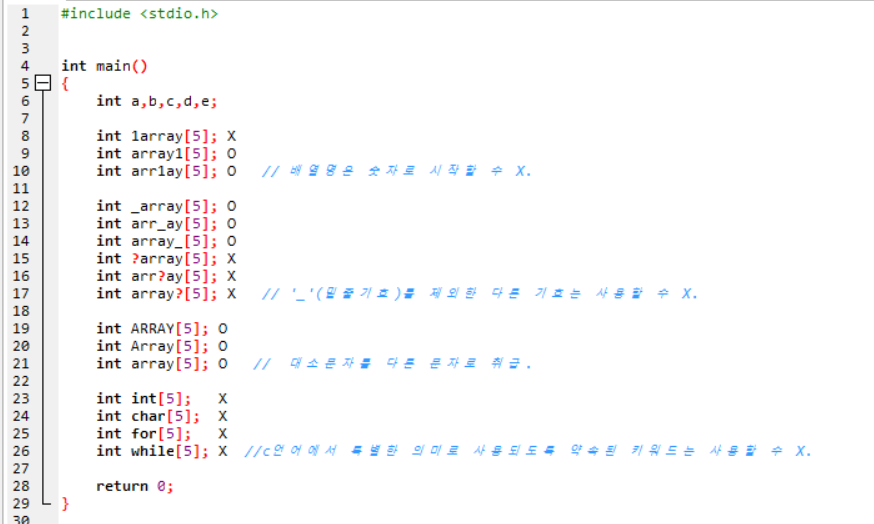
위 코드를 보면, 크기가 5인 배열 std 를 선언해서 한 번에 다섯 개의 변수를 선언한다. 입력도 for 문을 통해 훨씬 효율적으로 받을 수 있음을 알 수 있다.

5-1-1. 배열의 선언

배열은 변수의 확장 개념이기 때문에 선언하는 방법이 변수와 매우 비슷하다. 아래의 예시를 통하여 참고하자.



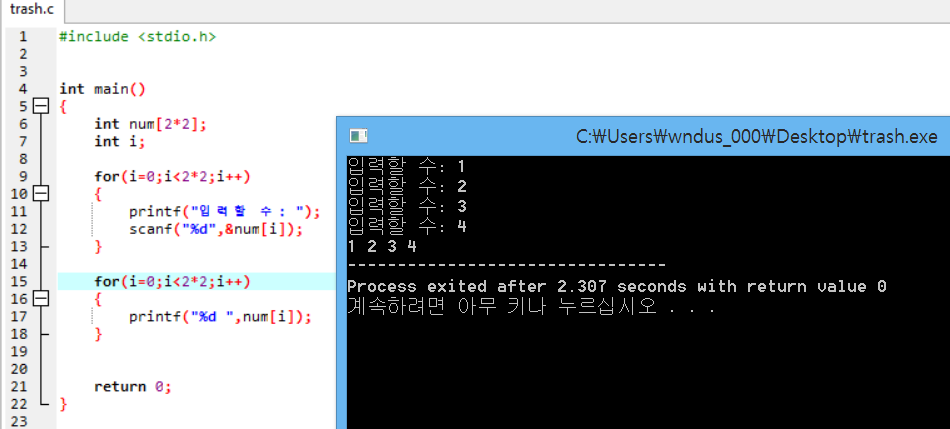
변수의 선언과 비슷하게 데이터 형과 배열 명을 선언해야 한다. 변수 이름을 정할 때 사용하면 안되는 이름이 있고, 특수 문자 사용 시 규칙이 있듯이 배열에서도 동일한 규칙이 적용된다.



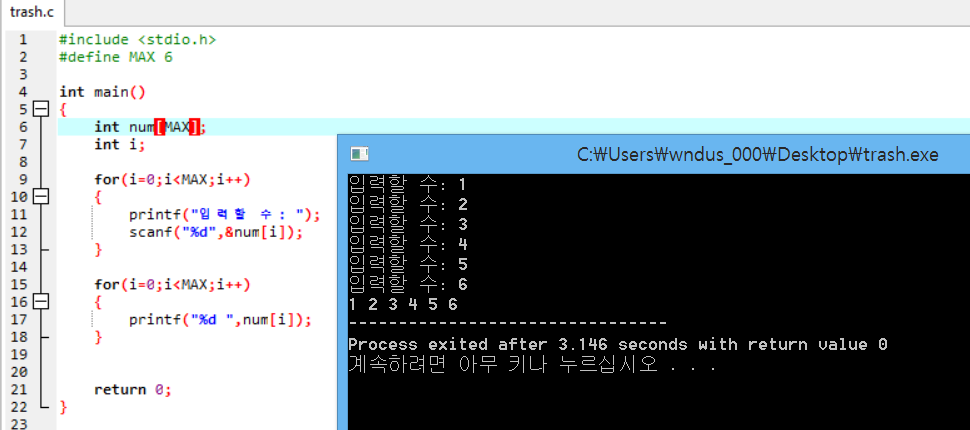
하지만 변수의 선언과는 다르게, 배열의 선언에서는 배열의 크기를 표시해 준다. 배열의 크기란 배열 안에 들어갈 수 있는 변수의 개수를 말한다.

(그림: 배열 선언->컴퓨터가 변수를 여러 개 저장하는 모습을 보여줌)

그리고 이러한 배열의 크기를 선언할 때에는, 반드시 0보다 큰 정수로 지정해야 한다. 배열의 크기로는 매크로 상수나 수식을 사용할 수 있다.



배열의 크기를 ‘2\*2’ 즉 4로 지정하는 것은 가능하다.



매크로 상수 MAX를 사용하여 배열의 크기를 6으로 정했다.

5-1-2. 배열의 메모리

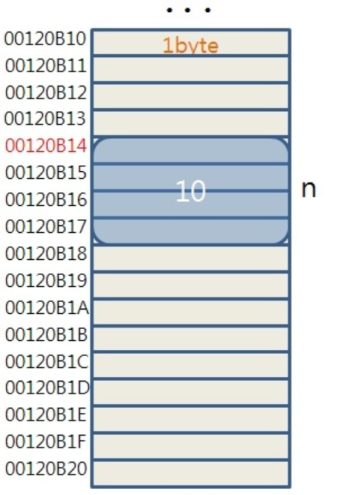
변수의 설명에서 잠깐 언급하였지만, 변수를 상자로 비유하자면 작업실(메인 함수)에서 작업을 할 때 그릇의 이름과 내용도 물론 중요하지만, 상자가 어디에 놓여있는지, 즉 그릇의 위치도 중요하다. 그래야 작업할 때 컴퓨터가 편리하기 때문이다. 이 과정을 실제로는 컴퓨터가 변수를 어느 메모리에 저장하였는지 그 메모리의 주소 값을 기억한다고 말한다. 메모리의 주소가 변수가 어디 있는지 알려주는 지표인 것이다.

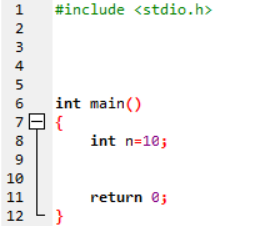
(그림: 사람이 상자를 놓으며 “그래 이 ‘사과를 담은 상자’는 식탁 위에 놨어!”

