1. 설치

2018년 10월 20일 토요일 오후 9:35

c 자료구조 !
공부순서 :
1. C(하드웨어,컴파일러,운영체제)
2. 자료구조(모든곳에있는요소)
-> 자료구조종류 : list, stack, queue, tree, graph
3. C++
-> 게임(구로디지털단지)
, opencv(카메라,자율주행) --> 포인터를 알아야함! -> c언어를 공부해
, qt5(자동차계기판ui)
4. STL(공동환강사님)

1. 절차지향 => 2.객체지향(c++ 을 간략화-> java -> c#) => 3.일반화(STL)

www.visualstudio.com
커뮤니티 2017 실행

C++를 사용한 데스크톱 개발 -> 설치

비쥬얼 스튜디오를 쓰는 이유: -> 디버깅이 편함
우리는 .c를 짜면 컴파일,링킹은 편집기가 해줌
[.c] --complie--> [.o] --linking --> [.exe]
□ ---> □virtual memory (window:4기가를 잡고 유저에게 2기가를 줌,linux : 4기가잡고 3기가를 줌) [memory□] <---cpu가 메모리를 읽음
↑이걸 천천히 보는걸 debugging이라함!

2. 컴파일, 링킹, 실행

2018년 10월 20일 토요일 오후 9:36

똑같은 소스코드를 각각 다른 컴파일러로 보내
c -> window 에서는 [Visual Studio 2017]이 .obj .exe로 컴파일
c -> linux 에서는 [gcc]가 .o .exe로 컴파일

tcp/ip(책 : 윤성우) ->system-programming(책:에이콘출판사 -linux api) 공부하기 !

java -> clss파일을 통해 어떤 플랫폼에서 돌려도 다 돌아가지만 c는 컴파일 시켜야함

파일 - > 새 프로젝트
좌측 Visual C++ 선택, 기본값 : 빈프로젝트, 위치 : D:₩C₩, 이름: Day1

우측 : 좌측 : 추가 : 새항목 --> C++선택, zoo.c 확장자는 반드시 .c로 끝나야해!

3. 2차원배열

2018년 10월 20일 토요일 오후 9:36

①int ②a = 7 ①int ③a[②3]={3,4,5}
□<-7이들어감 □□□<-345
a = 7 a[0] = 3, a[1]=4, a[2]=5

<<<< 2차원배열 읽는 순서 >>>>

①int ④a [③2][②3] 나는 int형 3개짜리가 총 2개필요해

a:2차원배열명 // a[0],a[1]->2차원배열 안에 있는, 1차원배열명
int a [2][3] = {1,2,3,4,5,6};
a[0] : □□□ <- 1,2,3
a[1] : □□□ <- 4,5,6

a[0][1] = 1, a[0][2] = 2, a[0][3] = 3
a[1][1] = 4, a[1][2] = 5, a[1][3] = 6

& : 주소연산자
ex) &a 붙이면 a의 주소(위치값)가 나옴 -> 배열명은 선두주소다!

a[0]의 주소(번지)에 접근하는 것이 중요함
1. &a -> 전체를 감싼 주소
[□□□
□□□]

2. a -> 2차원배열 값 선두요소(int형3개짜리의 첫번째) -> 층을 옮김 (행)
[□□□]
□□□

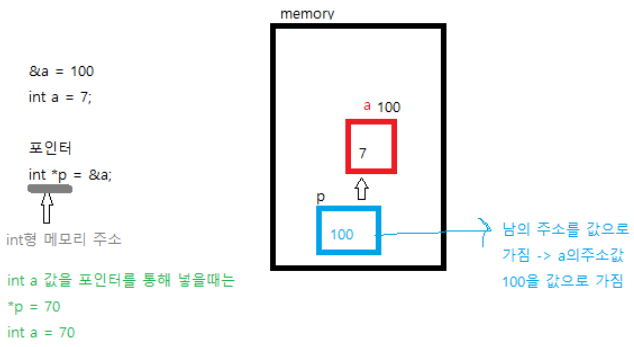
3. &a[0] ->2번과 완전히 똑같은 코드, 1차원배열을 통째로 감싸버릴거야!
[□□□]
□□□

4. a[0] -> 호실을 옮기는 표현(열)
[□]□□
□□□

5. &a[0][0] ->4번과 완전히 똑같은 코드임!
[□]□□
□□□

4. 포인터

2018년 10월 20일 토요일 오후 9:36

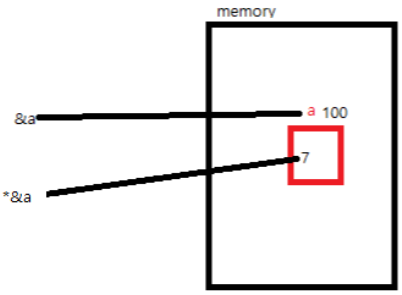


모든 메모리에는 주소가 있고
메모리에 접근하려면
접근하려는 곳의 메모리의 주소를 갖고있어야함

ex)
int a = 7;
주소:100 : □<-7

---------- [* : 간접연산자] ----------
: 항상 뒤에 주소가 나옴

pf("%d", a); ->direct Acc => 7출력
pf("%d", &a); -> a네집주소 address => 100출력
pf("%d", *&a); -> a네집주소 100 안쪽의 값이 뭐냐! => 7출력
간접 : indirect Operater



1. Day2

& 주소
* 간접
--> 그림

<call by value>

```
void func(int *a)  //a와 b의 내용(값)을 바꾸는 함수
{
	int *a = 70;
}
int main()
{
	int a = 7;
	func(&a);	//  == func()  -> call by value: 값이 copy됐다.
	printf("%d₩n", a);

	return 0;
}
```

<포인터>

```
int main(void)
{
	int a = 7;
	int *p = &a; //a의 주소값을 가짐

	printf("%d₩n", a);  //a의 데이터 7
	printf("%d₩n", &a); //a의 주소  54123
	printf("%d₩n", p);  //a의 주소  54123
	printf("%d₩n", *p); //a의 데이터 7
	printf("%d₩n", *&a);  //a의 주소 7

	getchar(); //콘솔 화면이 사라지는 것을 멈추게 함
	return 0; //os에게 '프로그램이 잘 끝났다'라고 알림
}
```

<a와 b값 바꾸기>

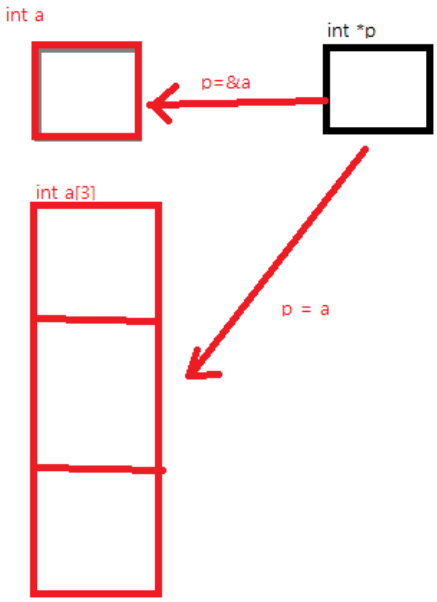
```
void func(int *a, int *b)   //a와 b의 내용(값)을 바꾸는 함수
{
	int t;
	t = *a;
	*a = *b;
	*b = t;
}
int main()
{
	int a = 7;
	int b = 5;
	//func(&a);		//  == func()  -> call by value: 값이 copy됐다.
	func(&a, &b);   //a와 b의 내용(값)을 바꾸는 함수
	printf("%d %d₩n", a, b);  //5  7

	return 0;
}
```

```
int main()
{
	int a = 7;
	int b = 5;
	int *pa = &a;
	int *pb = &b;
	// pa와 pb를 이용해 a,b를 바꿔라
	int t;
	t = *pa;
	*pa = *pb;
	*pb = t;

	printf("%d %d₩n", a, b);  //5  7

	return 0;
}
```

---

5. 싱글포인터

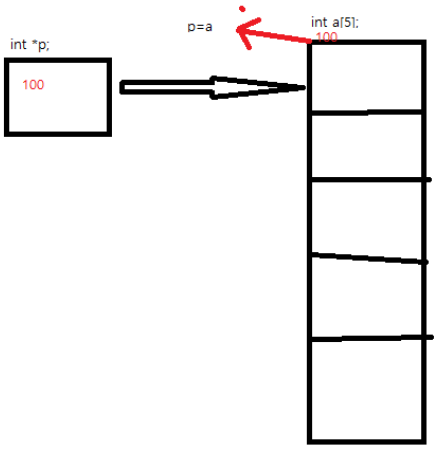
2018년 10월 20일 토요일 오후 9:37



p = &a;니까
scanf("%d", &a); -> 집어넣은 데이터를, &a자리에 p넣을수있음
print("%d", a); -> 화면에 출력 a자리에 *p넣을수있음

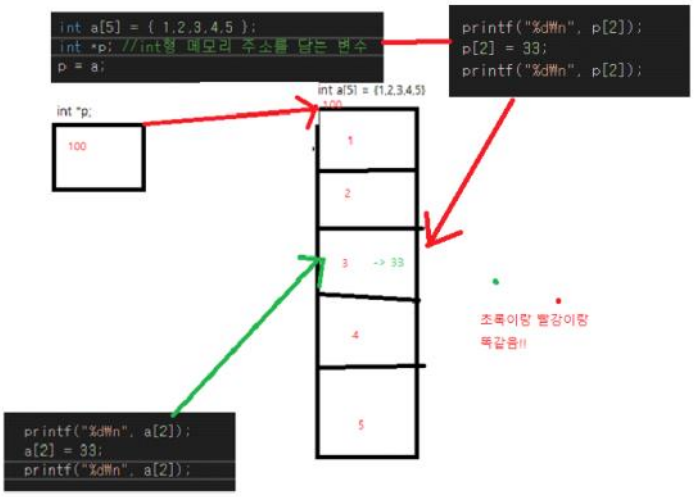
```
int main()
{
	int a[5] = { 1,2,3,4,5 };

	printf("%d₩n", a[2]);
	a[2] = 33;
	printf("%d₩n", a[2]);
}
```

>>>>>>> 위와 아래 같음 <<<<<<<<<

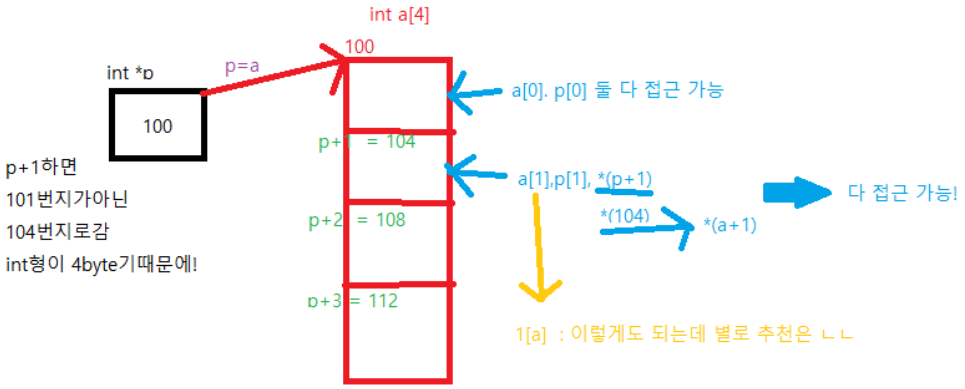



```
int main()
{
	int a[5] = { 1,2,3,4,5 };
	int *p; //int형 메모리 주소를 담는 변수
	p = a;	//&a안하는이유는 a(배열의이름)은 선두주소값이기에!

	printf("%d₩n", p[2]);
	p[2] = 33;
	printf("%d₩n", p[2]);
}
```

>>>>>>> 위와 아래 같음 <<<<<<<<<

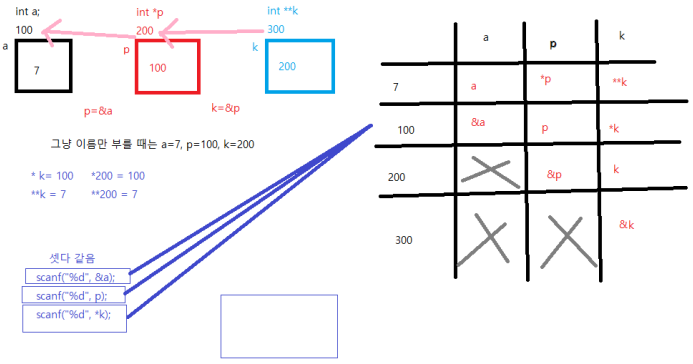


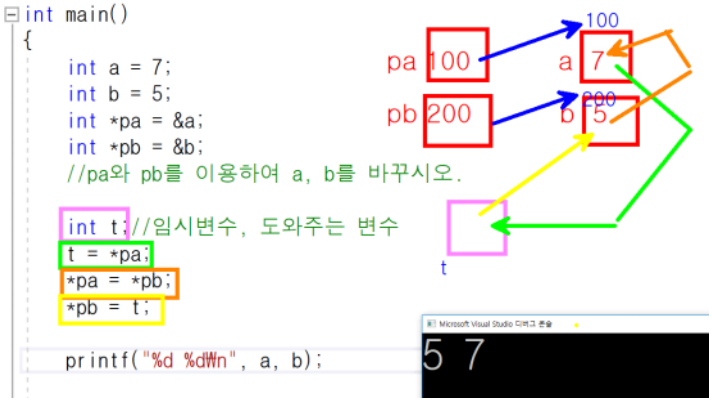

```
func(int *p)
{
	printf("%d₩n", p[2]);
	p[2] = 33;
	printf("%d₩n", p[2]);
}
int main()
{
	int a[5] = { 1,2,3,4,5 };
	func(a);
}  -> main 함수의 코드가 짧아짐
```


int a[4]
100
int *p
p=a
100
a[0]. p[0] 둘 다 접근 가능
p+1 = 104
a[1],p[1], *(p+1)
*(104)
*(a+1)
다 접근 가능!
p+2 = 108
p+3 = 112
p+1하면
101번지가아닌
104번지로감
int형이 4byte기때문에!
1[a] : 이렇게도 되는데 별로 추천은 ㄴㄴ

