6 구조체

6-1. 구조체

6-1-1. 구조체의 개념

6-1-2. 구조체 정의하기

6-1-3. 구조체 변수의 선언과 사용

6-1-4. 구조체 변수의 초기화

6-1-5 구조체의 활용

6-1-5-1. 구조체를 멤버로 가지는 구조체

6-1-5-2. 구조체 배열

6-2. typedef

6-2-1. typedef 알아보기

6-2-2. typedef 의 쓰임

@더 알아보기: 공용체, 열거체

\*\*비유 알고 들어가기\*\*

변수 : (선물)상자,

변수형 : (선물)상자의 종류(ex. Int 형, char 형),

변수 값 : (선물)상자 안에 들어가는 선물

구조체 : (선물)상자를 담을 수 있는 큰(선물)상자,

Intro

앞서, 같은 종류의 상자들을 모아 놓는 배열에 대해 알아봤다. 그런데, 배열은 같은 종류의 변수만 모을 수 있다는 한계를 갖고 있다. 즉, 한 데이터 타입만 들어갈 수 있다는 것이다. 하지만, 실생활에서는 같은 것만 넣기보다 여러 종류를 넣어야 할 때가 많다. 지갑 안에만 봐도 지폐, 동전, 카드, 사진 등 다양한 종류의 것들이 들어간다.

(그림: 배열 vs 구조체 실생활-애가 물건 정리)

(책장이라는 배열 안에는 책만 넣을 수 있어!)

(지갑에는 돈, 카드 등 다양한 것들이 들어가지!)

(책장 :

<https://encrypted-tbn3.gstatic.com/images?q=tbn:ANd9GcSsFMJA453XkoKi7fqMqEo-_OIMQBEdlx7S2zqm1L3uDwaIQS8xAw>

지갑 : http://cfile230.uf.daum.net/image/114B80445018E9EA20AC5B)

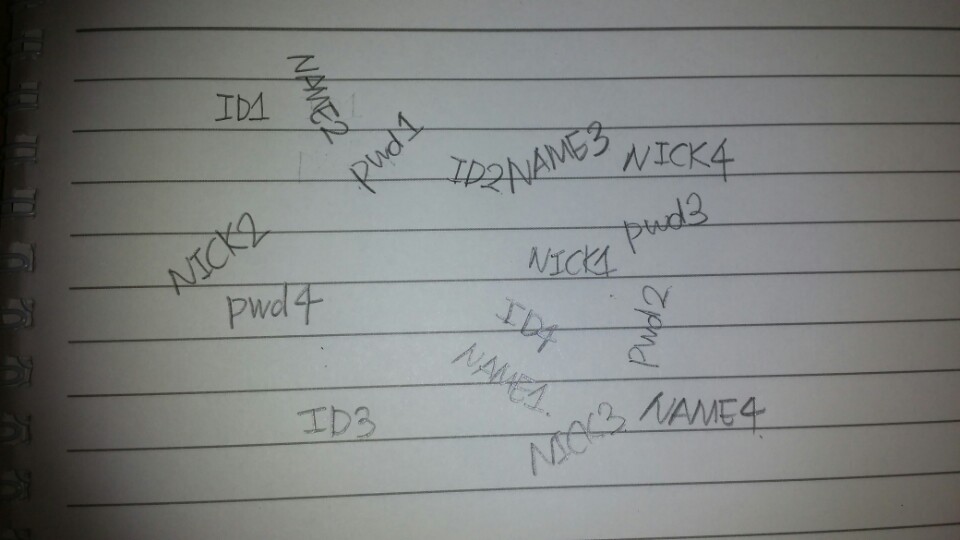
이번 단원에서는 다른 종류의 변수들도 넣을 수 있게 해주는 구조체에 대해 알아보자.

6-1. 구조체

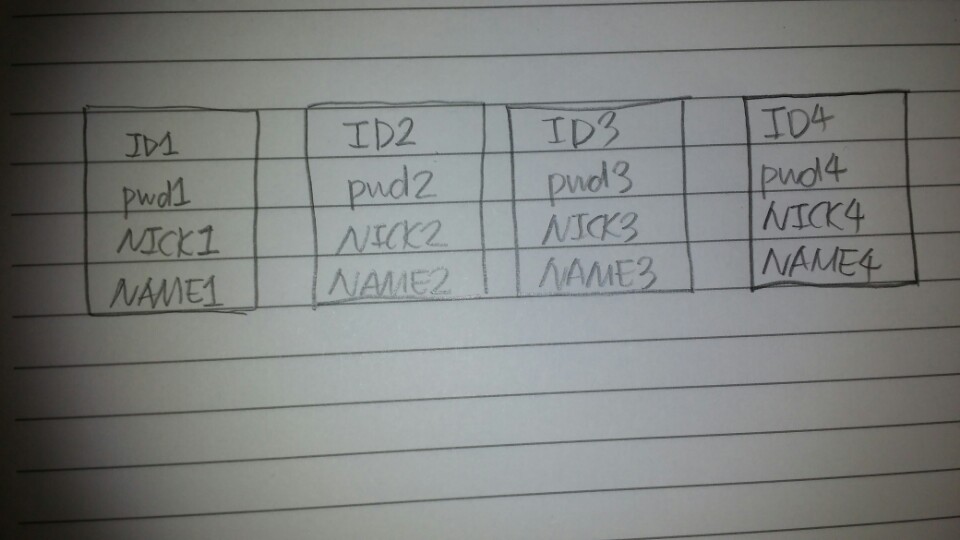
6-1-1. 구조체의 개념

구조체의 의미를 손 쉽게 알아 볼 수 있는 것이 바로 게임 계정일 것이다. 여러분이 만약에 게임 운영자가 되었다고 가정을 해보자. 수 백 아니 수 천의 사람들이 그 게임을 하기 위해 아이디와 비밀번호를 만들고, 이름과 닉네임, 주민 번호 등 수 많은 정보들을 기입해서 하나의 계정을 만든다. 만일 이 항목들을 따로 따로 관리한다면, 운영자는 사람이 많아질수록 유저(user)들을 관리하는 데에 어려움을 느낄 것이다. 이런 경우에 구조체를 사용하게 되면 관리가 훨씬 쉬워진다.

