Busan Software Meister High School

MICROPROCESSOR

2309 양유빈

20230316

마이크로프로세서

손정웅선생님

OVERVIEW

- 배열 포인터
- 1차원 배열 포인터
- 2차원 배열 포인터
- 함수 포인터
- 구조체
- 구조체 포인터
- 구조체와 패딩
- 구조체와 공용체
- 프로그래밍 과제

배열 포인터

array pointer

📌배열 포인터: 주소를 배열로 만든 것

📌배열 포인터 선언: char data[]={"가나다", "ABC") //선언, 가나다의 시작 주소를 저장

```c
// 배열포인터 선언
#include<stdio.h>
int main(){
    char data[]={"가나다", "ABC"};
    printf("%s", data[0]);
    //%s: 첫번째 주소 ~null 앞까지 출력
    return 0;
}
//출력: 가나다
```

```c
#include<stdio.h>

int main(){
    int a[3]={10,20,30};

    printf("%p %p %p\n", a,a+0,&a[0]);
    printf("%p %p\n", a+1, &a[1]);
    printf("%p %p\n", a+2, &a[2]);
    printf("%ld %ld %ld\n", sizeof(a), sizeof(a+0), sizeof(&a[0])); //12byte
    return 0;
}
```

// a==a+0==&a[0] -> 모두 주소가 같다.

1차원 배열 포인터

one-dimensional array pointer


```c
// 1차원 배열 포인터
int a[2]; //배열명은 주소 값이다, a[0] 배열명은 첫 번째 배열의 주소값

#include<stdio.h>
int main(){
    int a[5]={0,1,2,3,4};
    int *p;
    p=a; //a 배열이 주소이기 때문에 p=&a X
    printf("%d", *(p+1)); //*p == *a
    return 0; //*(a+1) 4byte씩 떨어짐
}
//출력: 1
```

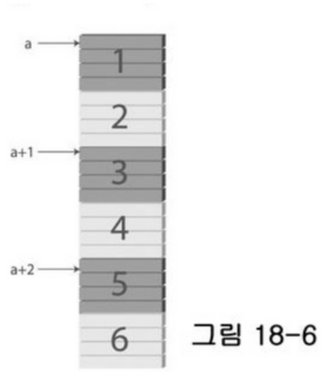


그림 18-6

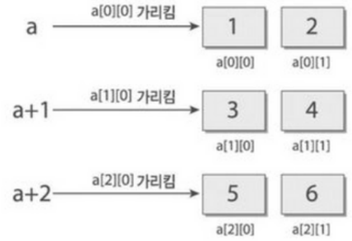


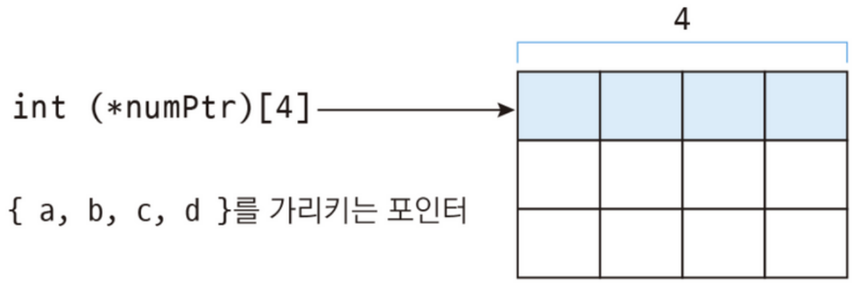
그림 18-7

```c
#include<stdio.h>

int main(){
    int a[3] = {10,20,30};
    int *data[3];
    int *p;
    p=a;
    for(int i=0;i<3;i++){
        printf("%d %d %d\n", *(p+i), *&p[i], p[i]);
    }
    for(int j=0;j<3;j++){
        printf("%d %d %d\n", *(a+j), *&a[j], a[j]);
    }

    printf("%ld", sizeof( &data[1]));
    return 0;
}
```