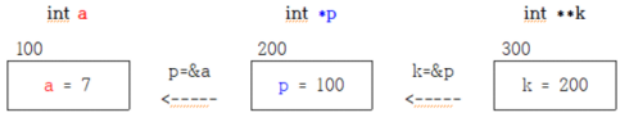
6. 더블포인터

2018년 10월 20일 토요일 오후 9:38





● 그림



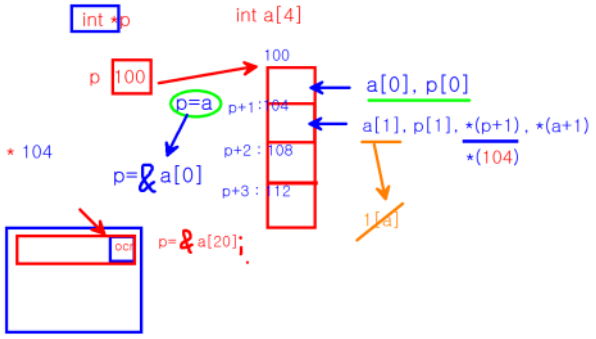
● 표

	a (사람)	p (택배)	k (택배사)
7	a	*p	**k
100	&a	p	*k
200		&p	k
300			&k

**k = **200

k = *100 = *&a

scanf = ("%d", &a); = scanf = ("%d", p); = scanf = ("%d", *k);



7. 포인터 배열

2018년 10월 20일 토요일 오후 9:40

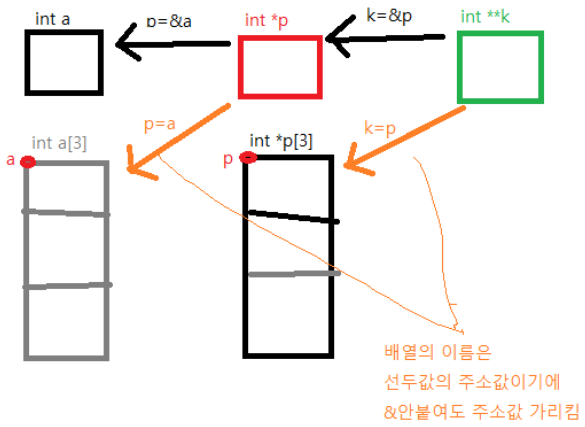
1. 포인터 여러개 만들기

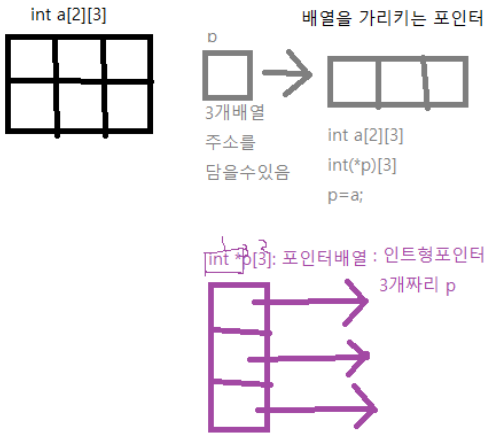
포인터 배열은 더블포인터가 봄
int *p[3]이런식 : int형 포인터가 3개있다

int(*p)[3] : 배열 포인터 인트형메모리가 3개있는데, 그 배열을 가리키는 포인터
- 인트형 3개짜리 포인터를 가리키는 포인터 p

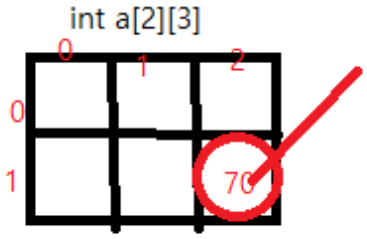
int *p[3] : 포인터배열
인트형 포인터가 세개있구나! 그 배열 이름이 p구나

1-1 포인터 배열





1-2 2차원배열 표현

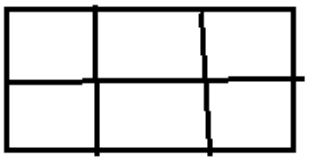


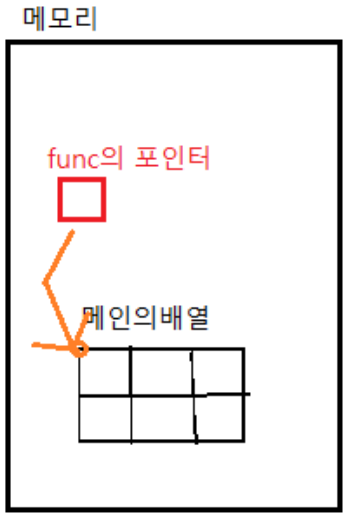
a[1][2] / a[1]=T라고하면
T [2]
*(T+2)
*(a[1]+2)
((a+1)+2)
(*(a+1))[2]
: 위의 식 모두 a[1][2]
가리킬 수 있음 =70

1-3 포인터사용이유

1-3 포인터사용이유

메인함수가 길때,
func()으로 빼면,
func의 배열포인터가
메인의 배열에 접근할수있음



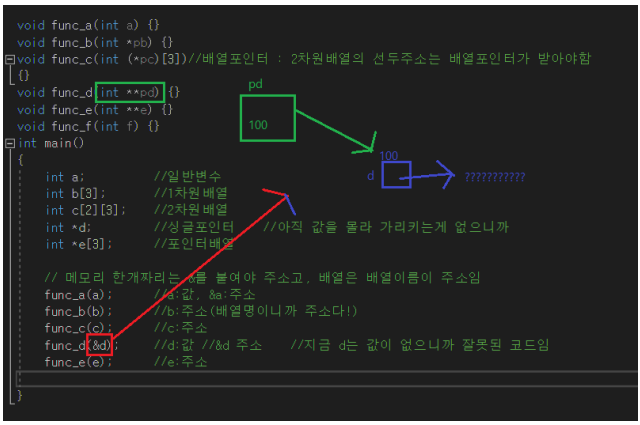


8. 포인터 func으로 값 받기

2018년 10월 20일 토요일 오후 9:41

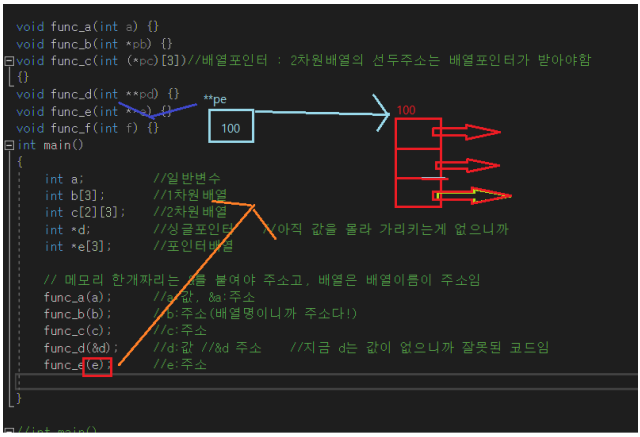
2. 포인터 func으로 값 받기
코드

```
void func_a(int a) {}
void func_b(int *pb) {}
void func_c(int (*pc)[3])//배열포인터 : 2차원배열의 선두주소는 배열포인터가 받아야함
{}
void func_d(int *d) {}
void func_e(int **e) {}
void func_f(int f) {}
int main()
{
	int *d;			//싱글포인터	//아직 값을 몰라 가리키는게 없으니까
	func_d(d);		//d:값 //&d 주소	//지금 d는 값이 없으니까 잘못된 코드임
}
```



그래서 func_d(&d) 포인터 값을 보내야함

그러니까
func_d(int **pd) 이렇게 받아야함



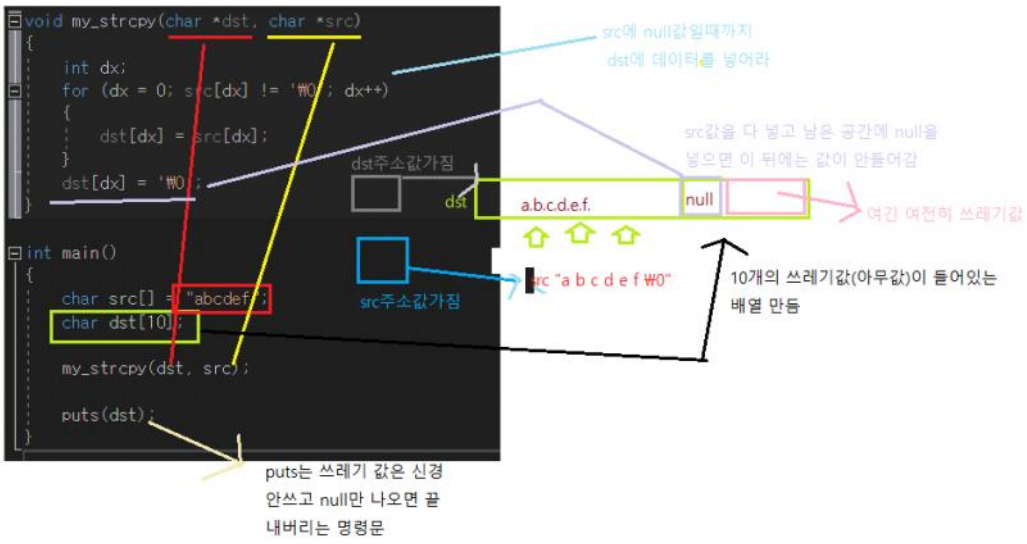

```
void func_a(int a) {}
void func_b(int *pb) {}
void func_c(int (*pc)[3])//배열포인터 : 2차원배열의 선두주소는 배열포인터가 받아야함
{}
void func_d(int **pd) {}
void func_e(int **pe) {}
int main()
{
	int a;			//일반변수
	int b[3];		//1차원배열
	int c[2][3];	//2차원배열
	int *d;			//싱글포인터	//아직 값을 몰라 가리키는게 없으니까
	int *e[3];		//포인터배열

	// 메모리 한개짜리는 &를 붙여야 주소고, 배열은 배열이름이 주소임
	func_a(a);		//a:값, &a:주소
	func_b(b);		//b:주소(배열명이니까 주소다!)
	func_c(c);		//c:주소
	func_d(&d);		//d:값 //&d 주소	//지금 d는 값이 없으니까 잘못된 코드임
	func_e(e);		//e:주소

}
```

9. 배열 복사

2018년 10월 20일 토요일 오후 9:41

3. 배열 복사

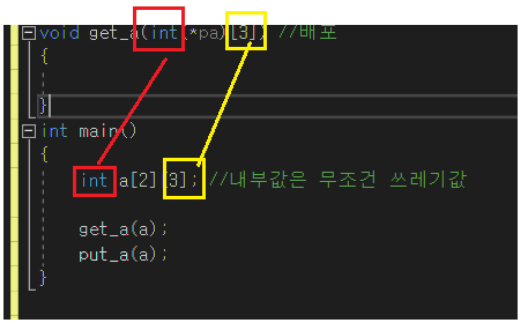
```
void my_strcpy(char *dst, char *src)
{
	int dx;
	for (dx = 0; src[dx] != '₩0'; dx++)
	{
		dst[dx] = src[dx];
	}
	dst[dx] = '₩0';
}

int main()
{
	char src[] = "abcdef";
	char dst[10];

	my_strcpy(dst, src);

	puts(dst);
}
```



10. 2차원 배열을 2차원,1차원처리

2018년 10월 20일 토요일 오후 9:42

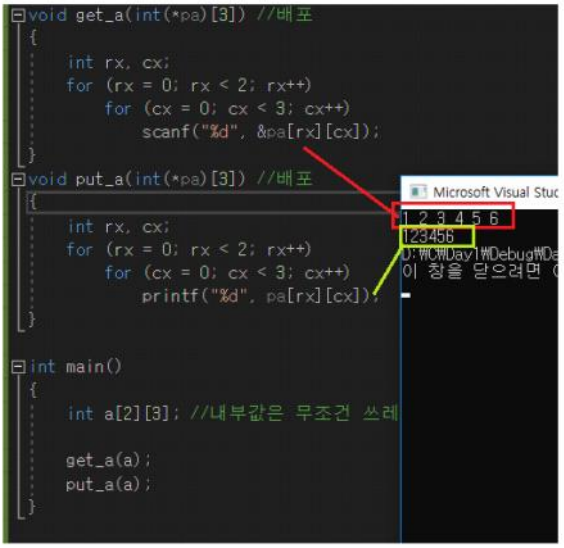
절대적으로 같아야함!!!



4. 2차원 배열을 2차원 답게 처리

```
void get_a(int(*pa)[3]) //배포
{
	int rx, cx;
	for (rx = 0; rx < 2; rx++)
		for (cx = 0; cx < 3; cx++)
			scanf("%d", &pa[rx][cx]);
}
void put_a(int(*pa)[3]) //배포
{
	int rx, cx;
	for (rx = 0; rx < 2; rx++)
		for (cx = 0; cx < 3; cx++)
			printf("%d", pa[rx][cx]);
}

int main()
{
	int a[2][3]; //내부값은 무조건 쓰레기값

	get_a(a);
	put_a(a);
}
```

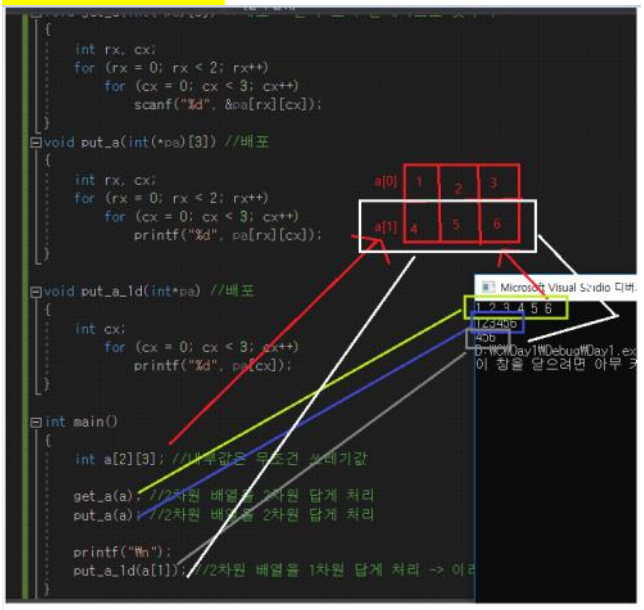
4-2 2차원배열 2차원으로 처리



-------1차원으로 처리---------

```
void get_a(int(*pa)[3]) //배포 : 열의 크기 절대적으로 맞추기
{
	int rx, cx;
	for (rx = 0; rx < 2; rx++)
		for (cx = 0; cx < 3; cx++)
			scanf("%d", &pa[rx][cx]);
}
void put_a(int(*pa)[3]) //배포
{
	int rx, cx;
	for (rx = 0; rx < 2; rx++)
		for (cx = 0; cx < 3; cx++)
			printf("%d", pa[rx][cx]);
}

void put_a_1d(int*pa) //배포
{
	int cx;
		for (cx = 0; cx < 3; cx++)
			printf("%d", pa[cx]);
}

int main()
{
	int a[2][3]; //내부값은 무조건 쓰레기값

	get_a(a); //2차원 배열을 2차원 답게 처리
	put_a(a); //2차원 배열을 2차원 답게 처리

	printf("₩n");
	put_a_1d(a[1]); //2차원 배열을 1차원 답게 처리 -> 이러면 한개짜리 메모리 날라감 2번째줄의 선두데이터 주소
}
```