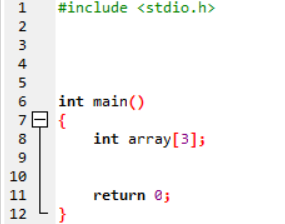
vs

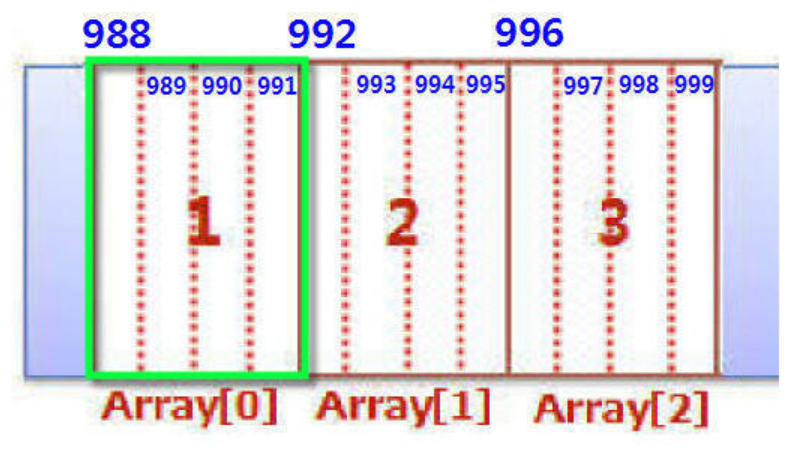
(그림: 컴퓨터가 변수를 저장하며”그래 이 int 형 변수 a는 12395라는 주소 값을 지닌 메모리에 놨어!”)

변수가 그렇듯이 배열 또한 어딘가 메모리에 저장될 것이고, 그 메모리는 당연히 주소 값을 가진다. 그렇다면 배열의 메모리는 어떻게 저장될까? 배열은 변수의 모임이라는 것을 기억한다면 쉽게 이해할 수 있다.

 그릇 묶음을 어느 위치에 놓던 묶음 안의 그릇들끼리는 항상 인접한다. 비슷하게 배열의 각 원소의 주소 값 또한 연속되어있다. 배열의 메모리는 배열 원소들의 메모리의 합과 같다. int 형 변수를 5개 가진다면 그 메모리는 당연히 20bytes이다! ( 4bytes(int 형 변수의 메모리 크기) \* 5 = 20bytes)

(변수 하나를 선언했을 때)

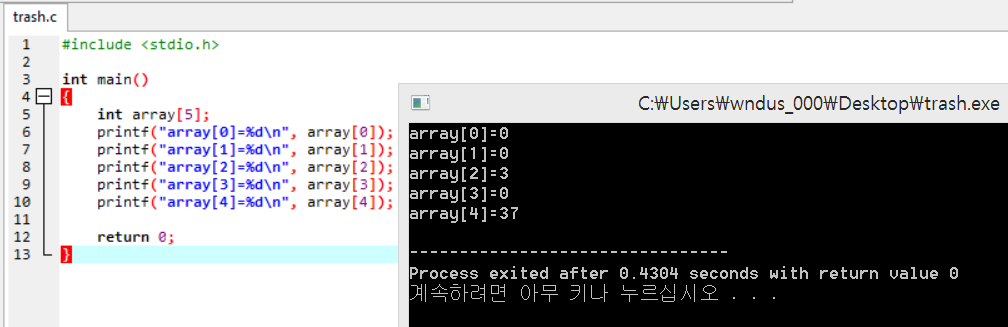


(크기가 3인 배열을 선언했을 때)

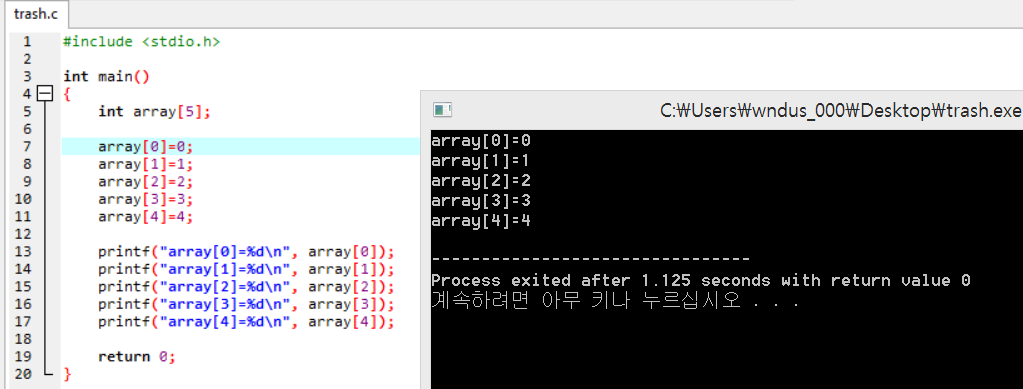
배열에서 배열의 이름은 배열의 주소를 말하기도 한다. 즉 array배열에서 array는 array 배열의 시작 주소를 의미한다. 위 그림에서는 “998 = array = 배열의 시작주소(&array[0])”라고 할 수 있는 것이다. 여기서는 ‘배열의 이름=배열의 시작 주소’ 라는 것만 알아 놓고, &에 대한 설명은 포인터 단원에서 더 자세히 배우도록 하자.

5-1-3. 배열의 사용

배열도 변수이므로 처음 선언 후 초기화를 하지 않으면 쓰레기 값을 지닌다.

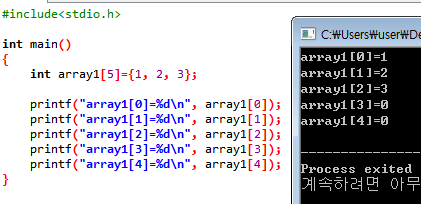


초기화 방법은 간단하다. 처음 선언부에 아래의 그림과 같이 초기화 해주면 된다.

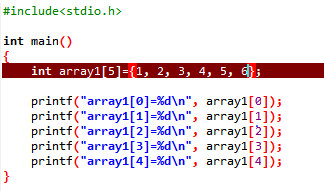


이 때, 배열 크기보다 적은 수만 초기화하면 나머지 원소들은 자동으로 0으로 초기화 되고, 더 큰 수의 데이터를 초기화하면 컴파일 에러가 뜬다.

(배열 크기보다 적은 수를 초기화한 경우)