우선 구조체를 사용하지 않고 자료 입력하는 경우를 살펴보자.



위 그림에서는 어떤 데이터가 어떤 학생의 것인지 구분하기 어려워서 정보를 관리하는데 매우 불편하다. 그래서 우리는 아래의 그림처럼 구조체를 이용해 학생이라는 단위로 묶어 자료를 정리할 수 있다.



앞서 변수 단원에서 변수를 상자라고 했다. 그렇다면, 구조체는 다양한 종류의 변수들을 담는 커다란 ‘상자’라고 할 수 있다. 여기서 우리는 구조체 역시 변수라는 점을 알 수 있다. 구조체가 변수이기 때문에 주의할 점이 있는데 기본적인 것들을 다룬 후에 알아보도록 하자. int, float, char 등의 데이터 형들은 저마다의 바이트(byte) 크기를 갖고 있다. 그렇다면 그런 변수들이 모여있는 구조체의 크기는 어떻게 결정될까? 당연히 모인 변수들의 크기만큼이 구조체의 크기가 된다. 이와 같이 구조체는 사용자가 내부의 모습을 다양하게 정할 수 있기 때문에 ‘사용자 정의 자료형’이라고 한다.

(그림: )

한 칸 더 쌓을까?

(도시락1 : <http://image.gsshop.com/image/13/28/13283423_L1.jpg>

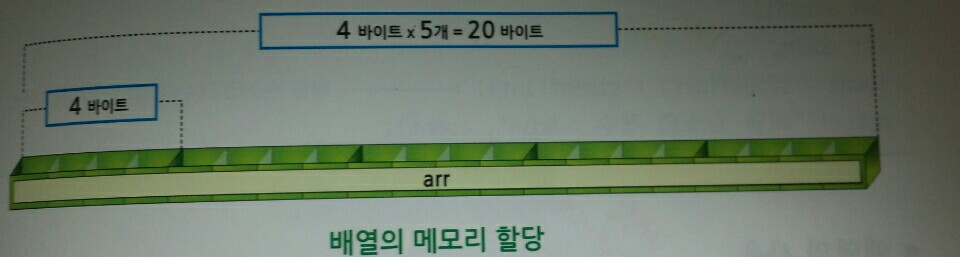
도시락2 : <http://kawaya.co.kr/web/product/medium/kawaya_5578.jpg>

사람 :

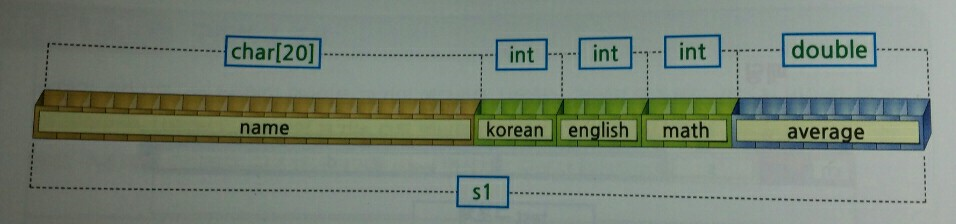
http://1.bp.blogspot.com/-rnhsBkS6uoA/UMg34TDoDgI/AAAAAAAABjM/m9aupICJ3dA/s1600/%EA%B6%81%EA%B8%88%ED%95%9C%ED%91%9C%EC%A0%95.png)

참고) (여기에 더 정의적인 설명이 있다. 이해가 되는 내용이라면 읽어보고, 만약 이해가 되지 않더라도 고민하지 말자. 중요한 부분을 확실하게 집고 넘어 갈 것이다.) C언어에서의 자료형은 기본 자료형과 파생 자료형으로 분류 할 수 있다. 기본 자료형은 char, int, double과 같은 기본적인 자료형이다. 파생 자료형은 이들 기본 자료형에서 파생된 것으로 배열, 포인터(다음 단원에서 배울 예정), 열거체, 구조체 등을 들 수 잇다. 구조체는 파생 자료형 중에서도 가장 일반적인 구조를 가진다. 구조체는 서로 다른 형태의 데이터를 결합시키는 아주 유용한 파생 자료형이다.

배열과 구조체는 여러 변수들을 하나로 묶는다는 공통점이 있지만, 배열은 여러 개의 같은 자료형을 하나로 묶은 것이고, 구조체는 여러가지의 자료형들을 하나로 묶은 것이라는 차이점이 있다. 당연히 메모리에서도 차이가 생긴다. 배열은 정해진 자료형의 개수만큼 메모리가 잡히고, 구조체는 사용자가 정의한 대로 메모리가 다양해진다.



vs



6-1-2. 구조체 정의하기

1-1에서도 언급했듯이, 구조체는 사용자 정의 자료형이다. 사용자가 임의로 구조체를 새로 하나 정의할 때 어떤 요소가 필요할지 정의 방법에 대해서 알아보도록 하자.

구조체의 기본적인 문법은 아래와 같다.

struct 태그(구조체의 이름 정의) {

자료형 멤버 1;

자료형 멤버 2;

…. };

예를 들어, int형 변수 두 개를 가지는 구조체를 만들어보면 아래와 같다.

struct point {

int x;

int y;

};

이렇게 point 라는 이름의 구조체 안에는 int형 변수 x 와 y 가 있는 셈이 된다. 한편, point라는 구조체는 int 라는 하나의 자료형만 사용했기 때문에 배열과 같다는 것도 알아두고 가자.

Tip!! 괄호를 닫고 ‘ ; ’ 써야함!!