4-3 2차원배열 1차원으로처리

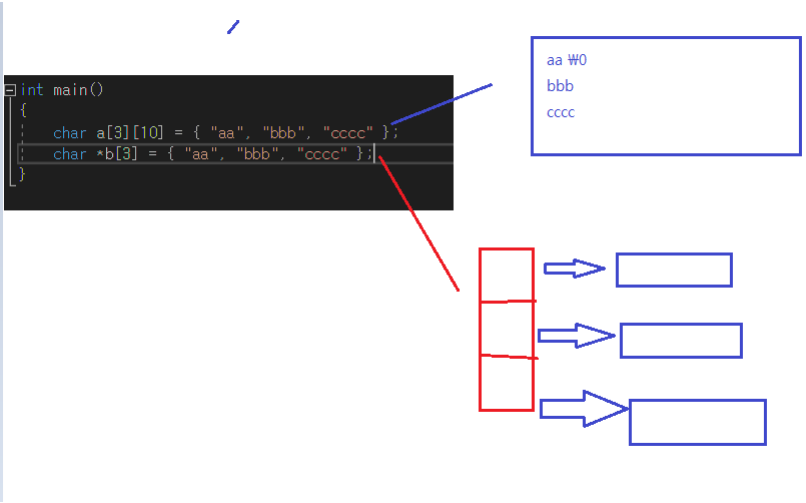


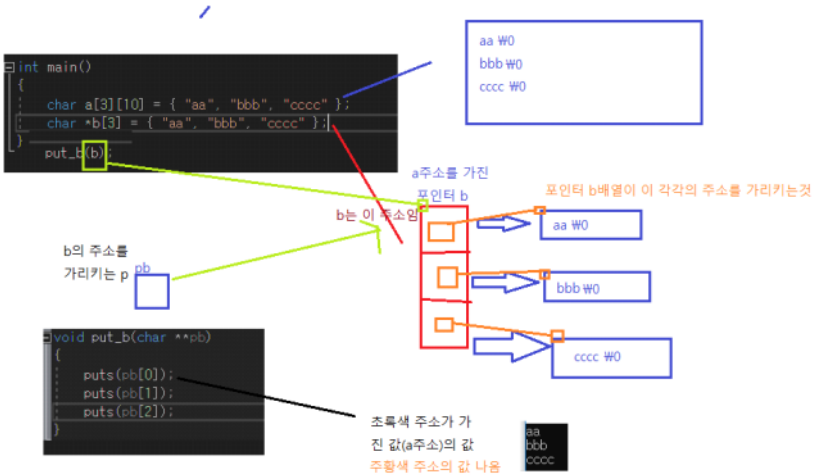
11. 포인터배열

2018년 10월 20일 토요일 오후 9:43

5. 포인터 배열

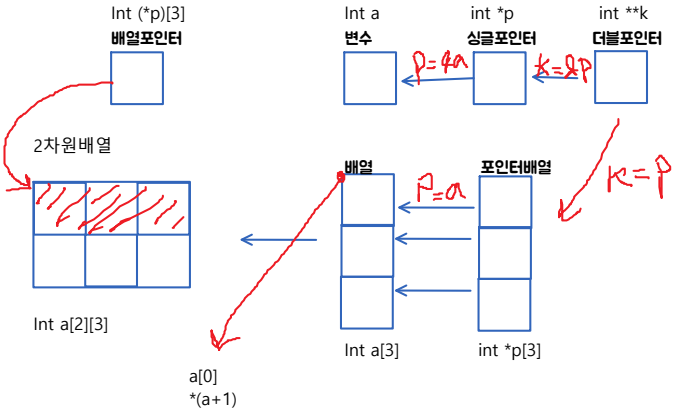
```
void put_b(char **pb)
{
	puts(pb[0]);
	puts(pb[1]);
	puts(pb[2]);
}


int main()
{
	char a[3][10] = { "aa", "bbb", "cccc" };
	char *b[3] = { "aa", "bbb", "cccc" };

	put_b(b);
}
```

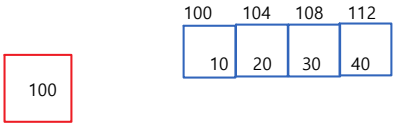




----------포인터 정리----------

2018년 10월 23일 화요일 오전 10:00





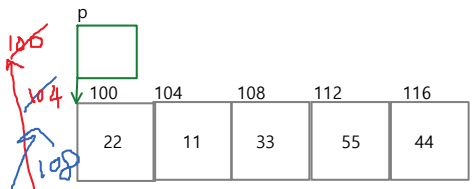
값 : 주소

*p	10:100
*(p+1)	20 : 100
*p+1	11:100
*p += 1	p = 11 :100
*p ++ -> p = p+1	10 : 104
(*p)++	20(다음에는21) : 104
++ *p	22 : 104
* ++p	30:108

(*104)번지값 20출력하고 20에++해라

현재 포인터가 보고있는 값을 ++

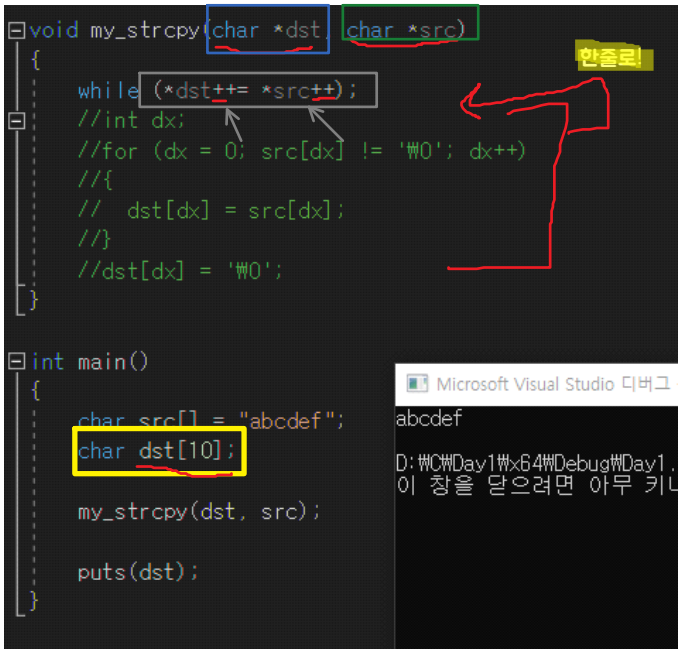
<예제>

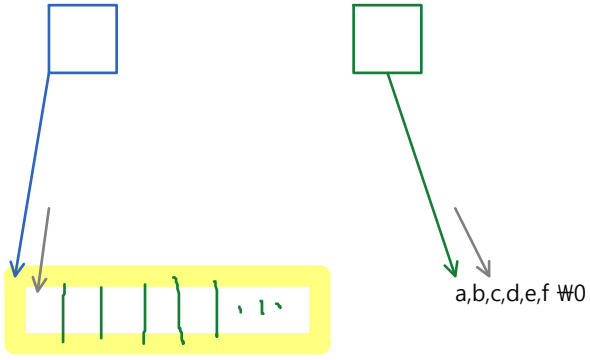


*p+=1	23:100	값 :영구	**포인터 값에 +1**
*p++	23:104	주소:영구	현재 p가 가리키는 주소의 값 출력하고 주소에 ++
++ *p	12:104	값:영구	104번지에 있는 값을 ++해라
*(p+1)	33 :104	주소:임시	
(*p)++	12(13):104	값:영구	**값을 ++ 한다. 다음 이 주소 출력시 13**
*++p	33:108	주소:영구	
*p+1	34:108	값:임시	
	값:주소	**변한것:지속**	

->예제

2018년 10월 23일 화요일 오전 10:31





코드가 짧아지고, 걸어다니면서 수박 던지는 느낌으로 코드가 빨라짐

전처기

2018년 10월 23일 화요일 오전 10:37

#ifndef apple (apple이 있니?) : 만약 apple이 없다면 이 코드는 true!
위의 코드가 true가 되면 아래코드로

```
#define apple  : 너 사과없어? 그럼 이거 실행해!
--코드들
-----
---
--

#endif          => 이렇게 끝남 나중에
```

#ifdef apple : 이제 위에서 apple 받았으니까 이 코드는 false!!!
위의 코드가 false가 되면 밑에 코드 생략

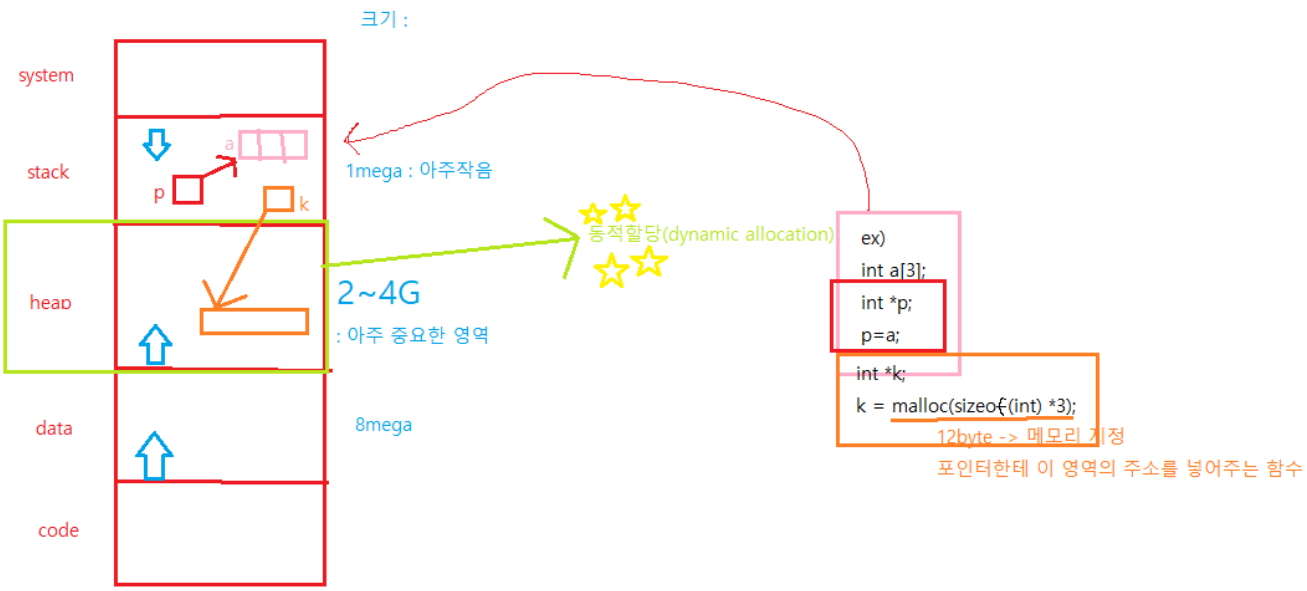
```
#define apple
--코드들
-----
---
--

#endif          => 이렇게 끝남 나중에
```

=> 정의되지 않았으면 정의 해줄게, 정의가 됐다면 안 해줘!

#pragma once -> 만약 #ifndef가 참조되지 않으면 이걸로 막겠다 !

12. 동적 메모리 할당

2018년 10월 20일 토요일 오후 9:43

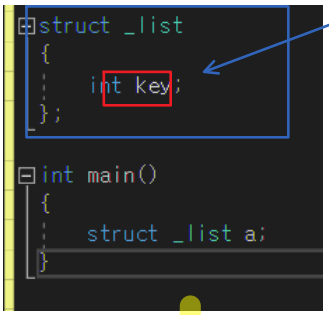


12-2. 동적 할당

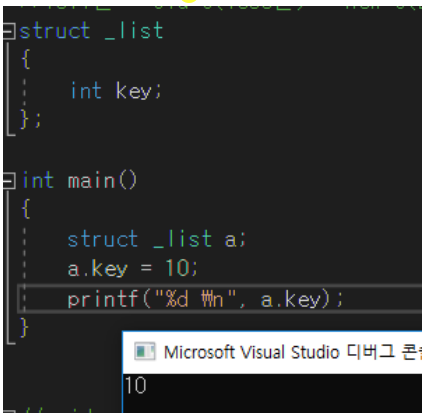
2018년 10월 23일 화요일 오전 10:49

변수

List구조에 int형 변수 메모리가 들어감



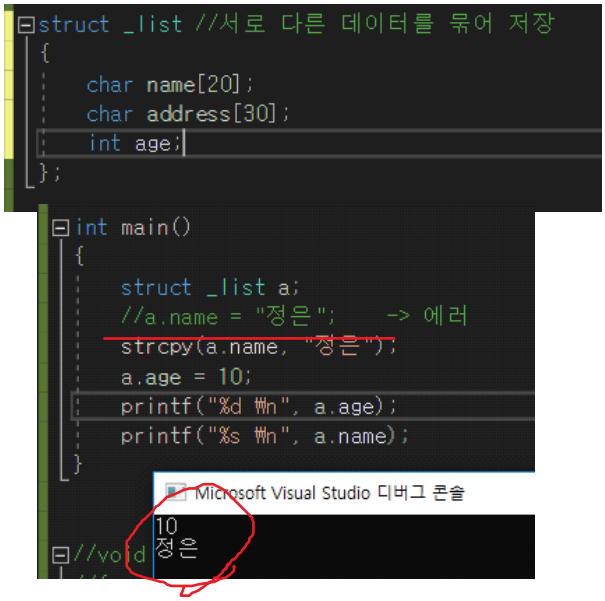
```
struct _list
{
    int key;
};

int main()
{
    struct _list a;
}
```



```
struct _list
{
    int key;
};

int main()
{
    struct _list a;
    a.key = 10;
    printf("%d ₩n", a.key);
}
```

Microsoft Visual Studio 디버그 콘

10

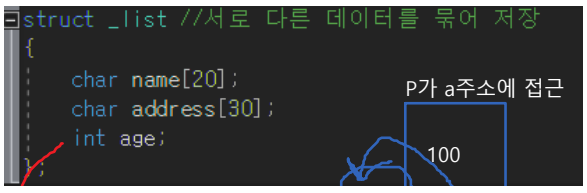


```
struct _list //서로 다른 데이터를 묶어 저장
{
    char name[20];
    char address[30];
    int age;
};

int main()
{
    struct _list a;
    //a.name = "정은";     -> 에러
    strcpy(a.name, "정은");
    a.age = 10;
    printf("%d ₩n", a.age);
    printf("%s ₩n", a.name);
}
```