2차원 배열 포인터

2D array pointer

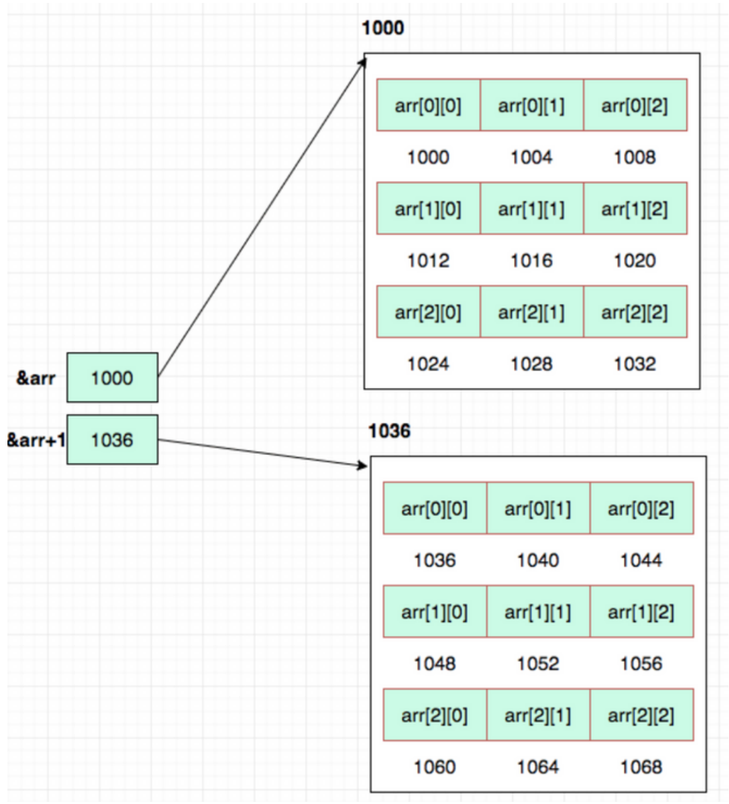
▼ 그림 34-3 2차원 배열과 포인터



참고 | int *numPtr[4]

int (*numPtr)[4]에서 괄호를 뺀 int *numPtr[4]는 int형 포인터 4개를 담을 수 있는 배열이라는 뜻입니다.
즉, 괄호가 있으면 배열을 가리키는 배열 포인터, 괄호가 없으면 포인터를 여러 개 담는 포인터 배열입니다.

```
int num1, num2, num3, num4;
int *numPtr[4] = { &num1, &num2, &num3, &num4 };    // int형 포인터를 4개 담는 배열
```




```c
#include<stdio.h>

int main(){
    int a[3][4] = {
        {1, 2, 3, 4},
        {5, 6, 7, 8},
        {9, 10, 11, 12}
    };
    int (*p)[4] = a;
    printf("%p\n", *p);
    printf("%p\n", *a);
    printf("%d\n", p[2][1]);
    printf("%ld\n", sizeof(a));
    printf("%ld\n", sizeof(p));
    return 0;
}
```

함수 포인터

function pointer

// 함수포인터란 함수를 가리킬 수 있는 포인터를 의미.

함수 포인터 선언 하는 법:

①int ②(*ptrSum)③(int a,int b)

일반 포인터와 마찬가지로 주소를 가리킬때는 *을 사용해서 포인터라고 알려줍니다.

①은 함수의 반환형을 의미합니다.
②는 함수포인터의 이름을 의미합니다. (변수명과 같이 임의로 정해줍니다.)
③은 매개변수를 의미합니다. 매개변수가 없을 때는 빈 괄호나 void를 사용합니다.

네, 위 세가지만 지켜주면 됩니다.

그러니까 ptrSum이라는 함수포인터는 반환형이 int형이고 매개변수 2개를 갖는데, 둘 다 int형 매개변수인 함수포인터가 되겠습니다.

Q. 왜 씀 함수 포인터?ㅋ

A. 예를 들면 함수 자체를 매개변수로 받고 싶을때가 있을 겁니다. 여러 사람과 협업을 해야하거나, 라이브러리를 제공할때, 함수에 함수자체를 매개변수로 받아야할 때가 있습니다. 누가 어떤 함수를 필요로 할지 모르니, 어떤 형식으로 함수를 정의해서 매개변수로 전달하게 되면 그 함수를 내부에서는 호출하게 되는 식으로 말이죠. + 클린코드

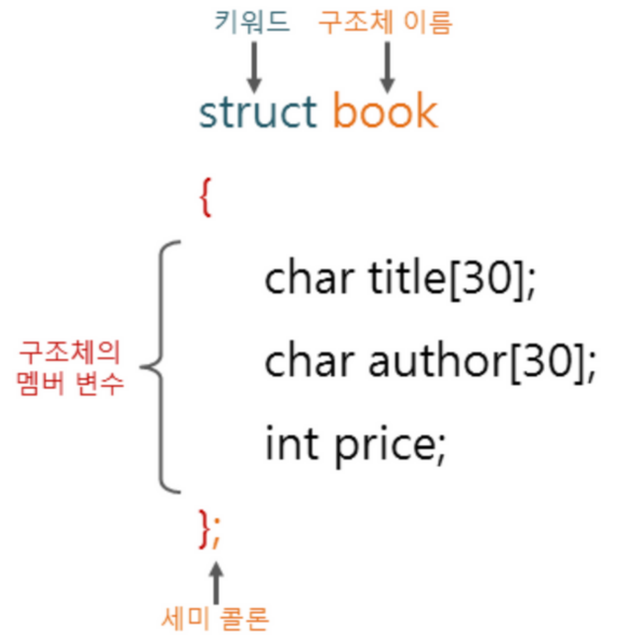
```c
#include<stdio.h>
int add(int a, int b){
    return a+ b;
}
int sub(int a, int b){
    return a-b;
}
int main(){
    int (*fp)(int, int);
    fp = add;
    printf("결과 값:%d\n", fp(10, 20));
    fp=sub;
    printf("결과 값:%d\n", fp(10, 20));
    return 0;
}
```