이번에는 이전에 코드를 작성해봤던 학생 개개인의 데이터를 만들어 보는 구조체를 만들어 보자

구조체 태그명

struct student{

int number; // 학번--------------------------------------|

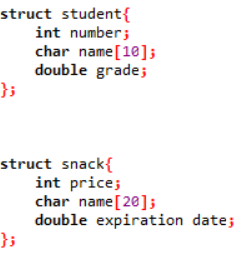
char name[10]; // 이름 |->구조체 멤버

double grade; // 학점-----------------------------------|

};

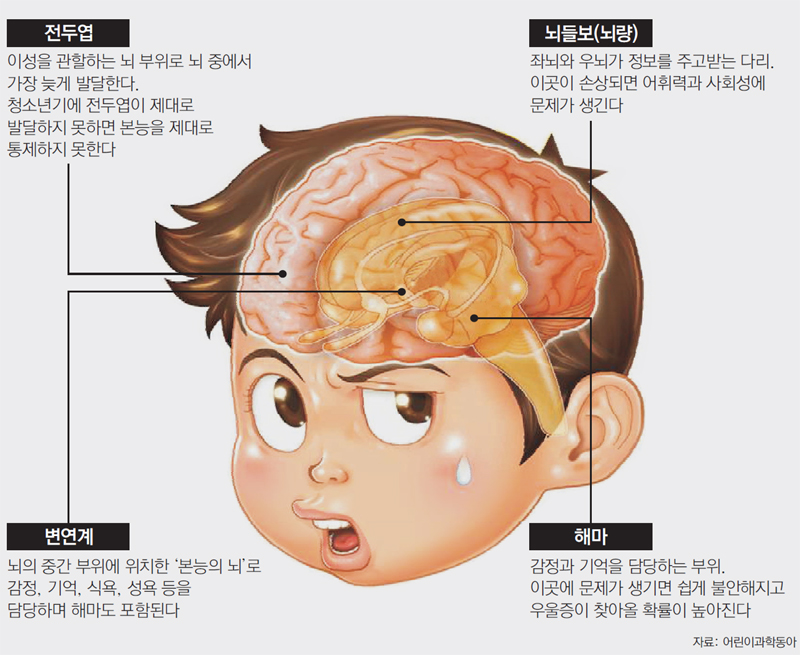
먼저 struct는 구조체를 정의할 때 사용하는 키워드이다. 이어서 나오는 student 를 구조체 태그(struct tag)라고 한다. 한마디로 구조체의 이름이다.

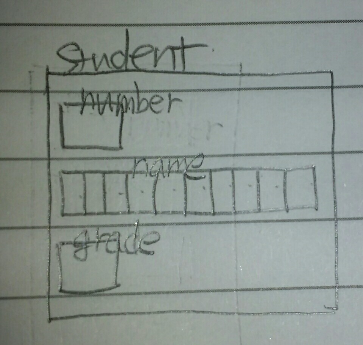
.



서로 다른 구조체 태그명을 정의 한다.

student 라는 구조체를 보면, 안에 number, name, grade 라는 구조체 멤버(struct member)들이 있다. number 안에는 int 형 변수가, name 에는 char 형 변수 10개가, grade 에는 double 형 변수가 들어갈 수 있다. 즉, student 라는 구조체는 학생의 학번과 이름, 성적을 넣을 수 있는 구조체로 학생 관리에 사용되는 구조체이다.

 여기서 주의 할 점은 구조체를 정의하는 것만으로는 구조체를 사용할 수 없다는 것이다. 구조체를 정의했다는 것은 int a; 와 같이 변수를 선언한 것이 아니라, int 와 같은 자료형을 만든 것이다. 사용자가 원하는 자료형을 구상해 놓은 것이다.



학번과 이름, 성적을 넣을 student 라는 자료형을 만들어볼까?

그렇다면 이제는 만들어 놓은 구조체를 사용해야 한다. 다음 절에서 구조체 변수를 어떻게 선언하고 사용하는지 알아보자.

6-1-3. 구조체 변수의 선언과 사용

구조체를 정의하는 것은 자료형을 만드는 것이라고 앞서 말했다. 자료형이란 ‘int a;’에서 int 를 말한다. 즉 구조체를 정의하는 것 자체로는 구조체를 사용할 수 없는 것이다.

struct student{

int number; // 학번

char name[10]; // 이름 구조체 정의

double grade; // 학점

};

int main(void){

struct student s1;// 구조체 변수 구조체 변수 선언

…|

}

독자들은 갑자기 머리가 복잡할 것이다. ‘뭐야 구조체를 정의하는 법을 기껏 배웠더니 구조체 변수라는 것을 또 배워야 되는 거야? 와 이거이거 되게 어렵네’ 라는 반응이 나올 수 있을 것이다. 차근차근 짚어보면 쉽게 이해될 내용이므로 이어지는 내용을 잘 따라오도록 하자.

구조체를 정의하는 것과 구조체 변수를 선언하는 것의 차이를 비유를 통해 알아보자. 내일 학교에서 소풍을 가게 되어서 도시락을 싸야 한다고 해보자. 먼저 도시락에 무엇을 넣을지 생각해야 하고, 그에 맞는 도시락 통을 생각해야 한다. 여기까지가 ‘구조체를 정의하는 것’이다. 내용물은 커녕 도시락 통 자체도 준비되어있지 않지만, 어떤 도시락을 쌀지 미리 구상해 놓는 것이다.

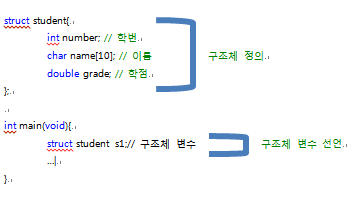
이 후에 도시락 통을 가져온다. 이 과정이 ‘구조체 변수를 선언하는 것’이다. 변수를 선언했을 때 빈 상자(사실 쓰레기 값이 차 있지만)가 생기듯이, 구조체 변수를 선언을 하면 도시락 통을 하나 만드는 것이다. 남은 과정은 도시락 통에 내용물을 채워 넣는 일이다. 이 과정은 변수를 초기화 해주듯 구조체 멤버 변수들을 초기화 해주는 과정이다.