Microsoft Visual Studio 디버그 콘솔

10
정은

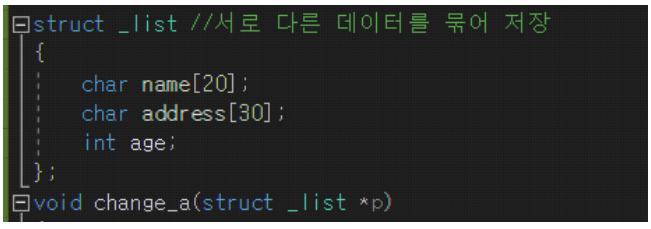
변수 구조체 바꾸려면



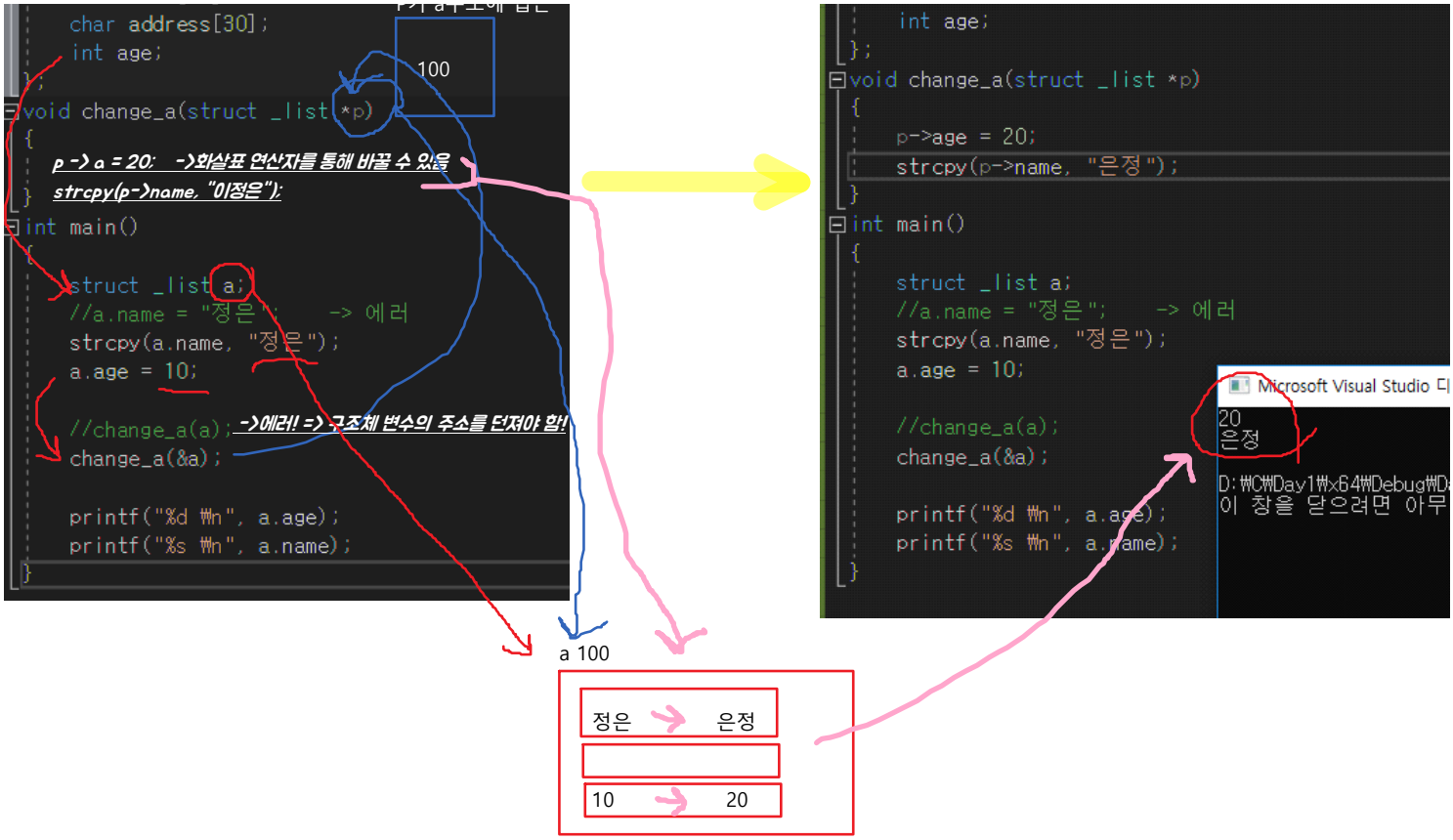
```
struct _list //서로 다른 데이터를 묶어 저장
{
    char name[20];
    char address[30];
    int age;
};
```

P가 a주소에 접근

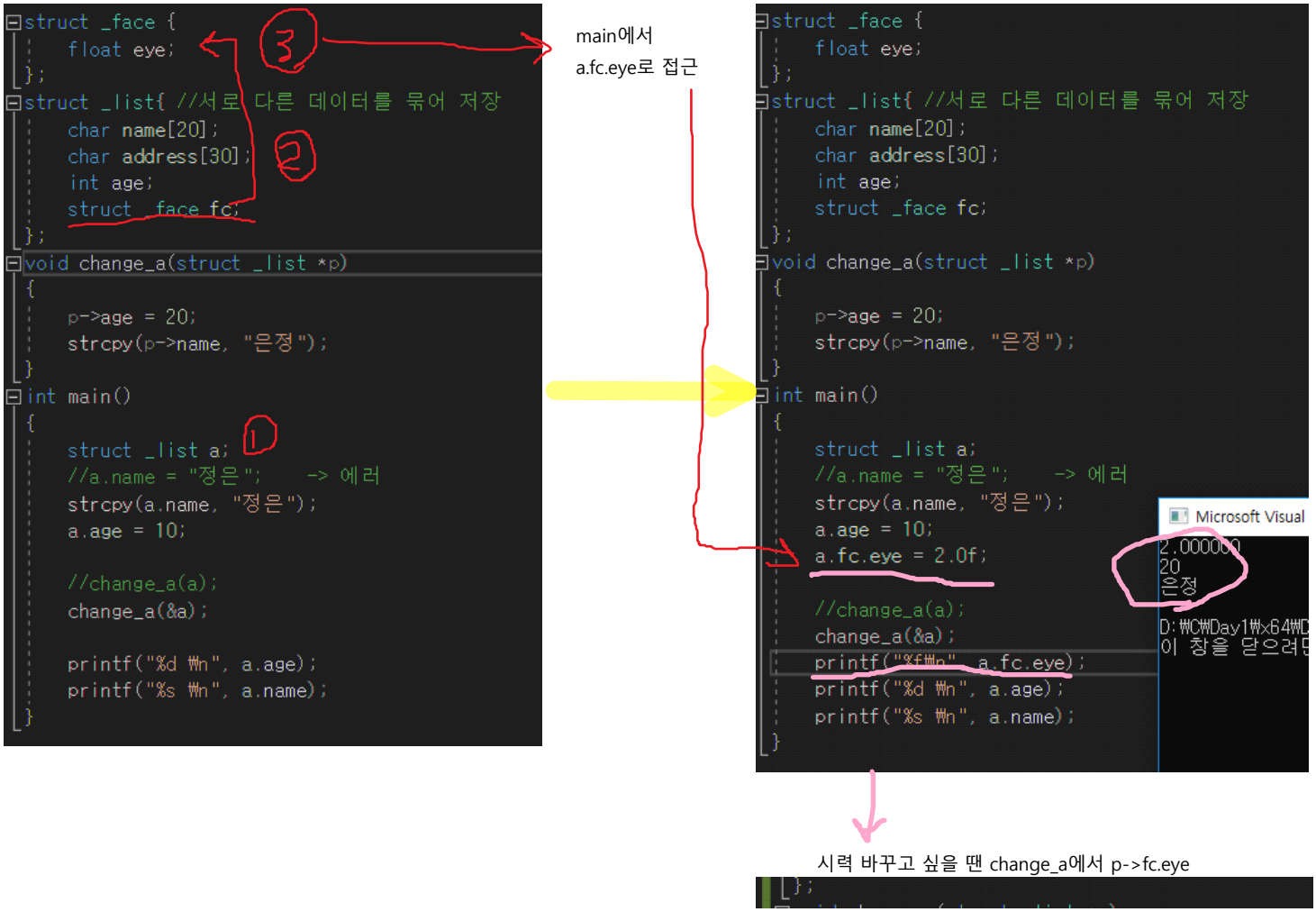
100

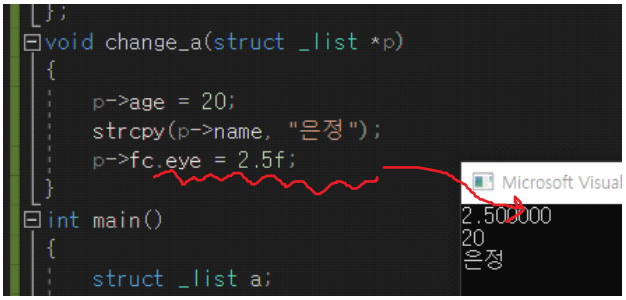


```
struct _list //서로 다른 데이터를 묶어 저장
{
    char name[20];
    char address[30];
    int age;
};
void change_a(struct _list *p)
```



코드 볼 때는 struct먼저보고 main



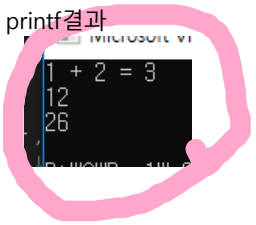

};
void change_a(struct _list *p)
{
p->age = 20;
strcpy(p->name, "은정");
p->fc.eye = 2.5f;
}
int main()
{
struct _list a;
Microsoft Visual
2.500000
20
은정

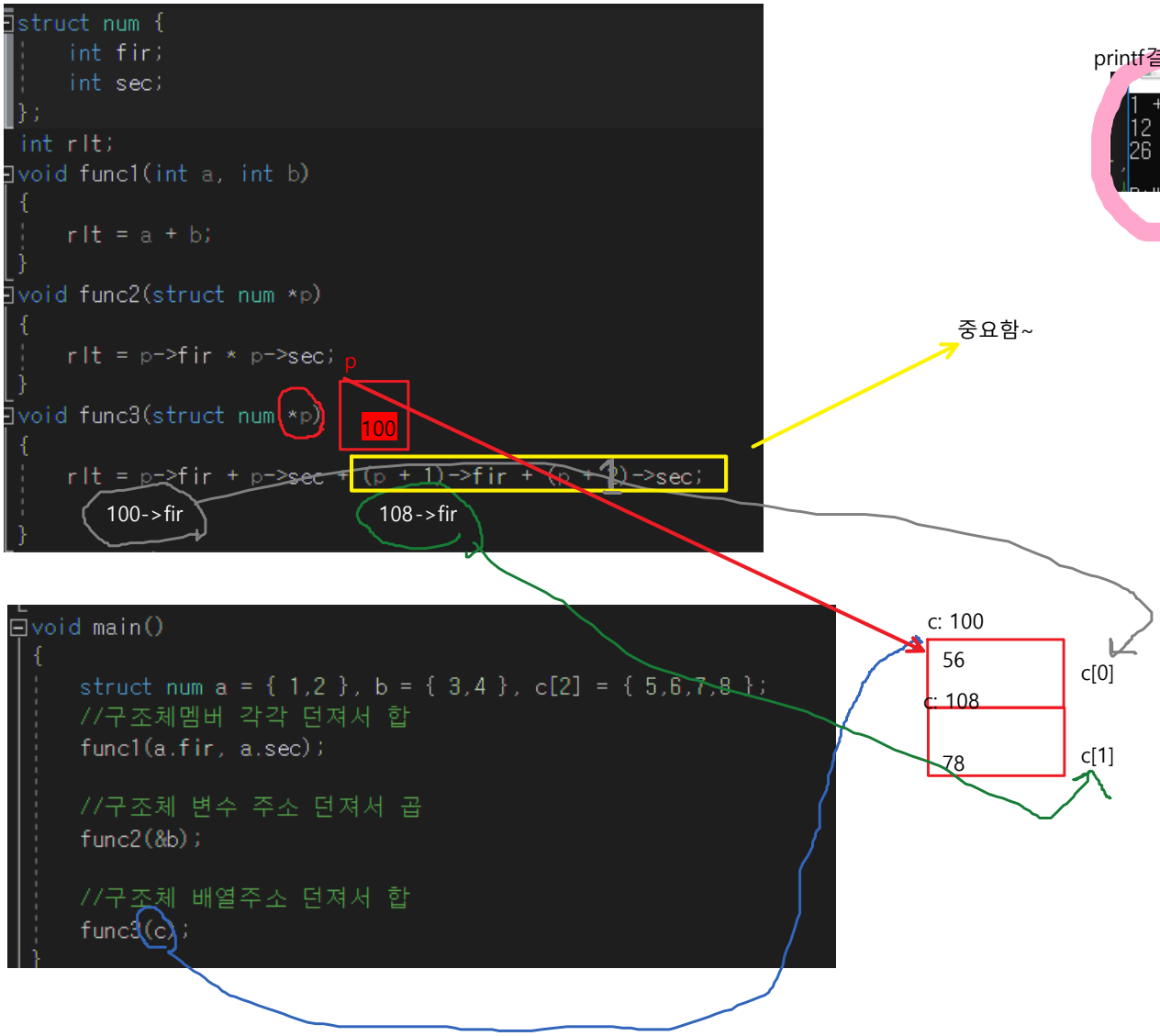
12-2-1 예제1

2018년 10월 23일 화요일 오전 11:27

```
struct num {
    int fir;
    int sec;
};
int rlt;
void func1(int a, int b)
{
    rlt = a + b;
}
void func2(struct num *p)
{
    rlt = p->fir * p->sec;
}
void func3(struct num *p)
{
    rlt = p->fir + p->sec + (p + 1)->fir + (p + 1)->sec;
}
```

```
void main()
{
    struct num a = { 1,2 }, b = { 3,4 }, c[2] = { 5,6,7,8 };
    //구조체멤버 각각 던져서 합
    func1(a.fir, a.sec);

    //구조체 변수 주소 던져서 곱
    func2(&b);

    //구조체 배열주소 던져서 합
    func3(c);
}
```