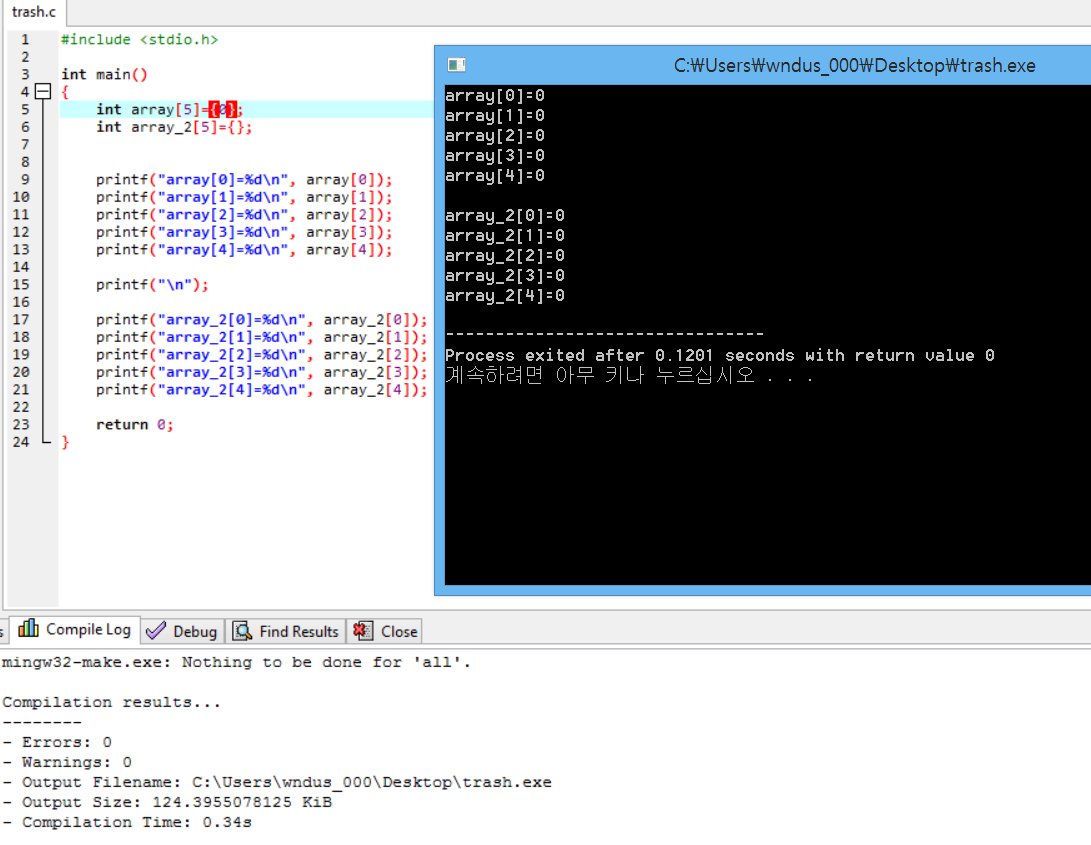


(배열 크기를 넘어서 초기화한 경우)

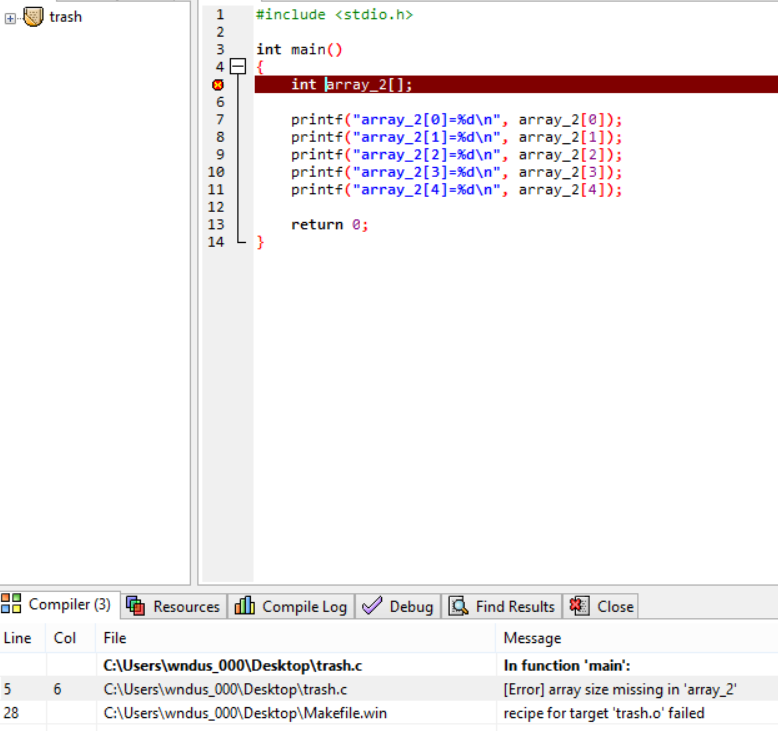
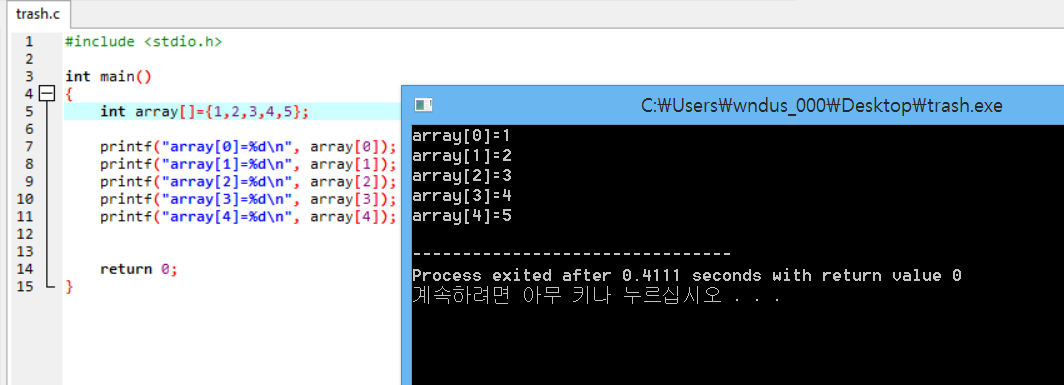


배열을 선언하는 부분에서 에러가 뜨게 된다.

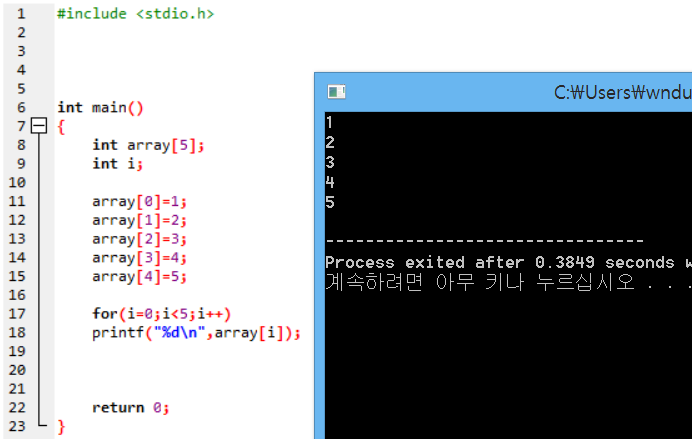
배열 전체를 0으로 초기화하려면 아래의 그림과 같이 ‘{0}’ 혹은 ‘{}’을 써주면 된다.



마지막으로, 배열의 선언과 동시에 초기화를 해주는 경우, 배열의 크기를 입력하지 않아도 입력한 개수만큼의 크기가 자동으로 할당된다. 하지만 초기화 없이 크기를 생략한다면, 컴파일 에러가 발생한다.

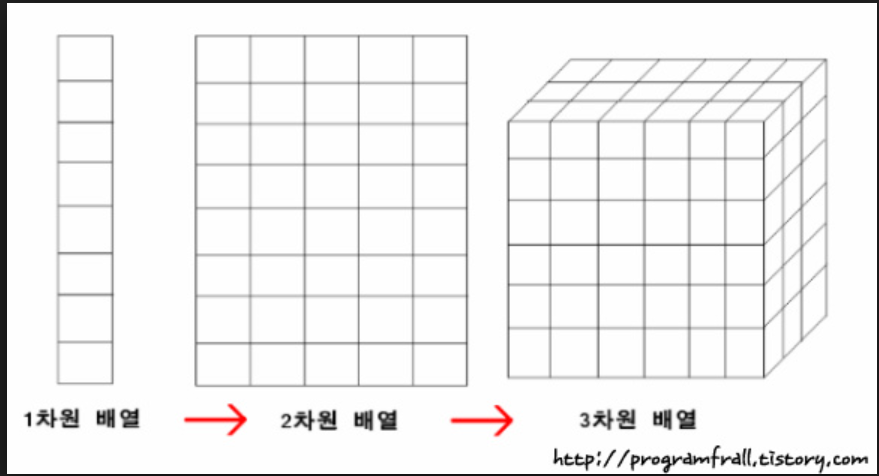


배열의 선언에 대하여 자세히 알아봤으니, 이제 본격적인 사용법을 알아보자. 배열의 사용법은 변수와 크게 다르지 않다. 배열의 각 원소를 변수 사용하듯이 사용하면 된다. 하지만 주의할 점이 있다!! N개의 원소를 선언하면, 배열의 맨 처음 원소의 번호는 0이기 때문에 N-1번째까지 사용할 수 있다. 아래는 N=5인 배열의 코드이다.