(칸이 보이는 도시락통 그림 추가)

구조체를 정의한다 -> 도시락 통에 무엇을 넣을지 정하는 것

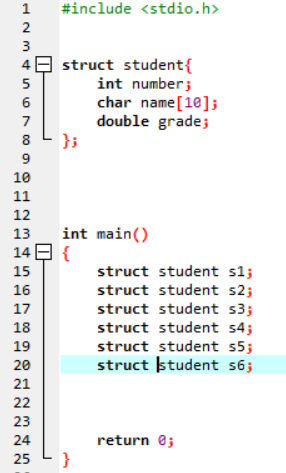
구조체 변수를 선언한다 -> 도시락 통을 만드는/가져오는 것

앞에서도 언급 했듯이 구조체를 정의하는 것과 구조체 변수를 선언하는 것은 다르다고 설명했다. 여기 아래에 구조체 변수를 선언하는 방법이 나온다.



위의 방법이 구조체 변수를 선언하는 방법이다. 독자들은 당연히 의문점이 많이 생겼을 것이라고 생각 한다. ‘s1은 어떻게 사용 되나요?’, 그럼 구조체 변수는 어떻게 사용하나요?’의 질문들을 예상할 수 있다. 계속해서 친절한 설명이 이어질 것이니 한 글자 한 글자 잘 따라와주길 바란다.

가장 먼저 student 라는 구조체 태그(구조체 이름)를 정의했다. 이것은 단순히 약속일 뿐이다. 다시 말해, number 와 name, grade 를 내용물로 하는 student 라는 도시락을 구상한 것이다(구조체 정의하기). 그 다음에는 구상한 도시락을 만든다. student 라는 종류의 도시락 s1을 만든 것이다(구조체 변수 선언). 그렇다면 s2도 만들 수 있을까? s3는? s4는? 몇 개까지 더 만들 수 있을까? 구조체는 변수이기 때문에 한번 student를 정의하면 컴퓨터 메모리를 다 쓰지 않는 한에서 얼마든지 구조체 변수를 만들어 낼 수 있다.



한 가지 주의할 점은 구조체 변수를 선언할 때 ‘student s1;’이라고만 선언하면 안된다는 것이다. 구조체의 경우 사용자가 구조체를 이용했다는 것을 표시하기 위해 앞에 struct라는 키워드를 꼭 붙여야 한다.

int a ;

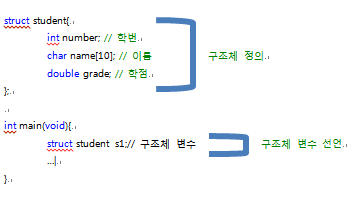
struct student s1 ;



**자료형**

**변수**

지금까지 구조체 변수 선언 방법에 대해서 알아보았다. 구조체 변수 선언은 다음과 같은 방법으로도 할 수 있다.



구조체를 정의한 후 main 함수에서 구조체 변수 선언을 할 수 있다

struct student{

int number; // 학번

char name[10]; // 이름

double grade; // 학점

}**s1**;

구조체 정의와 구조체 변수 선언을 함께 할 수 있다.

struct {

int number; // 학번

char name[10]; // 이름

double grade; // 학점

}**s1**;

구조체 정의 시에 태그명을 생략하고, 변수 선언을 할 수 있다.

---🡪 구조체의 메모리

구조체의 메모리 사용에 대해 간단히 짚고 가자.

struct student{

int number; -> 4bytes

char name[10]; -> 1\*10bytes

double grade; -> 8bytes

}s1;

먼저 student 라는 구조체 안에 number는 int 형 변수로 4bytes를 차지한다. 두 번째로 name[10]는 char 형 변수가 10개이므로 총 10byte의 메모리를 차지한다. 마지막으로 grade는 double 형이기 때문에 8bytes의 메모리를 차지한다. 즉, 총 22byte의 메모리를 차지할 수 있는 것이다.

6-1-4. 구조체 변수의 초기화

몇몇의 학생들은 이 소단원 명을 보고 뭔가 이상하다고 눈치 챘을 것이다. “ 그럼 구조체를 초기화 하는 법은 없어요? “ 라고 질문을 할 것이라고 예상한다. 답은.. 구조체 자체를 초기화 하는 방법은 없다. 거듭 말하지만, 구조체는 사용자 정의 자료형이기 때문에 정의를 해야 하는 것이고, 구조체 변수를 선언한 뒤에 그 변수를 초기화를 하는 것이다.

구조체 변수는 선언과 초기화를 하는 거야.

구조체는 정의를 하는 거고.



구조체 변수를 초기화하는 방법을 두 가지로 볼 수 있다. 한 가지는 구조체 변수를 통해 초기화 하는 방법이고, 다른 한 가지는 구조체 멤버를 참조하는 것이다. 설명이 어려울 수 있지만 예시를 읽어보면 이해가 될 것이다. 구조체 변수를 통해 초기화 하는 방법은 다음과 같다.

struct student{

int number; // 학번

char name[10]; // 이름

double grade; // 학점

};

struct student s1 = {24,”KIM”, 4.3};

구조체 변수 s1을 선언함과 동시에 각 멤버 변수들을 위와 같이 초기화할 수 있다. 대괄호 안의 멤버 변수들을 순서대로 초기화 해주면 된다.