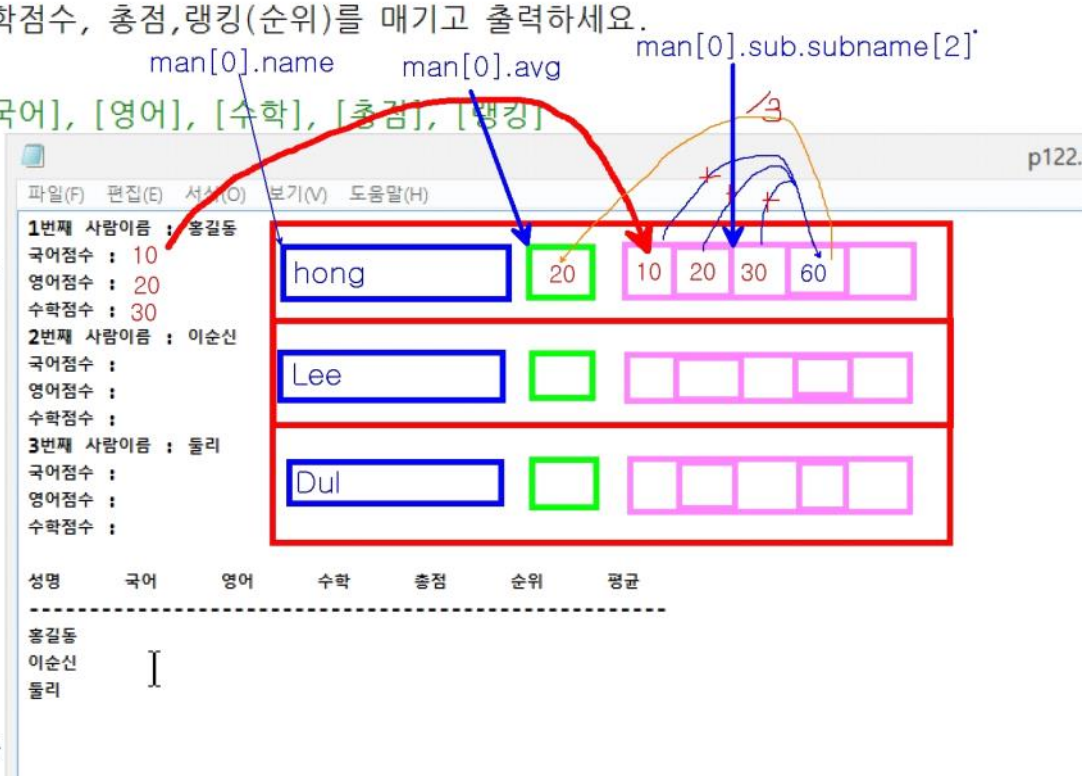


12-2-2 예제2

2018년 10월 23일 화요일 오전 11:48

문)
국어점수, 영어점수, 수학점수, 총점,랭킹(순위)를 매기고 출력하세요.

```
struct subject{
    int subname[5];//[국어], [영어], [수학], [총점], [랭킹]
};
struct sung{
    char name[10];
    float avg;  //평균
    struct subject sub;
};
```



결과)

1번째 사람이름 : 홍길동

```c
struct subject {
    int subname[5]; //국 영 수 총점 랭킹

};
struct sung {
    char name[10]; //이름
    float avg;     //평균
    struct subject sub; //각과목의 점수

};
int main()
{
    struct sung man[3]; //3명이 있어야함 [0]:홍길동 [1]:이순신 [2]:돌리
    char *s[] = { "국어", "영어", "수학" };
    int dx, cx;

    // 이름-자바칩 // 국영수
    for (dx = 0; dx < 3; dx++) //3번
    {
        scanf("%s", man[dx].name); // == gets(man[dx].name);   ->사람이름
        man[dx].sub.subname[3] = 0; //총점의 초기화  -> 밑에서 총점 한번 구하면 초기화하고 다시
        //국영수
        for (cx = 0; cx < 3; cx++)
        {
            printf("%s점수 : ",s[cx]);
            scanf("%d", &man[dx].sub.subname[cx]);
            man[dx].sub.subname[3] += man[dx].sub.subname[cx];
        }
        //점수 다 받으면 평균구행
        //왼쪽은 정수형이라 오른쪽 형변환(캐스팅)해줘야 값이 들어감
        man[dx].avg = man[dx].sub.subname[4]/(float)3;
    }


   for (dx = 0; dx < 3; dx++) //3번
   {
       for (cx = 0; cx < 3; cx++)
       {
           if (man[dx].sub.subname[3] < man[cx].sub.subname[3]) {
               man[dx].sub.subname[4]++;
           }
       }

    for (dx = 0; dx < 3; dx++) //3번
    {
        printf("%s", man[dx].name);
        for (cx = 0; cx < 5; cx++)
        {
            printf("%d \t", man[dx].sub.subname[cx]);
        }
        puts(""); //개행문자 한칸 내림
    }


}
```

13 알고리즘

2018년 10월 25일 목요일 오후 3:27

책 : 이재규 - C로 배우는 알고리즘 1,2 - 원론적으로 파기 좋음
윤성우(자료구조) - 쉬움, 인강제공
한빛미디어(뇌를 자극하는 알고리즘)

1. List(순서가있는자료구조) - Array(1,2,3), struct(linked list) --> kernel(double linked list)
2. stack - LIFO(last input first out) - 컴파일러, 함수 콜 스택 (비트연산자, 전처리기)
3. queue - FIFO - buffer, 층별순회(tree)-최단거리 알고리즘(graph)
   -우선순위 큐(priority Queue) - 우선순위탐색알고리즘(최소비용순회)
   - heap - 게임 A-Star
4. tree - 기업의 70%(사원) -이진트리(균형잡힌 이진트리에 대해 설명 : Red-Black Tree)
5. graph - 다익스트라알고리즘(최단거리)
6. sort(search) : 알고리즘의 꽃 - 일반적 정렬, queue를 알아야, tree 를 알아야 되는 정렬이 있음

   hash - 갤럭시 노트9,8,5는 내부구조가 hash-tree -> 속도가 항상 일정하게 빠름

   균형 잡힌:자동균형

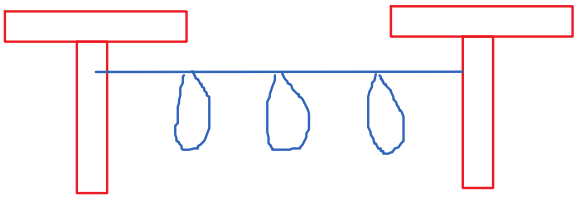
7. 자료구조(저장), 알고리즘(처리)

14 - single linked List

2018년 10월 25일 목요일 오후 3:43

typedef-> 구조체의 이름 미리 정함

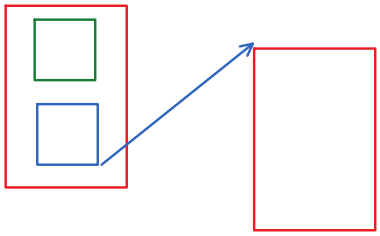
```
typedef struct _list List;
```



알고리즘? 양쪽에 기둥을 세우고 빨래 줄에 빨래를 거는 느낌
데이터를 어떻게 걸지.

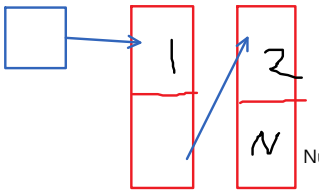
```
struct _list {
    int key;
    struct _list *next;
};
```

기차로 치면 key는 승객
Next는 기차와 기차간 연결



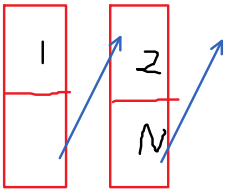
포인트 접근

1

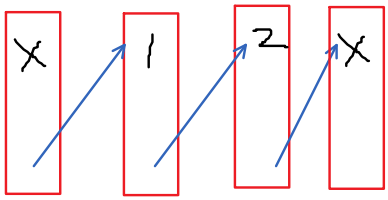


Null을보고 끝인걸 암

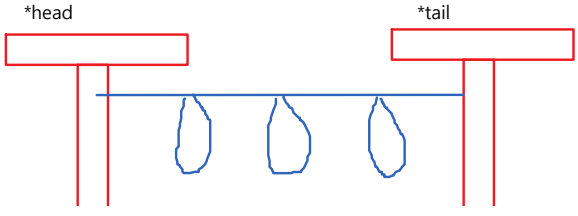
2



3

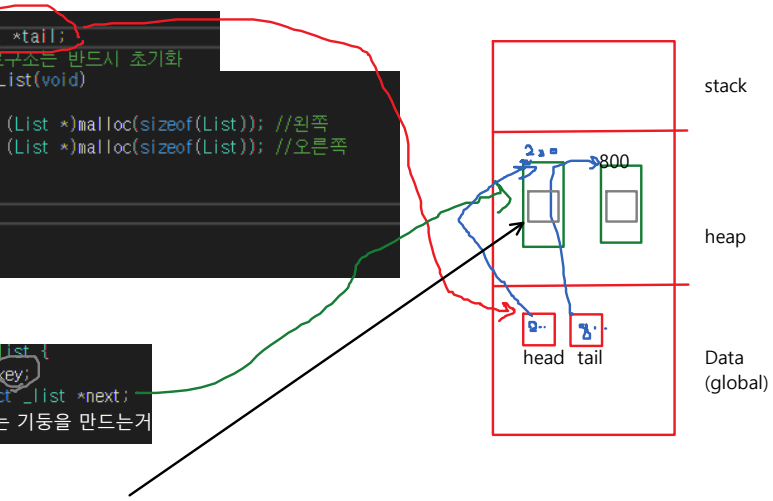



```
//모든 자료구조는 반드시 초기화
void init_List(void)
{
}
int main()
{
```



```
}
int main()
{
}
```

```
List *head, *tail;
//모든 자료구조는 반드시 초기화
void init_List(void)
{
    head = (List *)malloc(sizeof(List)); //왼쪽
    tail = (List *)malloc(sizeof(List)); //오른쪽
}
int main()
{
}
```




```
struct _list {
    int key;
    struct _list *next;
}; *next는 기둥을 만드는거
```

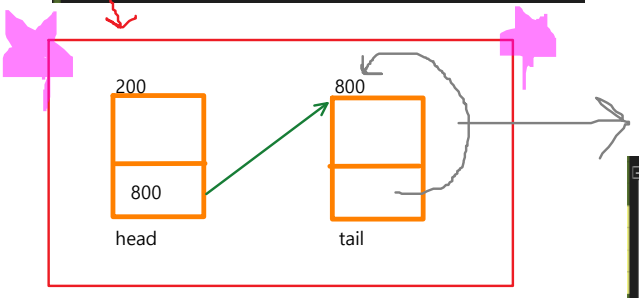
head -> key = 50; : 헤드가 보고있는 키 위치에 50을 넣겠다

```
typedef struct _list List;
struct _list {
    int key;
    struct _list *next;
};

List *head, *tail;
//모든 자료구조는 반드시 초기화 ->list, stack,Queue,tree,graph
void init_List(void)
{
    head = (List *)malloc(sizeof(List)); //왼쪽
    tail = (List *)malloc(sizeof(List)); //오른쪽
    head->next = tail; //빨래줄
}
int main()
{
    init_List(); //빨래기둥2개, 빨래줄
}
```

기둥만들고

빨래줄만듬

빨래 함수를 부르는 코드



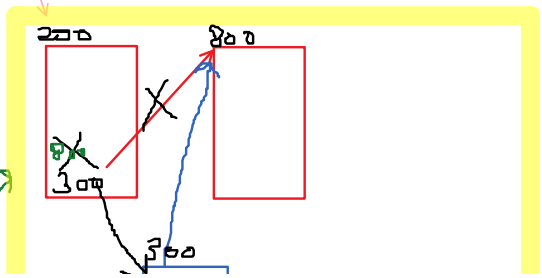
tail->next = tail;

```
void init_List(void)
{
    head = (List *)malloc(sizeof(List)); //왼쪽
    tail = (List *)malloc(sizeof(List)); //오른쪽
    head->next = tail; //빨래줄
    tail->next = tail;
}
```

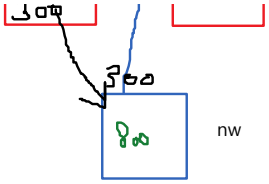
head

새로운 게 생기면 내가 보고있던걸
새로운 애 한테 가리키게 하고 내가 새로운 애를 봄

```
typedef struct _list List;
struct _list {
    int key;
    struct _list *next;
};

List *head, *tail;
//모든 자료구조는 반드시 초기화
void init_List(void)
{
    head = (List *)malloc(sizeof(List)); //왼쪽
    tail = (List *)malloc(sizeof(List)); //오른쪽
    head->next = tail; //빨래줄
    tail->next = tail;
}
```



```
    head = (List *)malloc(sizeof(List)); //왼쪽
    tail = (List *)malloc(sizeof(List)); //오른쪽
    head->next = tail; //빨래줄
    tail->next = tail;
}
void insert_head(int data) //숫자 빨래화
{
    List *nw = (List *)malloc(sizeof(List));
    nw->key = data;
}
int main()
{
    init_List(); //빨래기둥2개, 빨래줄
    insert_head(3); //첫번째 빨래 널기
}
```

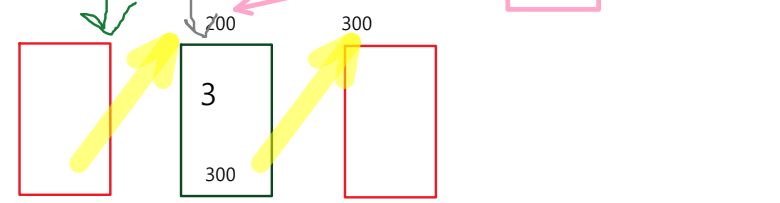


nw->next = head->next; : 내가 보는 거 니가 봐

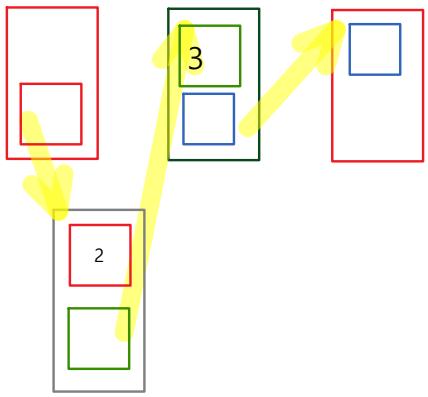
Head->next = nw : 새로운 건 내가 볼게

Head포인터가 가리키는 영역임

```
void print_List(void) //오로지 빨래만 봄
{
    List *wk;  //빨래 걷는 녀석
    wk = head->next;  //첫번째 빨래를 봐야함
    printf("%d₩n", wk->key);
}
int main()
{
    init_List(); //빨래기둥2개, 빨래줄
    insert_head(3); //첫번째 빨래 널기
    print_List();  //널린 빨래 확인
}
```

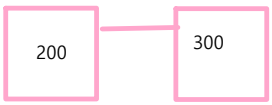


만약 또 추가되면




```
void print_List(void) //오로지 빨래만 봄
{
    List *wk;  //빨래 걷는 녀석
    wk = head->next;  //첫번째 빨래를 봐야함
    printf("%d₩n", wk->key);
}
int main()
{
    init_List(); //빨래기둥2개, 빨래줄
    insert_head(3); //첫번째 빨래 널기
    print_List();  //널린 빨래 확인
}
```