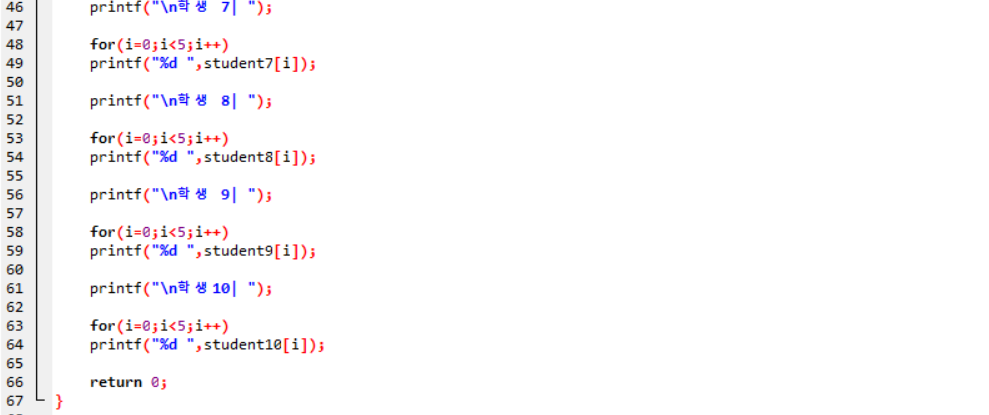
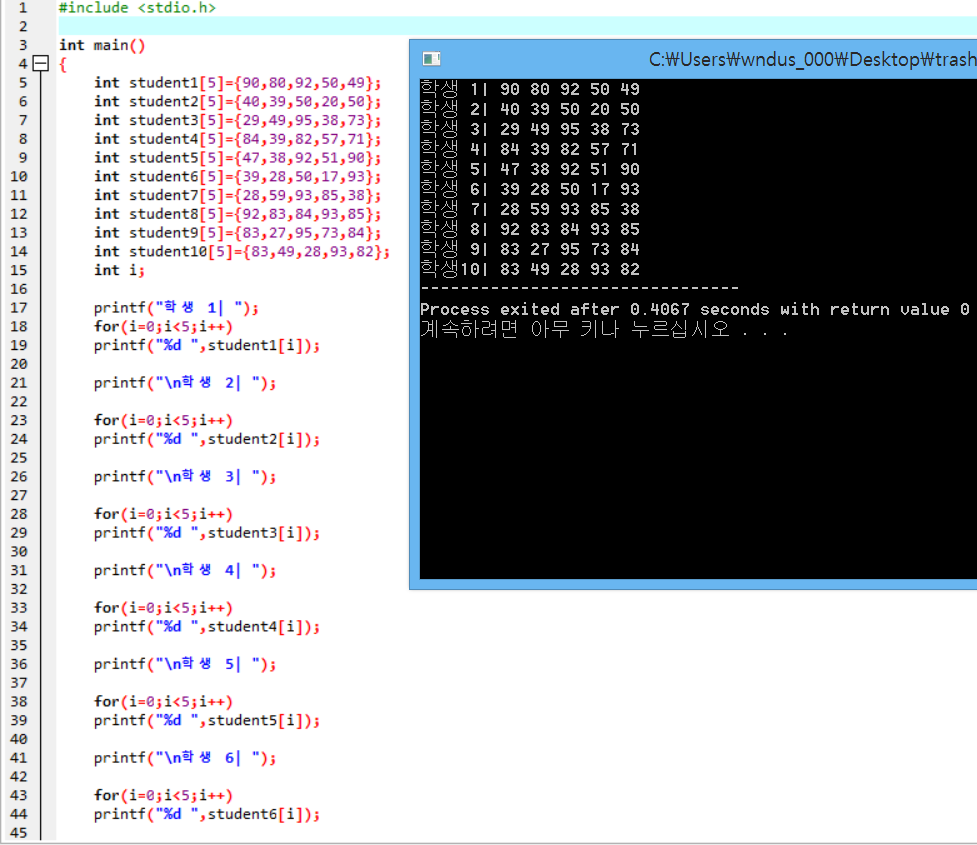


5-1-4. 다차원 배열

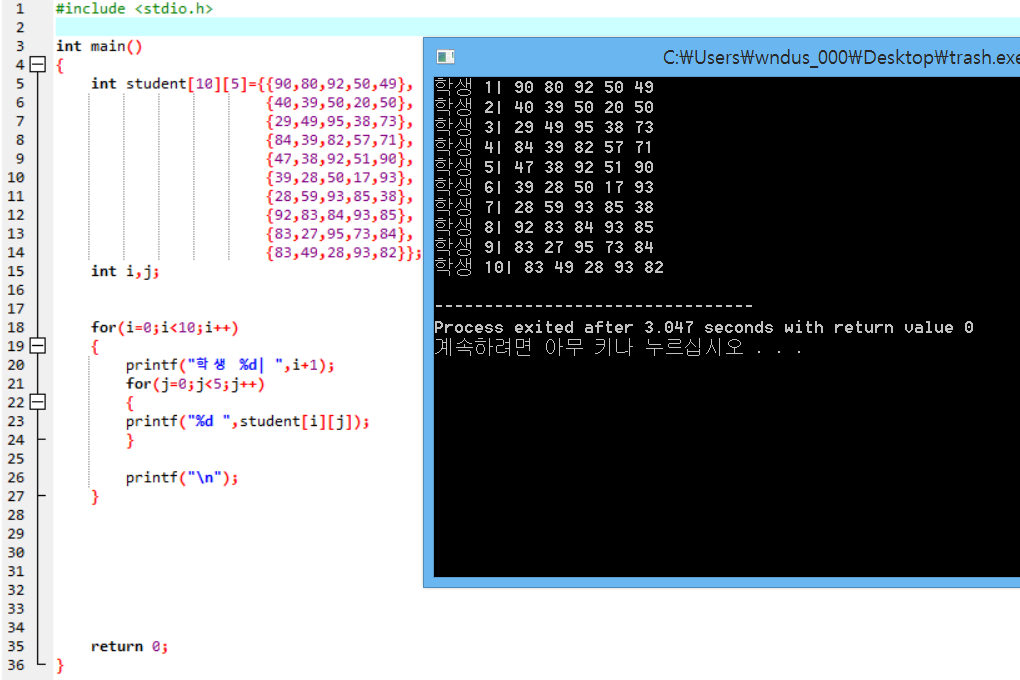
지금까지 일차원 배열의 정의와 배열의 사용법을 살펴봤다. 그렇다면 일차원 배열에서 더 나아가 다차원 배열에 대해 알아보도록 하자. 뭔가 이름이 어렵지만, 사실 아무것도 아니다! 여태까지 일렬로 나열된 배열을 다루었다면, 이제부터 하나의 평면으로 나열된 배열을, 혹은 입체적인 배열을 다루는 것이다. 마치 아래의 그림과 같이!