**구조체 멤버 참조**

구조체 변수에서 따로 변수를 초기화 해주려면 접근 연산자인 ‘.(마침표)’를 사용해주어야 한다. 아래 그림의 “s1.number”, “s1.name”, “s1.grade” 처럼 말이다.

struct student{

int number; // 학번

char name[10]; // 이름

double grade; // 학점

}s1;

int main(void){

s1.number = 2014310496;

s1.name = "배성현";

s1.grade = 4.0;

}

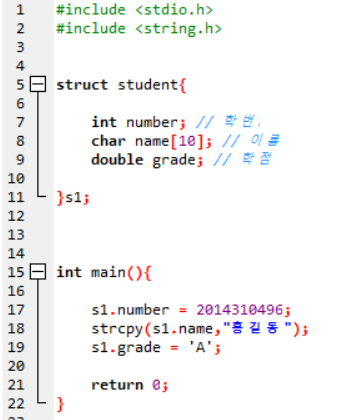
“접근 연산자는 또 무엇인가.ㅠㅠ”라며 당황할 수 있다. “s1.number”, “s1.name”, “s1.grade” 에서 ‘.(마침표)’는 구조체 s1의 멤버 변수인 number, name, grade에 접근 시켜주는 연산자이다.

저번 단원에서 배열을 확실하게 공부한 학생이라면 지금 위의 코드가(그림번호필요) 다소 이상하다고 느낄 것이다. 맞다. 위의 코드는 틀린 코드이다. ‘name은 길이가 10인 배열인데 주소로 참조(접근) 해야 되지 않나요?’라고 생각한 독자들은 내가 정말 공부를 제대로 하고 있구나 라고 생각해도 된다. 그렇지 않은 독자라면, 이 책을 우선 1독을 한 후 다시 공부하기를 추천한다. 만약에 구조체 정의에 ‘char name;’ 라고 정의가 되어 있고, 초기화를 “s1.name=’a’;” 라고 한다면 전혀 문제가 되지 않지만, char name[10]인 경우엔 문제가 된다.

독자들은 앞서 배열 단원을 공부할 때 배열의 이름이 곧 배열의 주소라는 것을 배웠을 것이다.

name = &name[0]

배열의 주소는 배열의 처음 값의 주소 name[0]의 주소이므로 따라서 s1.name는 s1.name[0], s1.name[1]와 같이 한 글자씩만 초기화해야 한다. 이런 모순의 상황일 때 쓰는 함수가 있는데 바로 strcpy(배열의 복사)이다. 결론적으로 다음 그림과 같이 된다.



strcpy 함수는 s1.name에 “홍 길 동”이라는 문자열을 대입하겠다는 의미이다. 위의 내용 전부를 바탕으로 코딩을 작성해보자

**예제1 : 학생 인적 사항을 기록.**

#include <stdio.h>

#include <stdlib.h>

struct student{

int number;

char name[10];

double grade;

};

int main(void)

{

struct student s1;

printf(“ 학번을 입력하시오 : ”);

scanf(“%d”, &s1.number);

printf(“ 이름을 입력하시오 : ”);

scanf(“%s”, &s1.name);

printf( 학점을 입력하시오 : ”);

scanf(“%1f”, &s1.grade);

printf(“학번 : %d\n”, s.number);

printf(“이름 : %s\n”, s1.name);

printf(“학점 : %1f\n”, s1.grade);

return 0;

}

**예제2 : 두 좌표 사이의 거리를 구함.**

#include <stdio.h>

struct point {

int x;

int y;

};

int main(void)

{

struct point p1, p2;

int xdiff, ydiff;

double dist;

printf(“ 점이 좌표를 입력하시오 (x y):”);

scanf(“ %d %d, &p1.x, &p1.y);

printf(“ 점이 좌표를 입력하시오 (x y):”);

scanf(“ %d %d, &p2.x, &p2.y);

xdiff = p1.x-p2.x;

ydiff = p1.y-p2.y;

dist = sqrt(xdiff \* xdiff + ydiff \*ydiff);

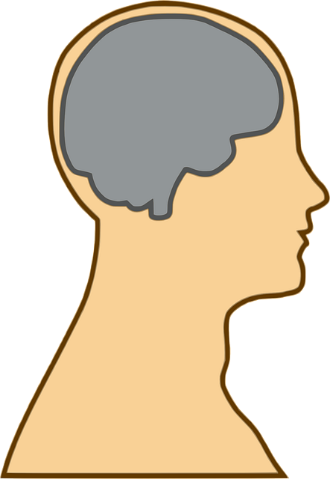
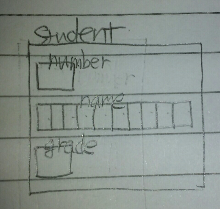
printf(거리는 %f 입니다. \n”, dist);

return 0;