이제 얘가 300을 가리켜야함. wk = wk->next;




```
void print_List(void) //오로지 빨래만 봄
{
    List *wk;  //빨래 걷는 녀석
    wk = head->next;  //첫번째 빨래를 봐야함
    printf("%d₩n", wk->key);
    wk = wk->next;
    printf("%d₩n", wk->key);
}
int main()
{
    init_List(); //빨래기둥2개,
    insert_head(3); //첫번째 빨
```

Microsoft Visual Studio

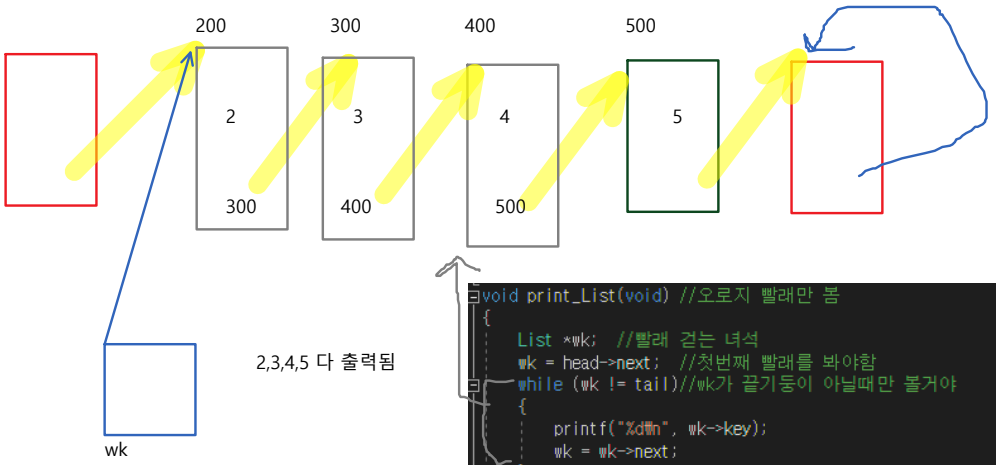
2
3

D:₩C₩Day1₩x64₩Debug
이 창을 닫으려면 아

```
int main()
{
    init_List(); //빨래기둥2개
    insert_head(3); //첫번째 빨
    insert_head(2); //첫번째 빨
    print_List();  //널린 빨래
}
```

```
2
3
D:₩C₩Day1₩x64₩Debug
이 창을 닫으려면 아
```

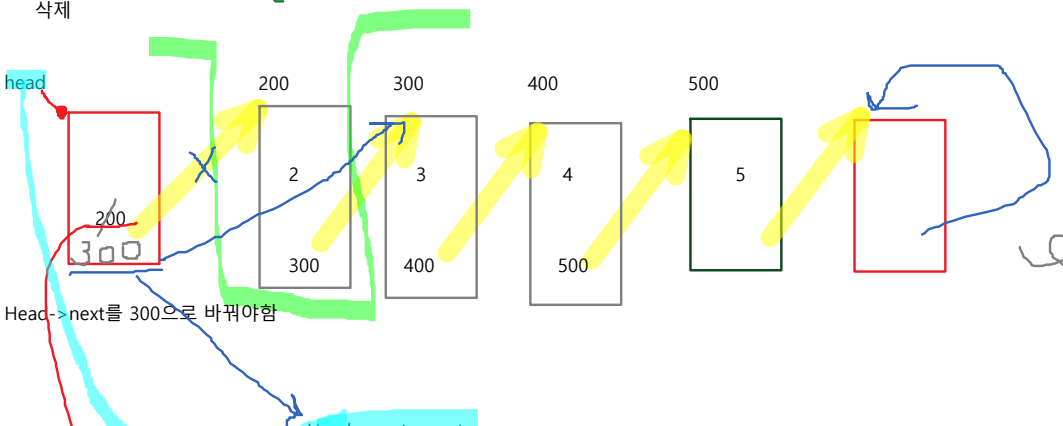
빨래가 여러 개면?

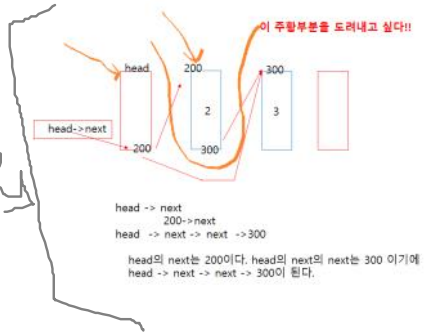


2,3,4,5 다 출력됨

```
void print_List(void) //오로지 빨래만 봄
{
    List *wk;  //빨래 걷는 녀석
    wk = head->next;  //첫번째 빨래를 봐야함
    while (wk != tail)//wk가 끝기둥이 아닐때만 볼거야
    {
        printf("%d₩n", wk->key);
        wk = wk->next;
    }
    //printf("%d₩n", wk->key);
    //wk = wk->next;
    //printf("%d₩n", wk->key);
}
```

삭제





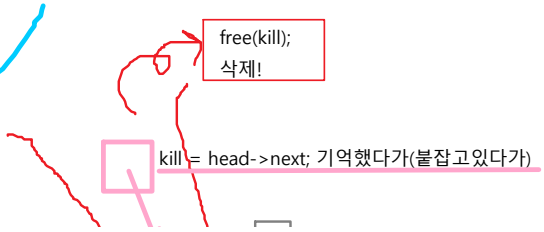
==> 이 방법은 200을 그냥 메모리 냅두니까 손해 -> 완전히 처리해야함

kill = head->next; 기억했다가(붙잡고있다가)

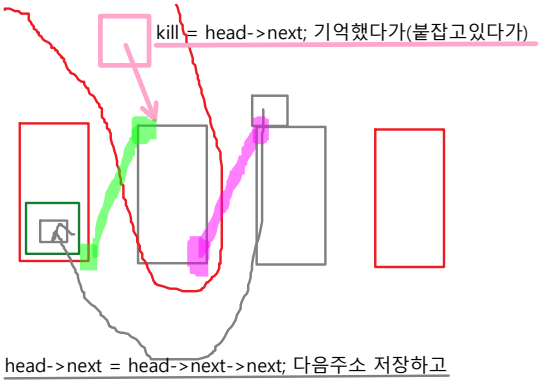
head->next = head->next->next; 다음주소 저장하고

free(kill); 삭제!

```
void delete_head(void)
{
    List *kill;
    kill = head->next;
    head->next = head->next->next;
```



```
{
    List *kill;
    kill = head->next;

    head->next = head->next->next

    free(kill);
}
int main()
{
    init_List(); //빨래기둥2개, 빨래줄
    insert_head(3); //첫번째 빨래 널기
    insert_head(2); //첫번째 빨래 널기
    print_List();  //널린 빨래 확인

    delete_head();
}
```

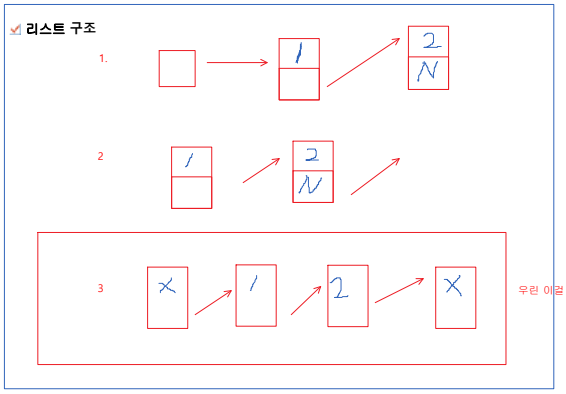


- 개념

2018년 10월 25일 목요일 오후 8:44

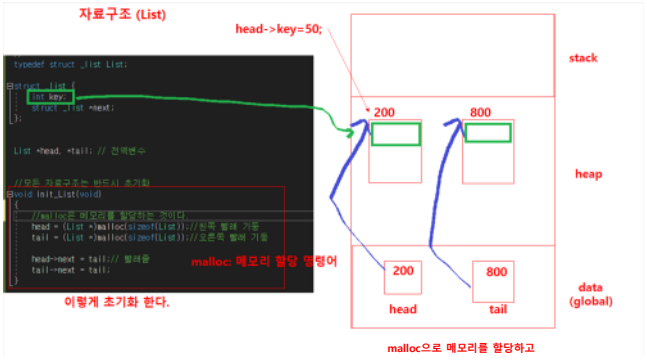
☑ 리스트
-순서가 있는 자료구조

☑ 리스트 구조



☑ 자료구조(List) 코드 적용(예제)

☑ 1. 메모리 할당하고 변수 설정하기

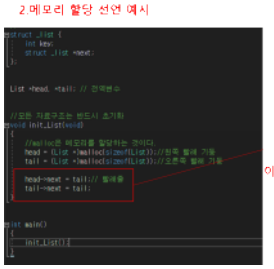


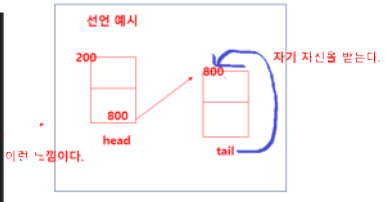
malloc으로 메모리를 할당하고
head와 tail의 메모리를 할당하고 주솟값으로 받는 구조이다.

200과 800은 예시다.

☑ 2. 초기화 및 두개의 메모리 생성

2 메모리 할당 선언 예시

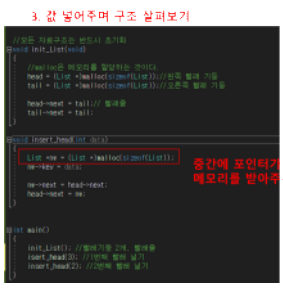




이런 느낌이다.

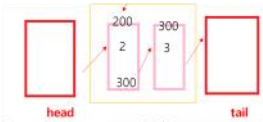
기초 세팅을 서로가 서로에게 연계될 수 있게 해준거다.

☑ 3. 값 넣어주며 구조 살펴보기



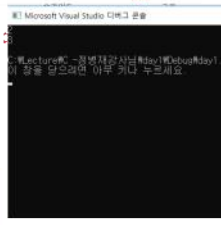
☑ 4.도와주는 포인터 만들기(메모리)

이 로직때문에 대부분의 공학도가 포기한다.!!



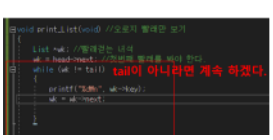
일하고 있는 녀석의 next를 가르킨다.

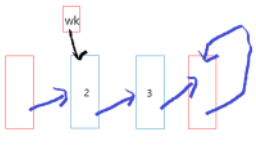
(2)가 head바로 앞에 들어가 된다.



☑ 5. 리스트로 모든 값 보여주기

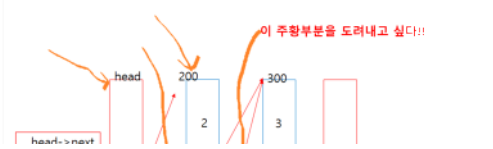
5.리스트 형태로 모든 값 보여주기

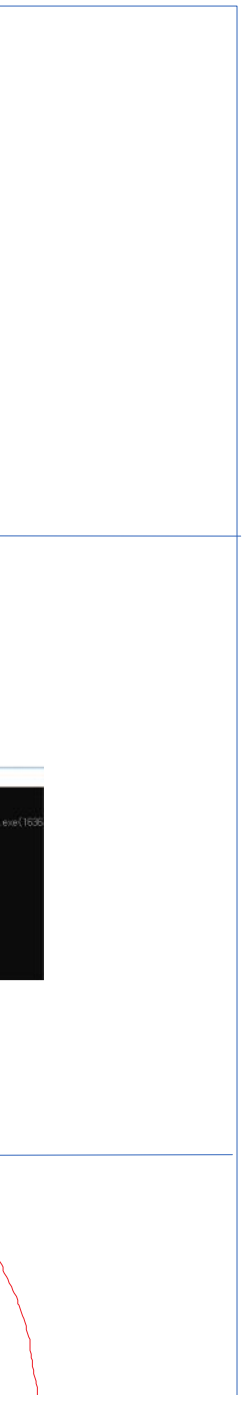


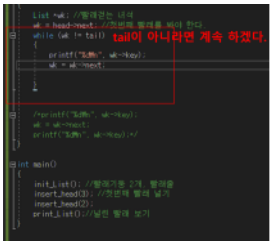


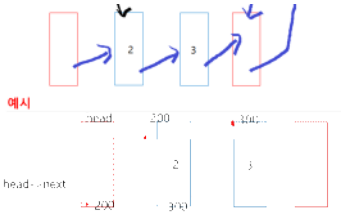
☑ 6-1값을 없애고 싶을 때 예시

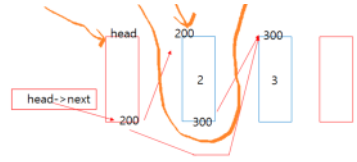
6.값을 없애고 싶을 때 로직 예시









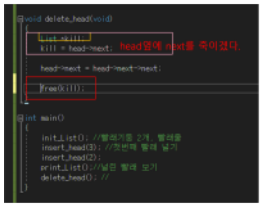


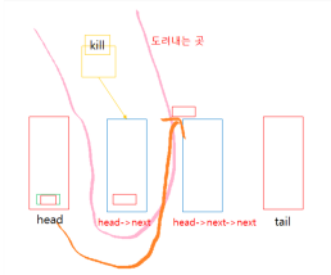
head -> next
200 -> next
next -> next -> next -> NULL

head의 next는 200이다. head의 next의 next는 300 이다.
head -> next -> next -> 300 된다.