그렇다면 이러한 배열은 언제 사용하면 유용한 것일까? 아래의 학생 10명의 5과목의 성적을 저장한 코드의 예시를 보자.



위 코드처럼 일차원 배열을 사용하려면, 배열을 최소한 5개를 선언해야 한다. 하지만 이차원 배열이라면 한번에 해결이 가능하다.



한편, 이차원 배열의 초기화 역시 일차원 배열의 초기화와 크게 다르지 않다.



array1 의 경우 array1[0][0], array1[0][1], array1[1][0], array1[1][1] 순으로 1, 2, 3, 4가 들어가고 array2 역시 array1[0][0], array1[0][1] 에 1과 2, array1[1][0], array1[1][1] 에 3과 4로 초기화된다.

유의할 점 역시 일차원 배열과 비슷하다. 배열로 할당한 개수보다 적은 수의 데이터를 초기화하면 나머지는 자동으로 0으로 초기화 되고, 더 큰 수의 데이터를 초기화하면 컴파일 에러가 뜬다. 다음은 배열로 할당한 개수보다 적은 수의 데이터를 초기화 했을 경우를 나타낸 그림이다.

1,2,3

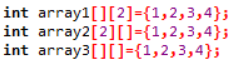
4,0,0



1,2,0

3,4,0

일차원 배열처럼 배열을 초기화할 때 행과 열 중에 배열의 첫째 값인 행의 값을 생략할 수 있다. 이 때, 열의 값을 생략하거나, 행과 열을 모두 생략할 수 없다.



array1의 경우는 행( | )의 수를 모르더라도, 열의 개수만큼 원소를 배치하면 행의 값이 항상 같다.

array2의 경우는 열(ㅡ)의 수를 모르므로, 각 행의 몇 번째 열까지 원소가 배치되어있는지 알 수 없어서 열의 값이 하나로 정해지지 않는다.

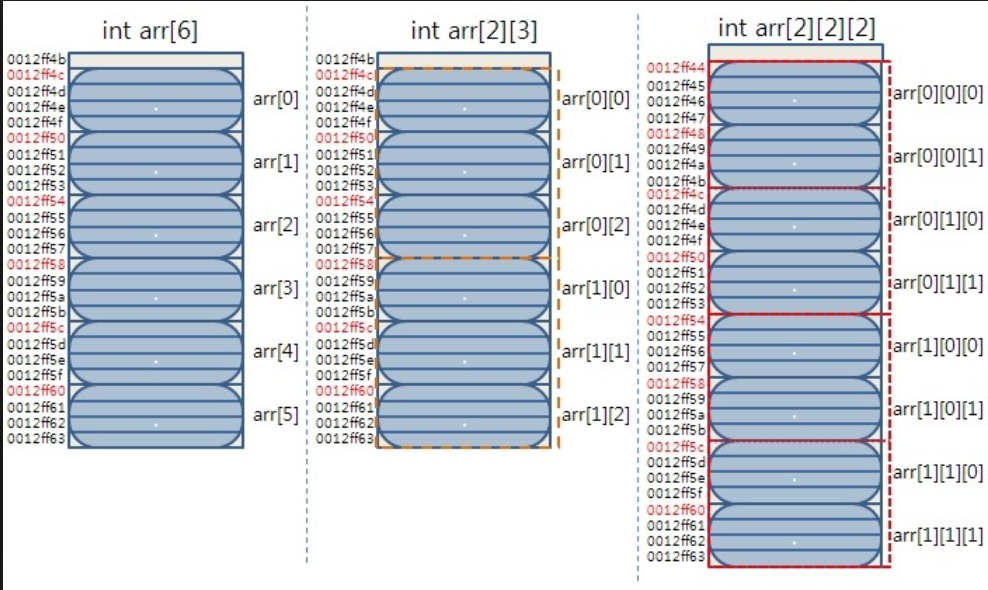
array3의 경우는 행( | )의 수를 모를 뿐만이 아니라 열(ㅡ)의 개수도 알 수 없으므로 행렬이 될 수 있는 경우의 수가 무수하다. 예를 들어 {{1,2,3},{4,0,0}}, {{1,2,0},{3,4,0}}, {{1,0,0},{2,3,4}}, {{1,0},{2,0},{3,0},{4,0}}등 행렬의 모양을 예측할 수 없다.

지금까지 일차원, 이차원배열에 대해 공부해보았다. 위에서도 언급했다시피 일, 이차원 뿐만이 아니라 삼차원 배열을 포함한 다차원 배열을 만들 수 있다. 다차원배열의 원소의 개수는 각 크기를 모두 곱한 값이므로 고차원이 될수록 담을 수 있는 데이터의 개수는 급증한다.

다차원 배열의 초기화에서 유의할 점은 이차원 배열과 마찬가지로, 배열의 첫 번째 크기만 생략할 수 있다는 점이다.



다차원 배열이 메모리에 저장되는 모습은 이차원배열의 확장이라고 할 수 있다.



5-2. 문자열이란?

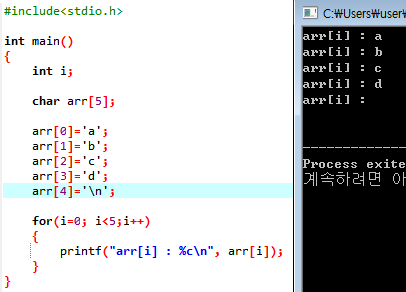
문자열이란 char 데이터 형의 배열을 의미한다. 연속된 문자들의 모임 그 이상도 이하도 아니다. 그러면 왜 수많은 배열 중 문자열만 따로 단원을 만들었을까? 그 이유는 문자열이 다른 배열에 비해 굉장히 많이 사용되기도 하고, 추가적인 설명이 필요한 유별난 배열이기 때문이다.

5-2-1. 문자열과 다른 배열의 공통점, 차이점

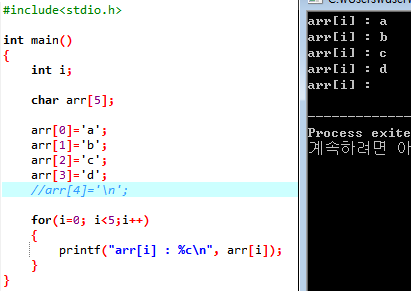
다른 종류의 배열들과 마찬가지로 문자열 또한 문자 변수들의 집합이다. 메모리 주소도 연속되어 있고, 선언방법 또한 같다. 다만 유의해야 할 점이 있다. 바로 배열의 끝에 반드시 널 문자"\0"를 저장해야 한다는 것이다. 만약 끝에 널 문자를 저장하지 않는다면, 컴퓨터가 알아서 배열의 마지막과 비어있는 원소에 널 문자를 저장한다.

그러므로 항상 문자 배열을 선언할 때에는 널 문자를 넣을 공간을 마련해줘야 한다. 즉 저장할 문자열의 길이보다 1만큼 크게 배열의 크기를 선언해야 한다.