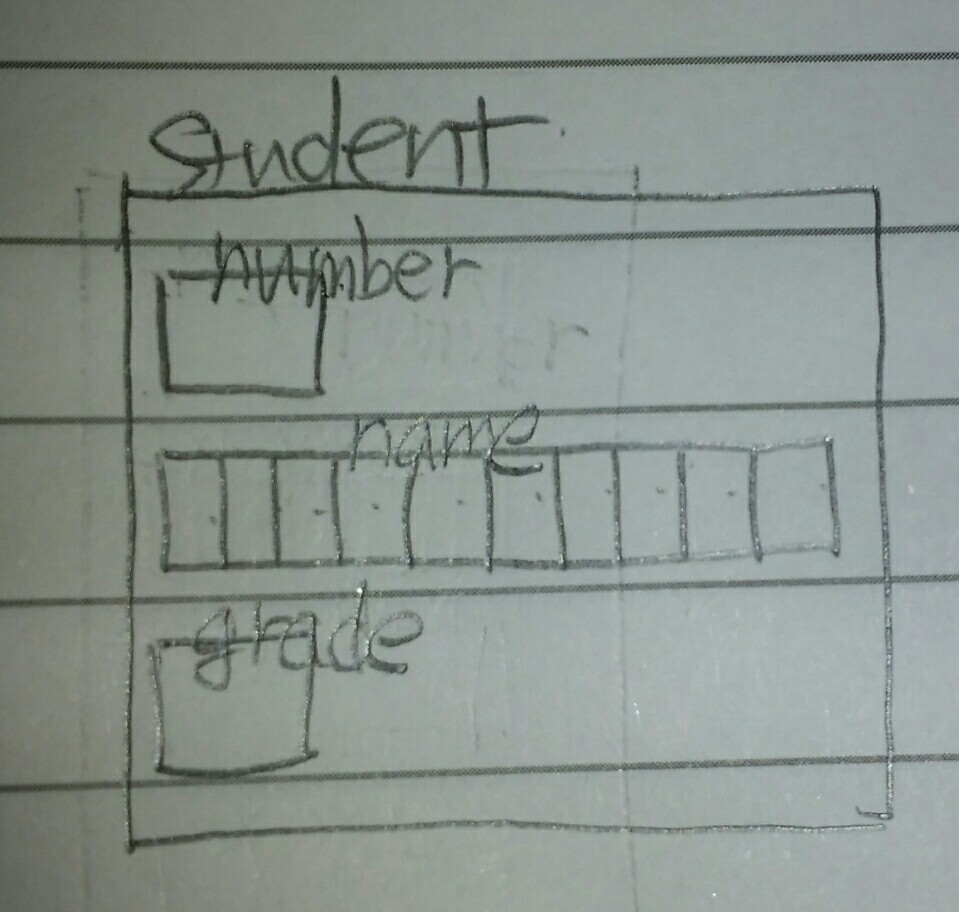
}

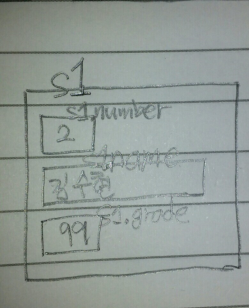
마지막으로 구조체의 정의, 구조체 변수의 선언, 구조체 변수의 초기화를 정리해보자.

구조체를 정의한다 -> 어떤 도시락을 쌀지 생각하는 것이다.

-> 도시락 그림으로

구조체 변수를 선언한다 -> 도시락을 쌀 도시락 통을 준비하는 것이다.

-> 도시락 그림으로

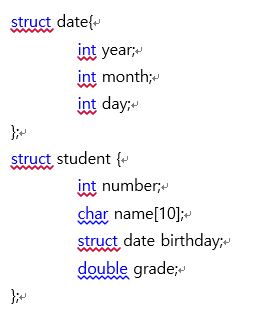
구조체 변수를 초기화한다 -> 도시락 통 안에 여러가지 음식을 넣는 것이다.

-> 도시락 그림으로

6-1-5 구조체의 활용

6-1-5-1. 구조체를 멤버로 가지는 구조체

모든 자료형은 구조체의 멤버가 될 수 있다. 즉, 구조체 역시 다른 구조체의 멤버가 될 수 있다. 구조체의 정의를 이해했다면 바로 이해가 될 것이다.



student 구조체 정의

date 구조체 정의

구조체 변수 선언

date 라는 구조체 안에는 년도, 월, 날짜가 각각 들어가고, student 구조체 안에는 학생의 개인정보인 학번, 이름, 생일, 성적이 들어간다. 여기서 birthday 는 생년월일을 나타내는 구조체 변수이다. 즉, student 구조체 안에 또 다른 구조체 변수가 들어간 것이다.

main 함수 안에서 struct student s1; 라고 구조체 변수를 선언했다고 하면, 이 때 s1 변수의 초기화는 아래와 같이 해주면 된다.

s1.number = 2016314355

s1.grade = 89;

s1.birthday.year=1997;

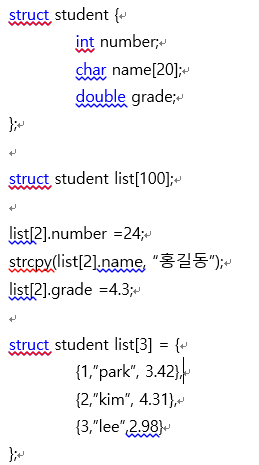
s1.birthday.month=8;

s1.birthday.day=2;

birthday 가 구조체 변수이므로 birthday 안으로 한 번 더 들어갈 수 있다. 이 때도 접근 연산자 .(마침표) 를 이용해준다.

6-1-5-2. 구조체 배열

struct student s1; 으로 구조체 변수 s1을 선언하면 학생 한 명에 대해서만 정보를 기입할 수 있다. 하지만, 학생관리 대상이 되는 학생은 한 명이 아니라 여러 명이다. 따라서 여러 개의 구조체 변수가 필요하게 된다. 이런 경우에 구조체 배열을 사용하게 된다.



크기가 100인 구조체 배열의 선언

멤버 변수들을 위와 같이 하나씩 초기화할 수도 있지만, 중괄호를 이용하여 한꺼번에 초기화할 수도 있다.

구조체 배열의 3번째 배열 요소를 초기화 해준다.

name 의 경우 strcpy 함수를 사용하여 초기화 한다.

코드에서 student list 가 크기만 다르게 두 번 선언되어 있는거 수정 부탁 list말고 다른 걸로!!

위에서 struct student list[100]; 라고 구조체 배열을 선언해주었다. 이 말은 struct student 라는 자료형 변수를 100개 선언했다는 것이다. 결국엔 배열과 같은 방식으로 사용된 것이다.