6-2리스트로 모든 값 보여주기

6-2. 값을 없애고 싶을 때 적용

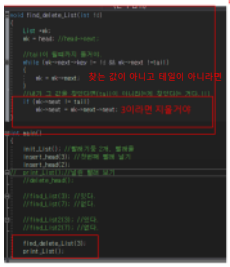


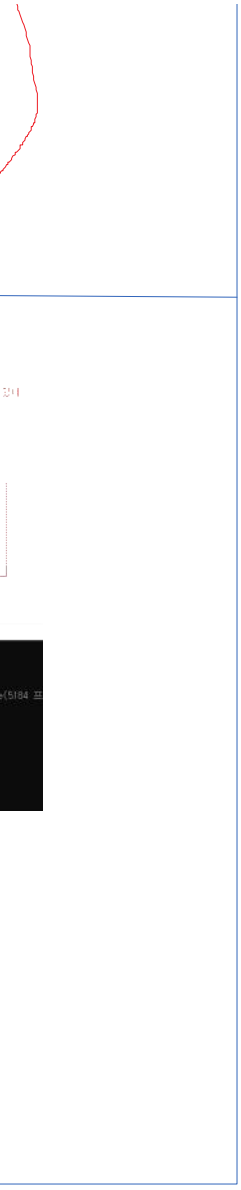


7.길찾기 예제

7. 값찾기

8.값 찾고 삭제하기

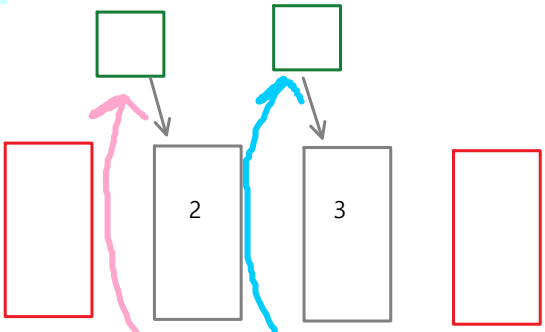




- 예제

2018년 10월 25일 목요일 오후 5:04

찾기




```
void find_List(int data)
{
    List *wk;
    wk = head->next;
    while (wk != tail)
    {
        if (wk->key == data)
            puts("있다.");
        wk = wk->next;

    }
}
int main()
{
    init_List(); //빨래기둥2개, 빨래줄
    insert_head(3); //첫번째 빨래 널기
    insert_head(2); //첫번째 빨래 널기
    print_List();  //널린 빨래 확인

    //delete_head();  //삭제

    find_List(3); //3이 있는지 없는지
}
```

```
//버전 1
void find_List(int data)
{
    List *wk;
    wk = head->next;
    while (wk != tail)
    {
        if (wk->key == data)
        {
            puts("있다.");
            break;
        }
        wk = wk->next;

    }
    if (wk == tail)
        puts("없다");
}
int main()
{
    init_List(); //빨래기둥2개, 빨래줄
    insert_head(3); //첫번째 빨래 널기
    insert_head(2); //첫번째 빨래 널기
    print_List();  //널린 빨래 확인

    //delete_head();  //삭제
    find_List(3);
    find_List(4); //3이 있는지 없는지
}
```

```
Microsof
2
3
있다.
없다

D:₩C₩Day1₩
```

```
//버전2
void find_List2(int data)
{
    List *wk;
    wk = head->next;
    while (wk->key != data && wk != tail)
    {
        wk = wk->next;
    }
    if (wk == tail)
        puts("없다");
    else
        puts("있다");
}

int main()
{
    init_List(); //빨래기둥2개, 빨래줄
    insert_head(3); //첫번째 빨래 널기
    insert_head(2); //첫번째 빨래 널기
    print_List();  //널린 빨래 확인

    //delete_head();  //삭제
    //find_List(3);
    //find_List(4); //3이 있는지 없는지
```

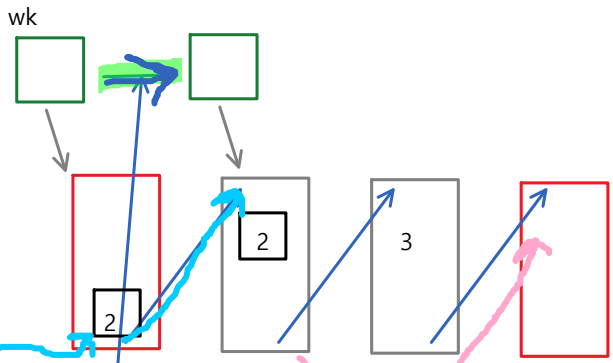
```
Microsc
2
3
있다
없다
```

```
    //delete_head();  //삭제
    //find_List(3);
    //find_List(4); //3이 있는지 없는지
    find_List2(3);
    find_List2(4);
}
```

```
3
있다
없다

D:₩CW₩Day1
이 창을 닫
```

찾아서 삭제하기

```
//찾아서 삭제
void find_delete_List(int fd)
{
    List *wk;
    wk = head;   //찾아서 삭제해야 하기에 head값을 줘야함

    while (wk->next->key != fd) //내가 찾는 값이 아니라면
    {
        wk = wk->next;   //wk를 다음값으로 옮겨
    }
}
```



2가
3 이아니니까

```
//찾아서 삭제
void find_delete_List(int fd)
{
    List *wk;
    wk = head;   //찾아서 삭제해야 하기에 head값을 줘야함

    while (wk->next->key != fd && wk->next != tail) //내가 찾는 값이 아니라면
    {
        wk = wk->next;   //wk를 다음값으로 옮겨
    }
    if (wk->next != tail)  //tail이면 찾았다는거 //위에서 tail을만나고 나왔다는 뜻은 fd를 못찾았다는거잖아
        wk->next = wk->next->next;
}
```

찾거나, 못찾으면 while끝,
wk->next->key != fd : 찾았다
wk->next != tail :못찾았다

Tail을 만나면 while문을 나오니까 -> 값을 못 찾았다
wk->next != tail이면 tail을 못 만나고 나온 거니까 중간에 fd가 있다는 뜻! 그럼
건너뛰라는 뜻

- 기능/예제

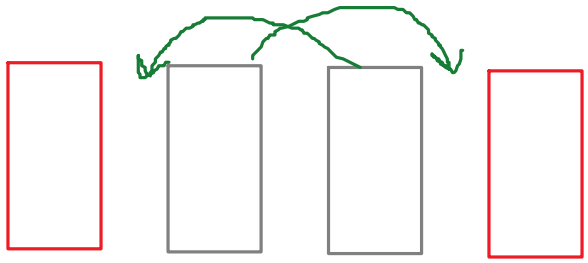
2018년 10월 26일 금요일 오전 10:04

1. **Move to front**

찾아서 맨앞으로 보내기

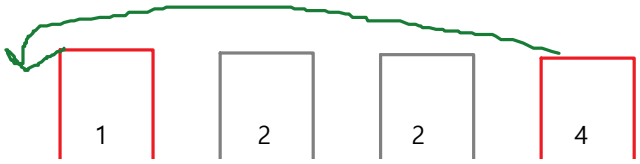

```
void movetofront(int ff)
{
    List *front;
    List *wk;
    wk = head;
    while (wk->next->key != ff && wk->next != tail)
    {
        wk = wk->next;
    }
    if (wk->next != tail)
    {
        front = wk->next;   //찾은 값을 기억해 앞으로 넣을거야 잡아놔
        wk->next = wk->next->next;
        insert_head(front->key);  //노드를 넣으면 안됨

    }

}
```

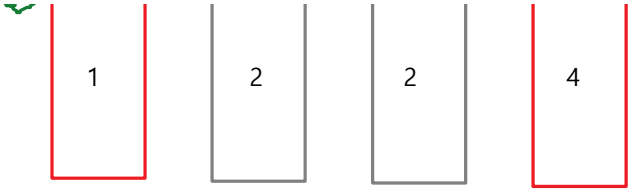
2. **transpose method //과제 ! 값 하나로 앞의 값이랑 무조건 바뀌게**



3. **Frequency count**

각 노드의 조회 횟수를 세서 제일 많이 검색한 것을 제일 앞으로 정렬




1
2
2
4

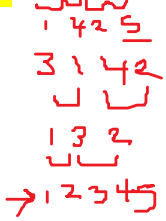
15 sort

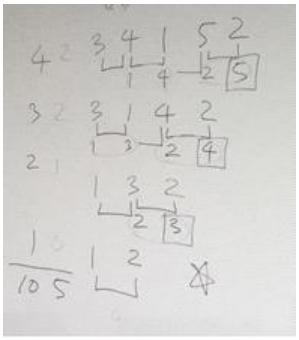
2018년 10월 26일 금요일 오전 11:09

sort

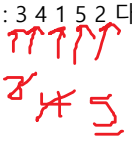
-버블, 선택, 삽입, 간접 삽입, 쉘, 퀵, 병합, 버킷, 분도수세기, tree, heap ... 정렬

버블 정렬 : 3 4 1 5 2 다 비교해서 바꾸는 거

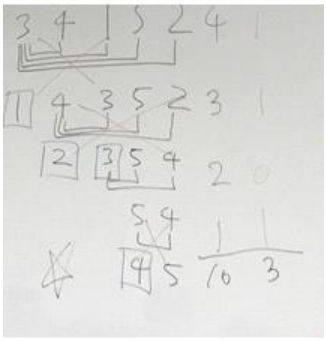




선택 정렬 : 3 4 1 5 2 다 비교해서 바꾸는 거



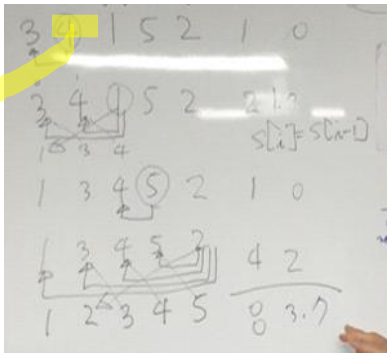
하나씩 비교해서 제일 큰 놈 하나만 기억 or
제일 작은 놈 하나 기억해서 앞으로



삽입 정렬 : 3 4 1 5 2 하나씩 비교해서 작으면 제일 앞으로 정렬

1번지 값이 기준임

2번지에 1번지값을 넣겠다
s[i] = s[i-1]



삽입 간접 정렬 :

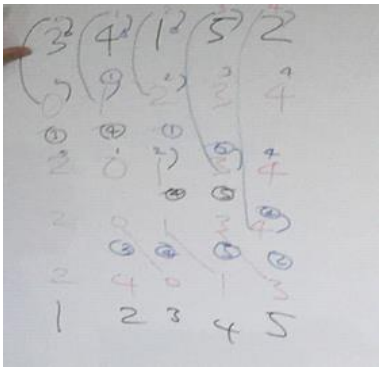
3 4 1 5 2

0 1 2 3 4 -> 위 배열의 index를 저장하는 배열을 하나 만듦

-> 1이 작으니까 1을 제일 앞 번지로

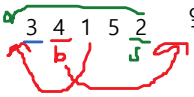
2 0 1 3 4 => 실제 배열 말고 번지수를 바꾸는 거임

2 4 0 1 3

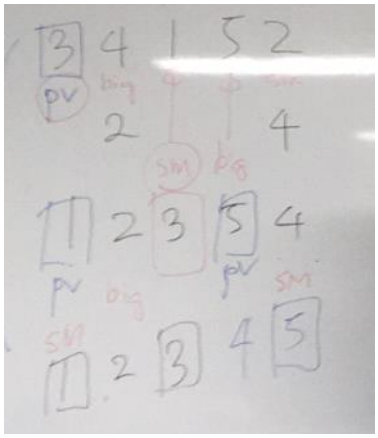


퀵 정렬 : **2. bsearch()**

1.Quick sort() : 3을 기준으로



앞에서 부터 3보다 큰 값을 big
뒤에서 부터는 3보다 작은 값 찾으러 다님 small
찾으면 small을 제일 앞으로 big을 제일 뒤로
바꿔버림 ==> **자기 정렬 끝나면 고정됨.**



-> 1 2 3 5 4 이 상태로 정렬 끝나고 이제 1을 기준으로 정렬 -> 자기 뒤에가 big이니까 바로 끝

->1 2 3 5 4 ->2 3 끝났으니까 이제 5를 기준으로 정렬 => 1 2 3 4 5

- 함수 포인터

2018년 10월 26일 금요일 오전 11:36

callback

cal(10,5,add)

=> cal(a,b,p(add함수의주소))

p(a,b); == add(10,5);

Ex)

int add(int a, int b);

int(*p)(int,int); -> add함수의 주소를 담을 수 있는 포인터

->int를 리턴하는 함수를 담은 포인터 p

===> **p = add;**

```
int add(int a, int b) { return a + b; }
int sub(int a, int b) { return a - b; }
int mul(int a, int b) { return a * b; }
int dvv(int a, int b) { return a / b; }

int main()
{
    int(*p)(int a, int b); // add,sub,mul,dvv 4개 다 저장 할 수 있는 함수
    p = add;
    printf("%d\n", p(10, 5));
    p = sub;
    printf("%d\n", p(10, 5));
    p = mul;
    printf("%d\n", p(10, 5));
    p = dvv;
    printf("%d\n", p(10, 5));
}
```

```
Microsoft Visual Studio 디버그 콘솔
15
5
50
2
```

반복문으로 돌릴 수 있음

```
int add(int a, int b) { return a + b; }
int sub(int a, int b) { return a - b; }
int mul(int a, int b) { return a * b; }
int dvv(int a, int b) { return a / b; }

int main()
{
    int(*p[4])(int, int) = { add,sub,mul,div }; //함수 포인터 배열
    int dx;
    for (dx = 0; dx < 4; dx++)
    {
        printf("%d\n", p[dx](10, 5));
    }
```

```
Microsoft Visual Studio 디버그 콘솔
15
5
50
2
```

Leaf function call

```
int add(int a, int b) { return a + b; }
int sub(int a, int b) { return a - b; }
int mul(int a, int b) { return a * b; }
int dvv(int a, int b) { return a / b; }

int middle(int a, int b, int(*p)(int, int))
{
    return p(a, b);
}
int main()
{
    printf("%d\n", middle(10, 5, add));
}
```

```
Microsoft Visual St
15
```

add (10,5)

↳ Callback

Main() middle() add()

Leaf function call

바로 main으로

- qsort

2018년 10월 26일 금요일 오후 4:19

```c
int main()
{
    int a[] = { 3,4,1,5,2 };
    int dx;
    // 배열이름, 데이터 수, 하나의 데이터의 메모리 4byte
    //qsort(a, 5, 4, int cmp);

    qsort(a, sizeof(a) / sizeof(a[0]), sizeof(a[0]), int cmp);

    for (dx = 0; dx < 5; dx++)
        printf("%d ", a[dx]);
}
```

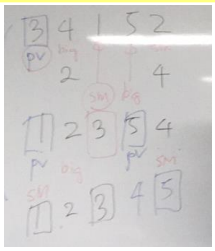
```c
int int_cmp(const void *a, const void *b)
{
    return *(int *)a - *(int *)b;
}
int main()
{
    int a[] = { 3,4,1,5,2 };
    int dx;
    // 배열이름, 데이터 수, 하나의 데이터의 메모리 4byte
    //qsort(a, 5, 4, int cmp);

    qsort(a, sizeof(a) / sizeof(a[0]), sizeof(a[0]), int_cmp);

    for (dx = 0; dx < 5; dx++)
        printf("%d ", a[dx]);
}
```