(마지막에 널 문자를 집어넣는 경우)

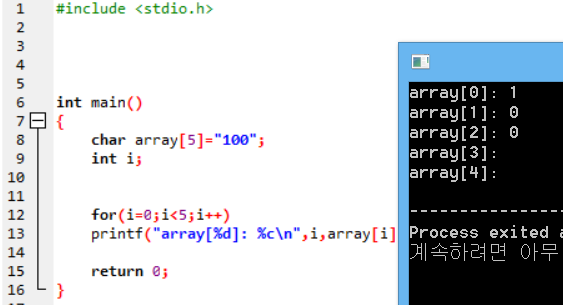


(마지막에 널 문자를 넣지 않는 경우)



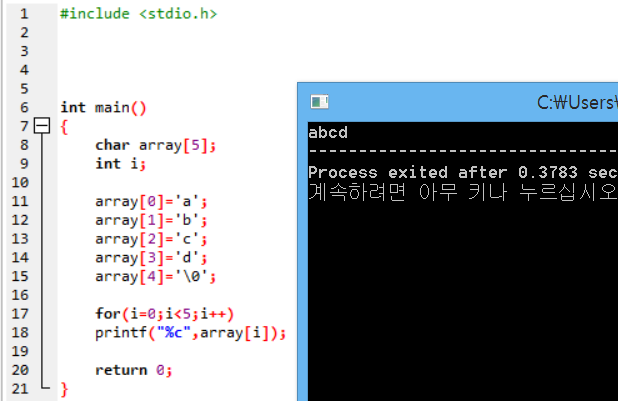
널 문자를 넣은 경우와 안 넣은 경우 모두 같은 결과가 나옴을 알 수 있다. 한편, 널 문자로 초기화하는 것 대신 arr[4]=0; 과 같이 작은 따옴표를 쓰지 않고, 0으로 초기화 해주면 아스키 코드에 의한 널 문자로 초기화 할 수 있다.

이외의 차이점으로는, 배열 원소들을 초기화하는 방법이 다르다. 아까 배열에서는 배열의 각 원소들을 따로따로 초기화했었다. 하지만, 문자열에서는 구성요소를 일일이 입력해도 되지만, 큰따옴표를 이용하여 한 번에 초기화할 수 있다. 초기화되지 않은 배열의 구성요소는 알아서 널 문자가 채워진다.

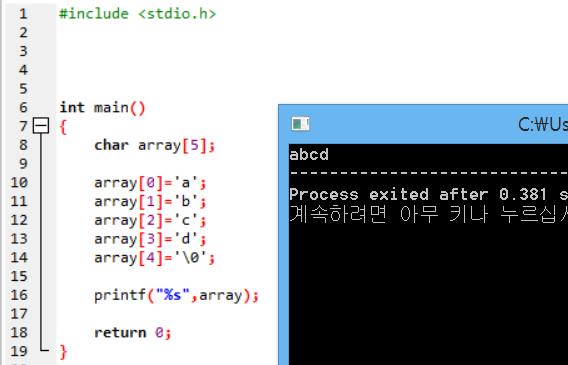


문자열은 원소 하나만으로 구성되지 않는다. 때문에 printf에서 %c를 사용하여 문자열을 출력하려고 하면 맨 앞의 문자만 출력이 된다. 이를 해결하기 위해 조건문을 사용하거나 printf를 여러 번 입력하는 방법으로 문자열을 출력할 수 있을 것이다. 이보다 더 쉬운 방법으로는 printf에서 형식문자열로 %s를 사용하고 문자열의 이름을 출력해주면 된다.(이 때 크기는 입력하지 않는다)

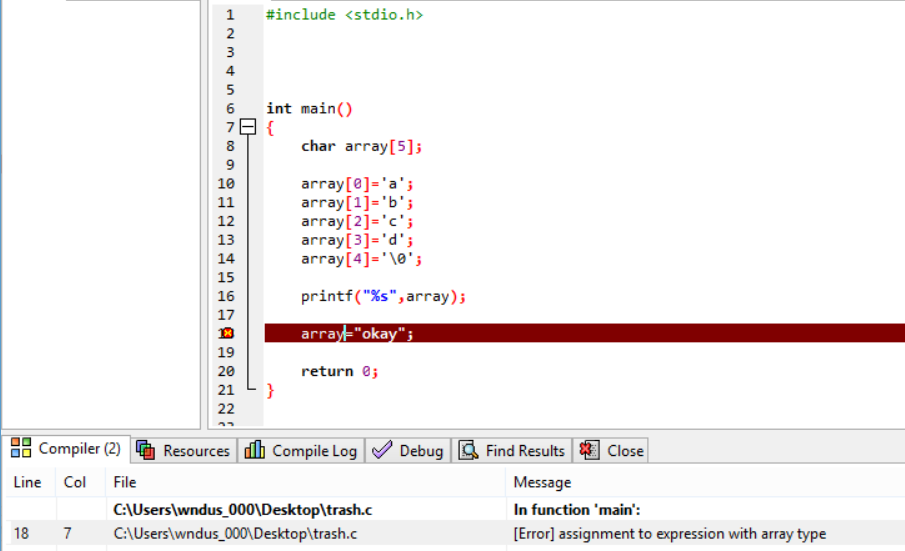
(for 문을 이용하여 문자열을 출력하는 방법)



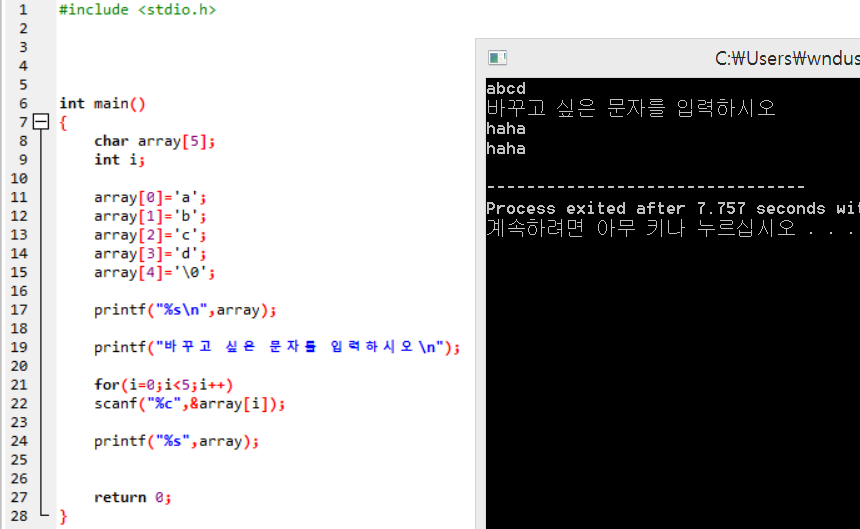
(%s 를 이용하여 문자열을 출력하는 방법)



그렇다면 문자열을 다른 문자열로 바꾸고 싶으면 어떻게 해야할까? 변수에 대입할 때 처럼 그냥 대입하면 되는걸까? 그렇지 않다.



배열의 이름에 무턱대고 대입을 하면 컴퓨터가 위치에 대한 정확한 정보가 없으므로 이렇게 에러가 발생한다. 따라서 시작 위치를 컴퓨터에게 알려주어야 한다.



for문에서 array 배열의 첫 요소부터 다시 입력을 받는다.

이렇게 하면 문자열을 변경할 수 있지만, 매번 문자열을 바꿀 때 마다 이 방법을 쓰기에는 너무나 불편하다. 그래서 사람들은 문자열에 대한 새로운 헤더를 만들어서 문자열을 다루기 위한 여러 도구들을 만들었다.