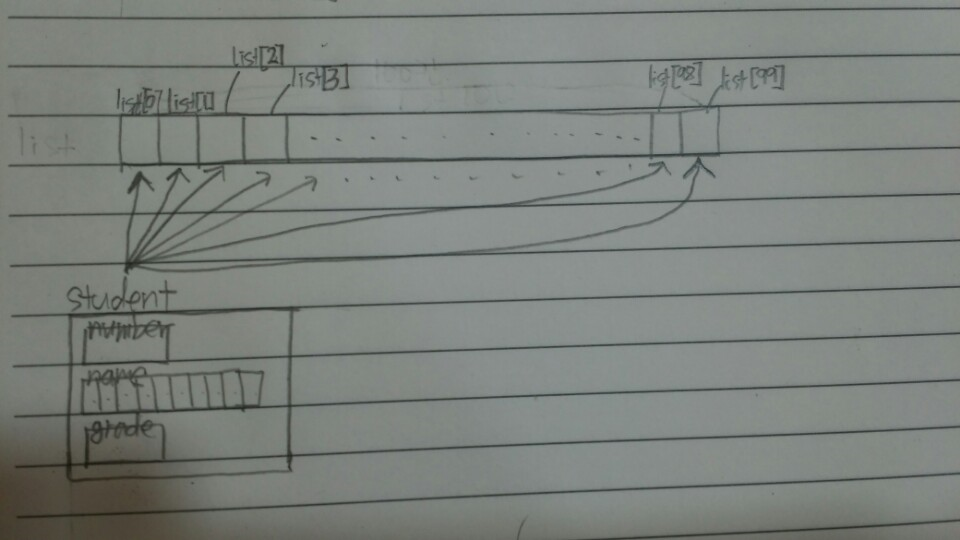
int a[30] ;

struct student list[30] ;



**자료형**

**배열**



6-2. typedef

6-2-1. typedef 알아보기

typedef 라는 뜻은 자료형 (type) + 정의 (define) 을 합친 말로, 자료형을 재정의 한다는 이야기이다. 좀 더 쉽게 설명하자면, 원래 입고 있던 옷을 리폼해서 입는 것이다.

간단한 예를 들어보면서 어떻게 쓰는지 알아보도록 하자.

typedef char CHAR;

typedef int INT32;

typedef short INT16;

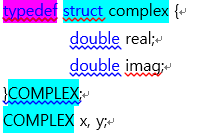
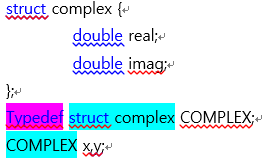
CHAR ch; //char ch; 와 같다.

INT32 i; // int i; 와 같다.

INT16 k; // short k; 와 같다.

typedef 라는 키워드를 이용해서 char 키워드를 CHAR로 바꾸고, int 키워드를 INT32로, short를 INT16으로 바꾼 것이다. 따라서 앞으로 문자형 변수는 “CHAR A;”, 정수형 변수는 “INT32 num;”와 같이 선언할 수 있다.

구조체 변수 또한 변수이기 때문에 typedef를 이용할 수 있다. 구조체는 typedef가 가장 유용하게 쓰이는 변수들라고 할 수 있다. 다음 그림은 구조체 변수의 이름을 typedef를 이용하여 바꾸는 코드이다.

 또는 

기존의 자료형인 struct complex를 COMPLEX라는 새로운 이름으로 다시 정의 했다. 따라서 “ struct complex x, y; ”와 “ COMPLEX x, y; ”가 모두 구조체 변수 x, y를 선언하는 방법이다.

6-2-2. typedef 의 쓰임

그런데 왜 typedef 를 사용하는 것일까? typedef 는 크게 두가지 장점을 지닌다. 한가지는 이식성이 좋다는 것이고, 다른 하나는 가독성이 높다는 것이다.

**ㄱ. 이식성** (빨리 쓸 수 있게 해준다)

코딩은 정확하고 빠르게 작성하는 것이 중요하다. 예를 들어,

typedef unsigned int size\_t;

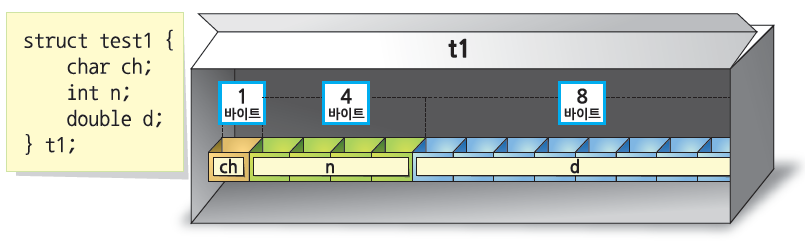
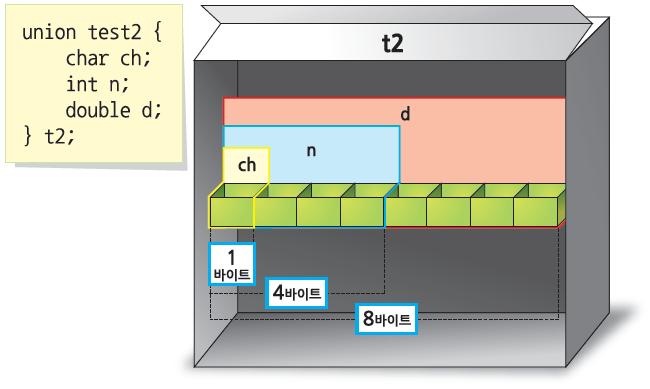
라는 코드가 있다고 가정해보자. 해석하면, unsigned int 대신에 size\_t 를 쓰겠다는 말이다. “unsigned int” 를 “size\_t”로 입력해도 같은 의미를 가지므로 신속하고 정확하게 코딩을 작성할 수 있다.

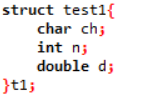
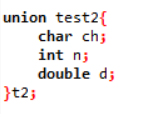
**ㄴ. 가독성**

사람에 따라 코딩하는 방법이 다르기 때문에 내가 짠 코드가 아닌 다른 사람의 코드를 보면 이해가 잘 안 될 때가 많다. 이 때, typedef 를 이용해 자료형의 이름을 이해하기 쉽게 바꾸면, 누구나 쉽게 코드를 읽을 수 있게 된다.