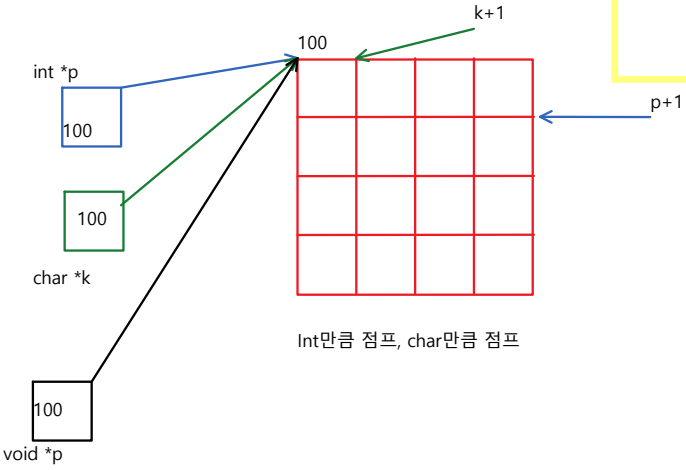
Microsoft Visual Studio 디버그 콘솔

```
1 2 3 4 5
D:₩C₩Day1₩x64₩Debug₩Day1.exe(
```

너가 갖고있는 100번지 int형 주소야 이 주소 안의 값 의미

qsort내부알고리즘을 통해 정렬됨





Int만큼 점프, char만큼 점프

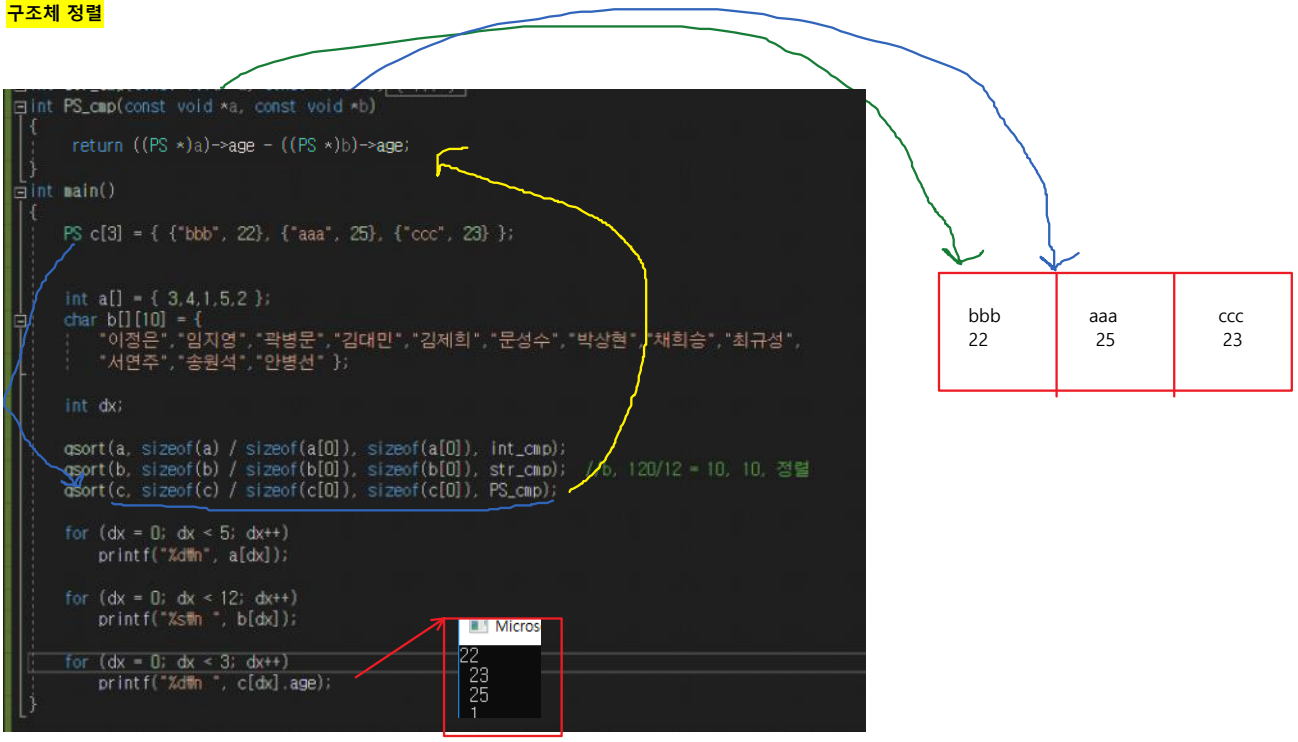
strcmp("Abc", "ABC");

string 비교해주는 함수

반드시 캐리터형 주소로 받으라고 알려줘야함

```c
int str_cmp(const void *a, const void *b)
{
    return strcmp((char *)a, (char *)b);
}
int main()
{
    char b[][10] = {
        "곽병문","김대민","김제희","문성수","박상현",
        "서연주","송원석","안병선","이정은","임지영",
        "채희승","최규성" };

    int dx;

    qsort(b, sizeof(b) / sizeof(b[0]), sizeof(b[0]), str_cmp);  //b, 120/12 = 10, 10, 정렬

    for (dx = 0; dx < 5; dx++)
        printf("%d ", a[dx]);
}
```

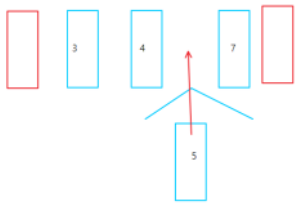
구조체 정렬

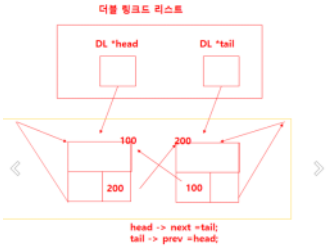

int PS_cmp(const void *a, const void *b)
{
return ((PS *)a)->age - ((PS *)b)->age;
}
int main()
{
PS c[3] = { {"bbb", 22}, {"aaa", 25}, {"ccc", 23} };
int a[] = { 3,4,1,5,2 };
char b[][10] = {
"이정은","임지영","곽병문","김대민","김제희","문성수","박상현","채희승","최규성",
"서연주","송원석","안병선" };
int dx;
qsort(a, sizeof(a) / sizeof(a[0]), sizeof(a[0]), int_cmp);
qsort(b, sizeof(b) / sizeof(b[0]), sizeof(b[0]), str_cmp);
qsort(c, sizeof(c) / sizeof(c[0]), sizeof(c[0]), PS_cmp);
for (dx = 0; dx < 5; dx++)
printf("%d₩n", a[dx]);
for (dx = 0; dx < 12; dx++)
printf("%s₩n ", b[dx]);
for (dx = 0; dx < 3; dx++)
printf("%d₩n ", c[dx].age);
}
Micros
22
23
25
1
bbb
22
aaa
25
ccc
23

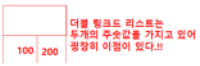
16. double linked list

2018년 10월 26일 금요일 오후 5:08

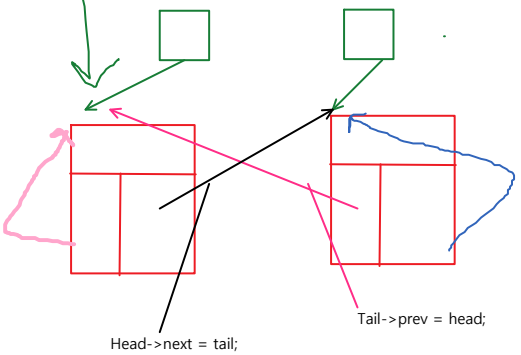
정렬로 넣기







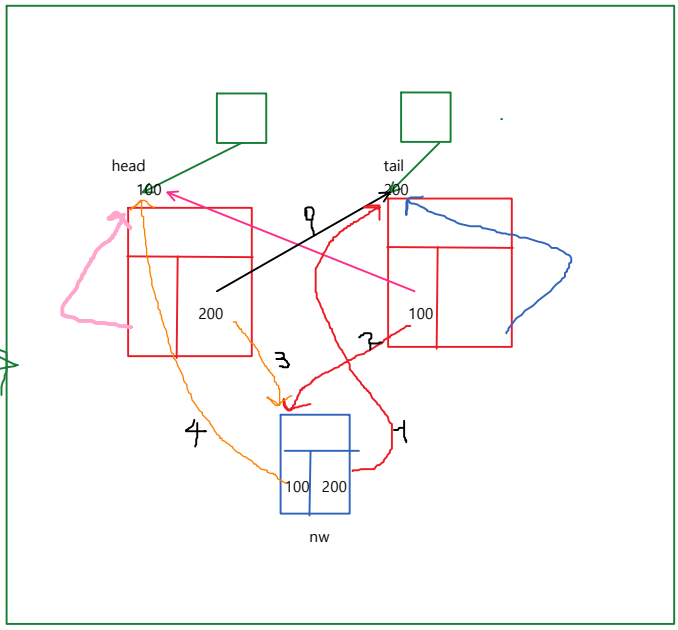

```
typedef struct _dlist DL;
struct _dlist {  //double linked list
    int key;
    struct _dlist *prev;
    struct _dlist *next;
};

DL *head, *tail; //빨래 양쪽 기둥
//모든 자료구조는 반드시 초기화
void init_head(void)
{
    head = (DL*)malloc(sizeof(DL));  //왼기둥
    tail = (DL*)malloc(sizeof(DL));  //오른기둥
}
```

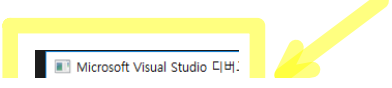


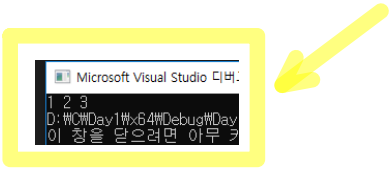

```
head->next = tail;
tail->prev = head;
head->prev = head;
tail->next = tail;
```

새로운 데이터 등장

```
typedef struct _dlist DL;
struct _dlist {  //double linked list
    int key;
    struct _dlist *prev;
    struct _dlist *next;
};

DL *head, *tail; //빨래 양쪽 기둥
//모든 자료구조는 반드시 초기화
void init_head(void)
{
    head = (DL*)malloc(sizeof(DL));  //왼기둥
    tail = (DL*)malloc(sizeof(DL));  //오른기둥

    head->next = tail;
    tail->prev = head;
    head->prev = head;
    tail->next = tail;
}

void insert_dlist(int data)
{
    DL *nw = (DL *)malloc(sizeof(DL));
    nw->key = data;

    nw->next = head->next;  //0,1  nw의next자리에 head의 next값을 넣어라
    head->next->prev = nw;  //2  헤드넥스트가 보는 위치 값의 prev위치가 nw를 봐라
    head->next = nw;        //3 헤드 넥스트가 nw보고
    nw->prev = head;        //nw의 prev가 head를 봐라
}
int main()
{
    init_dlist(); //빨래 기둥과 빨래 줄 완성
    insert_dlist(3);
}
```







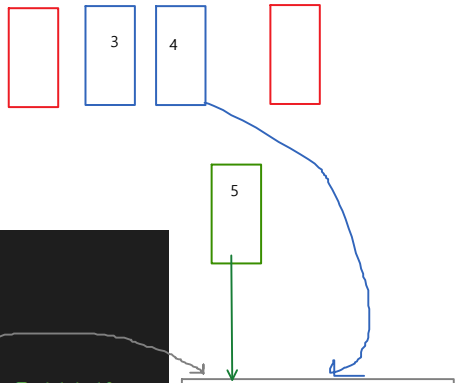

```
void print_dlist(void)
{
    DL *wk = head->next;
    while (wk != tail)
    {
        printf("%d ", wk->key);
        wk = wk->next;
    }
    puts("₩n----------");
    wk = tail->prev;
    while (wk != head)
    {
        printf("%d ", wk->key);
        wk = wk->prev;
    }
}
int main()
{
    init_dlist(); //빨래 기둥과 빨래 줄 완성
    insert_dlist(3);
    insert_dlist(2);
    insert_dlist(1);
    print_dlist();
}
```

```
1 2 3
----------
3 2 1
```

Ordered_insert

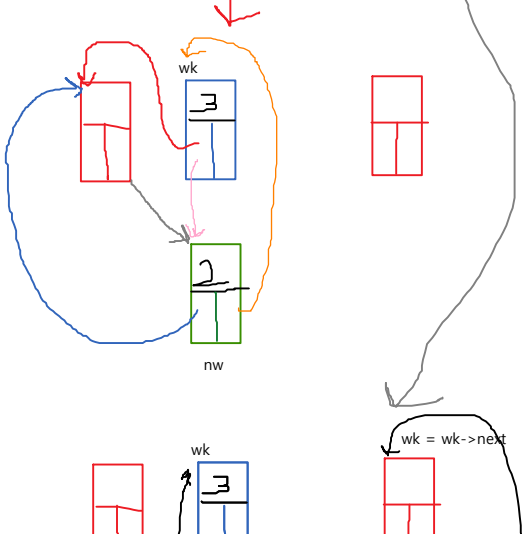
- 나보다 큰놈 앞에다 데이터 넣어

```
void ordered_insert(int data)
{
    DL *nw = (DL *)malloc(sizeof(DL));
    nw->key = data;
}
```



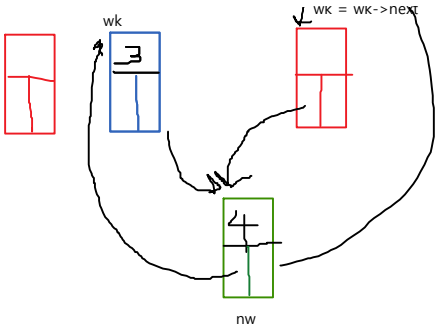

```
void ordered_insert(int data)
{
    DL *wk;
    wk = head->next;  //첫번째 빨래 보기

    DL *nw = (DL *)malloc(sizeof(DL));
    nw->key = data;

    while (wk->key < data && wk != tail)  //data보다 큰 wk->key를 만나면 멈춤
    {
        wk = wk->next;
    }
    //wk의 앞쪽에 nw를 삽입해야함
    nw->prev = wk->prev;
    wk->prev->next = nw;
    wk->prev = nw;
    nw->next = wk;
}
```

초록색보다 파란색 값이 작으면
wk->next로 넘어가




```
int main()
{
    init_dlist(); //빨래 기둥과 빨래 줄 완성
    //insert_dlist(3);
    //insert_dlist(2);
    //insert_dlist(1);
    //print_dlist();

    ordered_insert(3);
    ordered_insert(4);
    ordered_insert(1);
    ordered_insert(2);
    print_dlist();
}
```

```
1 2 3 4
----------
4 3 2 1
D:₩C₩Day1₩x64₩Debug₩Day1
이 창을 닫으려면 아무 키
```


wk
wk = wk->next
3
4
nw