5-2-2. 문자열 처리 함수

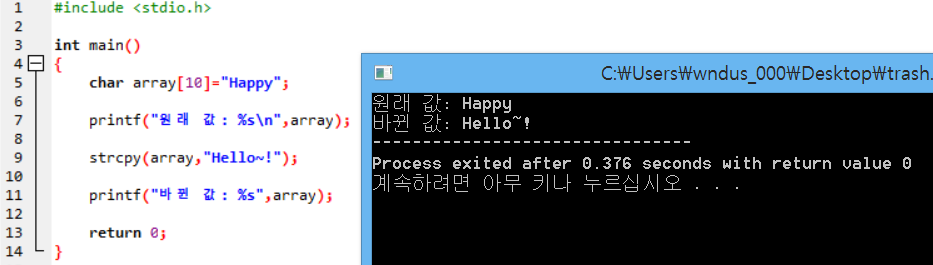
우리가 지금까지 사용해본 헤더는 <stdio.h>뿐일 것이다. 오늘부터 문자열처리 함수의 헤더 파일은 <string.h>이라는 것 한가지만 더 기억하도록 하자. string이 ‘줄’이라는 의미를 가지고 있듯이 문자열이 한 줄로 이어져있는 모습을 떠올릴 수 있을 것이다. 그럼 문자열 처리 함수에 어떤 것들이 있는지 알아보자.



5-2-2-1. strcpy

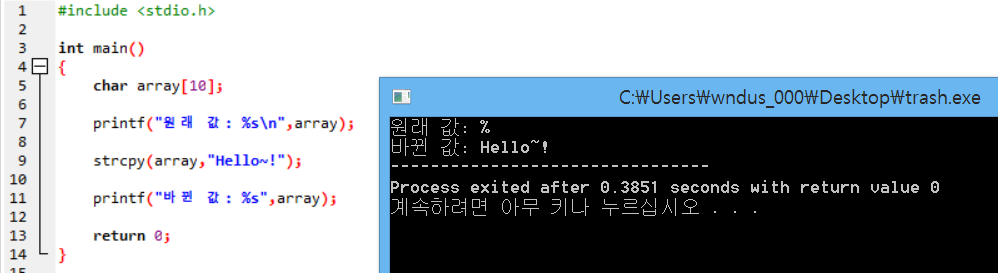
str + cpy = string을 copy하다.

“str”은 string의 줄임 말이고, “cpy”는 copy의 줄임 말이다. 즉, strcpy의 기능은 문자를 복사하는 것이다. 이 함수는 문자열을 초기화한 상태에서, 문자열의 요소를 하나하나 바꾸는 대신에, 바꾸고 싶은 문자열로 한번에 바꿀 수 있는 기능을 지닌다. 이제 strcpy를 사용하는 방법을 알아보자.



원래의 문자열을 ( ) 안의 앞부분에 입력하고, 뒷부분에는 바뀐 결과값이 되어야 할 문자열을 입력한다. 이 때, 바뀐 결과값이 되어야 할 문자열의 길이는 선언한 길이 값보다 커질 수 없다

문자열을 선언만 하고 내용을 입력하지 않은 상태에서도 strcpy함수를 사용할 수 있다. 사용자가 문자열을 초기화하지 않았어도, 이미 쓰레기 값이 담겨있기 때문이다.

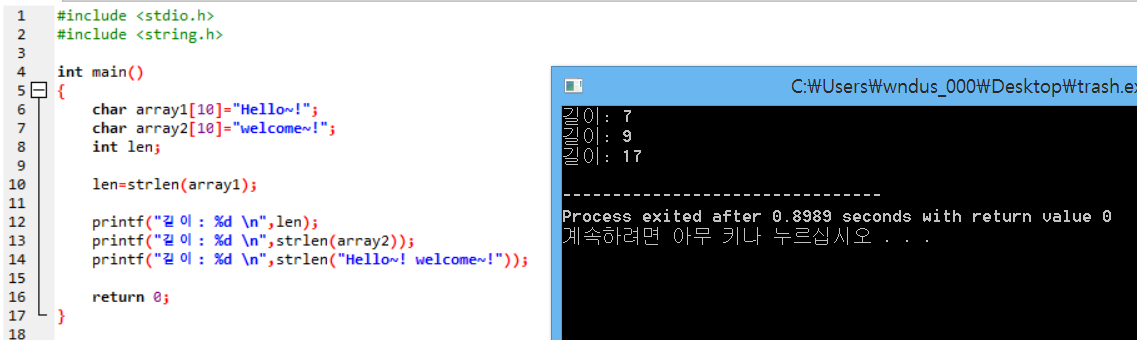


5-2-2-2. strlen

str + len = string의 length(길이)를 구하다.

“str”은 위에서도 언급했다시피, string의 줄임 말이고, “len”은 length(길이)의 줄임말이다. 자, 이제 한번 이 함수의 기능을 추측해보자. 그렇다. 이 함수의 기능은 문자열의 길이 값을 구하는 함수이다. 이제 strlen를 사용하는 방법을 알아보자.

괄호 안에 길이를 구하고 싶은 문자열을 입력하면 된다. 문자열배열의 길이를 구할 때는 “”없이 배열의 이름만 입력하면 된다. 배열이 아닌 문자열의 길이를 구할 수도 있는데, 이 때는 “”안에 문자열을 입력해야 한다.



5-2-2-3. 그 외의 함수들

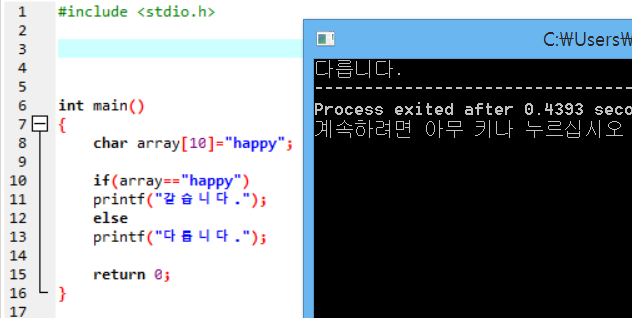
ㄱ. strcmp

str + cmp = string을 compare(비교하다).

이제 다시 적기에도 손이 아프다. “str”이 무슨 이미라고? string!! “cmp”는 compare(비교하다)의 의미를 가지고 있다. 그렇다면 이 함수의 기능은 무엇일까? 문자열을 비교하는 함수이다. 문자열을 비교하는 함수가 있는 이유가 무엇일까?

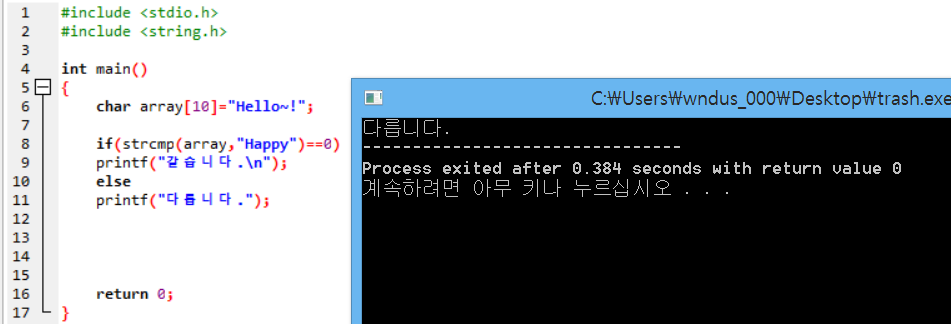
코딩을 하다 보면, 조건문을 통해 문자열이 같은지 다른지 비교해야 하는 상황이 생긴다.

예를 들어,



이런 상황이 닥쳤을 때, 필요한 함수가 strcmp이다. 이 함수가 필요한 이유도 알았으니, 이제 strcmp를 사용하는 방법을 알아보자.

괄호의 앞부분에는 비교해야 할 문자열 배열을 입력하고, 뒷부분에는 비교가 될 문자열을 입력하면 된다.