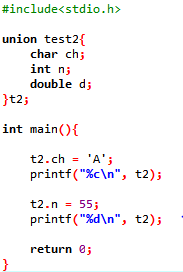
@더 알아보기: 공용체, 열거체

I. 공용체(union)

구조체와 비슷한 것으로 공용체라는 것이 있다. 공용체는 문법도, 변수 선언도, 초기화까지 구조체와 동일하다. 차이점은, 구조체는 멤버 변수들이 변수마다 메모리를 독립적으로 사용하고 있는 반면, 공용체는 멤버 변수끼리 메모리를 공유하고 있다는 것이다.



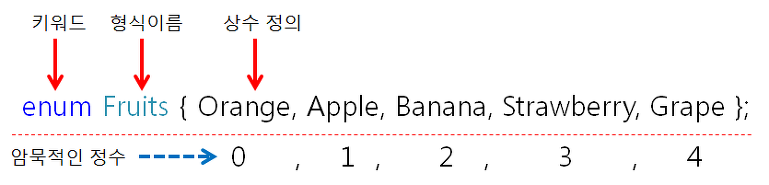
한 메모리 영역을 여러 개의 변수가 공유하는 이유는 크게 두 가지이다. 한 가지는 메모리를 절약할 수 있다는 점이고, 다른 한가지는 한 공간을 여러 자료형으로 사용할 수 있다는 점이다.

 위 그림과 같이 같은 멤버 변수들을 선언했더라도 구조체의 경우는 총 13 byte 를 사용하고, 공용체의 경우는 8 byte 만 사용한다. 공용체의 메모리 크기가 8 byte 인 이유는 메모리를 공유하는 멤버 변수들 중 가장 큰 메모리를 차지하는 것이 double 이기 때문이다. 유의할 점은, 공용체는 멤버 변수들이 메모리을 공유하기 때문에 한 순간에 하나의 멤버만 존재한다. 아래 코드를 보면, 먼저 공용체 변수 t2 안에 char 형 변수를 A로 초기화 했다. 그 후에 공용체 변수 t2 를 출력하면, A가 출력이 된다. 다음으로 int 형 변수 n 을 55로 초기화 하고 t2 를 출력해주면, A는 출력되지 않고 55만 출력이 된다. 공용체 안에는 하나의 변수만 존재할 수 있다는 것이다.

반면, 구조체는 한 순간에 모든 멤버가 존재한다. 즉, 공용체는 모든 멤버가 동시에 사용될 필요가 없을 때 사용되고, 구조체는 모든 멤버가 동시에 사용되어야 할 때 사용된다.

II. 열거체(enum)

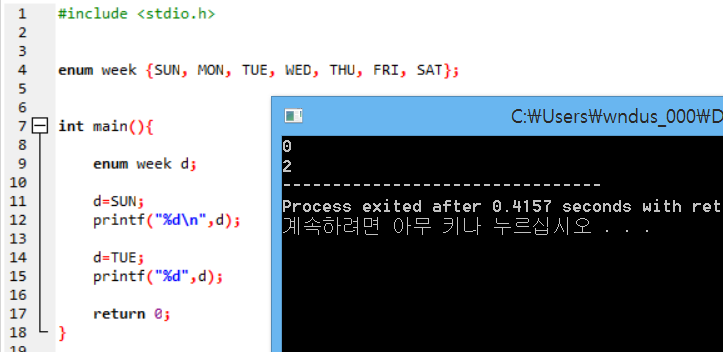
열거체란 비슷한 성질을 갖고 있는 것들을 모아놓은 것이다. 일상 생활에서는 월화수목금토일의 요일이나, 아래 예시와 같이 과일도 열거체가 될 수 있다.

[](http://www.google.co.kr/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=0CAcQjRxqFQoTCN6hweyirccCFYgYlAodWc4NEA&url=http://blog.daum.net/coolprogramming/114&ei=j1XQVZ6eLIix0ATZnLeAAQ&bvm=bv.99804247,d.dGo&psig=AFQjCNG355ipbd0xVfqcqt-dCnwsO0M3IA&ust=1439803128388658)

태그명

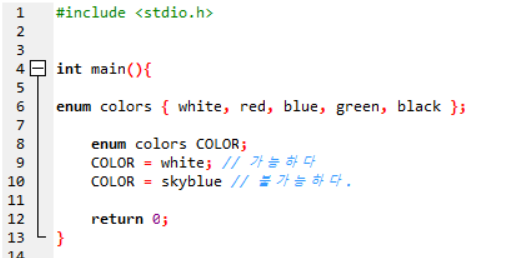
열거체는 enum 키워드, 태그명, 열거 상수들로 이루어져있다. 열거 상수는 정수형 상수이다. 열거상수에 해당하는 정수값을 열거체를 정의할 때 정하지 않으면 ‘(맨 앞에 있는 열거상수값) = 0 ‘으로 시작해서 1씩 큰 값을 갖는다

열거체 역시 변수를 선언해주어야 사용할 수 있다. 열거체 변수는 정의된 열거상수 중 하나만을 가질 수 있다. 요일을 예로 들어보자. 요일을 나타내는 변수 d를 선언했다고 가정하자. 이 때, 변수 d는 SUN, MON, TUE, WED, THU, FRI, SAT 중의 하나만을 가질 수 있다.



열거체를 정의할 때 열거상수에 특별한 값을 부여하지 않았으므로 SUN부터 차례대로 0, 1, … 7 의 값을 갖는다. TUE는 3번째 값이므로 2가 출력된다.

색깔을 또다른 예를 들어보자.



white 에는 0, skyblue 라는 값은 선언되어있지 않다. 따라서 COLOR 에 white 를 넣는 것은 가능하고, 밑줄은 당연히 불가능하다.



SUN=1, MON=2, TUE=3…SAT=7로정의가 된다.



이와 같이 지정을 하게 되면 SUN=7의 값이 나오고 다시 MON 부터 1로 시작해서 SAT에 6끝날 수 있도록 1씩 증가하게 된다.

열거체는 switch문에서 쓰이기도 하고, c콘솔창에서 글씨에 다양한 색을 입혀 출력하는데 사용하는 등 다양하게 이용될 수 있다.