ㄴ. strcat

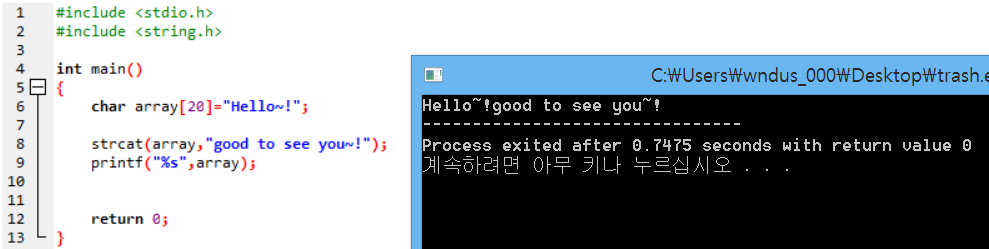
strcat = str + concatenation(연속) 이지만 concatenation이라는 단어가 어렵기 때문에 쉽게cat은 catch로 봐도 무방할 것 같다.

strcat = string의 끝을 catch(잡)다. 라고 이해하도록 한다.

원래 있던 문자열에 새로 입력 받은 문자열을 연결하는 함수이다. 입력 받은 문자열이 원래 있던 문자열의 끝을 잡는 모습이 연상이 가지 않는가?

(그림: str이라적힌 배열상자의 끝부분을 cat이라는 배열상자가 잡는 모습)

strcat를 사용하는 방법은 괄호의 앞부분에 문자열배열을, 뒷부분에는 덧붙일 내용을 적으면 된다.



5-2-2-4. gets, puts

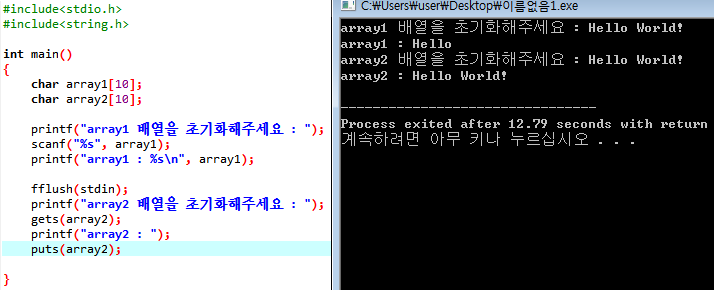
문자열 배열은 입출력을 하는 함수도 따로 있다. scanf, printf 가 이미 있는데 똑같은 함수를 왜 만들었는지 의문이 들 것이다. 먼저 함수들의 특징 및 유용성을 보자

>gets함수?? 뭘…. 받는 다는 의미일까??

scanf와 같이 입력을 받는 함수를 gets라고 한다. gets는 얻는다, 받는다는 의미를 지니므로 외우기 쉬울 것이다.

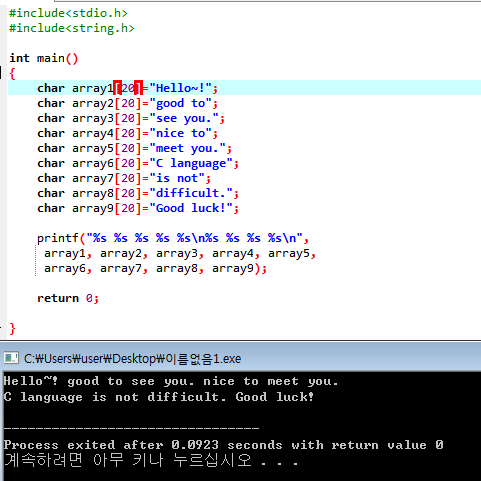
scanf 함수는 입력을 받을 때, 빈칸(스페이스바)을 문자로 인식하지 않는다. 따라서 빈칸을 포함한 문장을 입력 받을 때는 불편함이 많다. 반면, gets 함수는 빈칸을 포함한, 한 줄의 문자열을 입력 받는 데 사용된다. 따라서 문장을 입력 받을 때 유용하다. 뿐만 아니라, 함수 이름 안에 배열명만 적으면 되므로 간편하다.

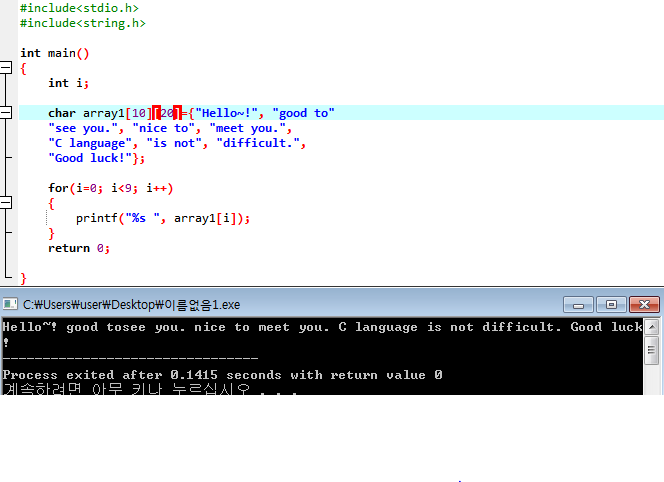
>puts함수? 뭘…. 놓는다는 의미인건가….??

 printf와 같이 출력을 하는 함수를 puts라고 한다. put은 놓는다는 의미를 지니므로, 입력을 통해 받은 문자열을 출력하기 위해 놓는다고 생각하면 외우기 쉽다. puts함수는 빈칸을 포함한 한 줄의 문자열을 출력하는 데 사용된다. printf로도 빈칸을 출력하는데 전혀 문제는 없지만 다른 장점들이 많다. \n을 입력하지 않아도 전 줄과 다음 줄을 구분하고 gets함수와 마찬가지로 함수 이름 안에 배열명만 적으면 되므로 간편하다.

5-2-3. 다차원 문자열 배열

문자열도 배열이므로 2차원 배열 형태로 나타낼 수 있다.



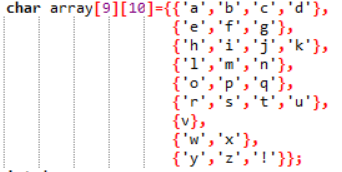
 어디선가 많이 봤던 상황이지 않은가? 그렇다. 이런 상황에서 이차원 배열을 이용하면, 더 편리하게 코딩을 작성할 수 있다.

위 그림에서 잠깐 이해가 안가는 부분이 있을 것이다. for문을 이용해서 array1을 출력하는 부분인데, 여기서 array[i]는 2차원 배열에서 i번째 행에 해당하는 문자열을 의미한다.

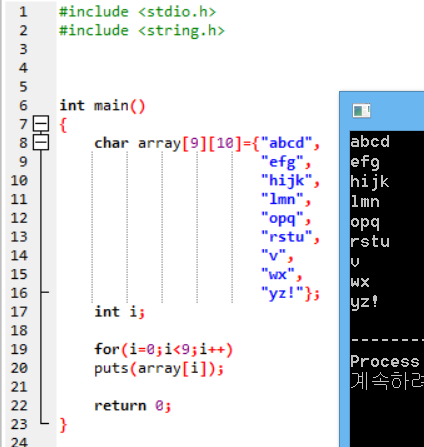
* 문자열 다차원 배열의 초기화

문자열 이차원배열도 초기화가 필요하다. 그 방법에 대해 알아보도록 하자.

먼저 문자 하나하나를 적어주는 방법이 있다.



다음으론 중괄호 안에 큰 따옴표(“”)를 이용하여 문자열을 한 행씩 안에 적는 방법이 있다.



문자열 이차원 배열도 다른 배열에서와 마찬가지로 첫 번째 크기를 생략할 수 있다.