구조체

structure

복잡한 데이터 표현 가능

다음은 book이라는 이름의 구조체를 정의하는 그림입니다.




```c
#include<stdio.h>

struct score{
    int mic;
    int lin;
    int pro;
};

int main(){
    struct score s1;
    s1.mic = 60;
    s1.lin = 70;
    s1.pro = 80;
    printf("각 점수 출력: %d %d %d\n", s1.mic, s1.lin, s1.pro);
    return 0;
}
```

구조체와 포인터

structs and pointers

- C언어 typedef 키워드: 이미 존재하는 타입에 새로운 이름을 붙일 때 사용
- 구조체 변수를 선언하거나 사용할 때에는 매번 struct 키워드를 사용하여 구조체임을 명시.

```c
#include<stdio.h>

typedef struct person
{
    char name[20];
    int num;
    unsigned int age;
}Person;

void ShowData(Person* ptr){
    printf("이름: %s \n", (*ptr).name); //ptr->nmae
    printf("번호: %d \n", (*ptr).num);  //ptr->num
    printf("나이: %d \n", (*ptr).age);  //ptr->age
}

int main(){
    Person ps={"이순신", 10,24};
    Person * ptr = &ps;
    ShowData(ptr);
    return 0;
}
```

// 포인터 == 어떤 변수의 주소를 담아서 가리키는 변수. 구조체 포인터도 마찬가지. 구조체를 가리키는 포인터를 구조체 포인터라고 함. 구조체는 struct [구조체이름]이 자료형이나 마찬가지. 따라서 struct student *ptr;과 같이 선언.

구조체와 패딩

structs and padding

구조체 패딩은 성능 향상을 위해 CPU가 접근하기 쉬운 위치 메모리에 구조체 필드를 배치하는 것.

- 32bit : ll에 접근하기 위해 메모리에 3번 접근
- 64bit : ll에 접근하기 위해 메모리에 2번 접근

하지만 아래와 같이 구조체에 패딩 값을 넣어준다면

패딩 비트의 크기만큼 메모리를 낭비하게 되지만

CPU가 ll을 읽어올 때 메모리에 접근하는 수가 줄어들어 성능 저하를 막을 수 있다.

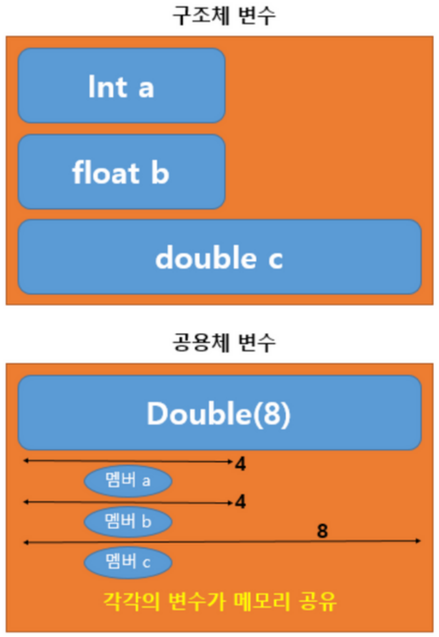
(CPU 연산 횟수 감소)

구조체와 공용체

structs and unions

공용체란?

- 공용체도 사용자가 정의한 자료형.
- 구조체와의 차이점은 메모리 공간을 공유한다는 점.



구조체 == 각 가정의 화정실,
공용체 == 공원의 공중 화장실

```c
#include <stdio.h>

typedef union num{
    char a;
    short b;
    int c;
}Num;

int main(void){
    Num n;
    n.c = 0x12345678;
    printf("%x\n", n.a);
    printf("%x\n", n.b);
    printf("%x\n", n.c);
    return 0;
}
```

프로그래밍 과제

programming challenge

다음을 프로그래밍 하시오

```
#include <stdio.h>

typedef struct student {
    int num;
    char name[20];
    int mic, lin, pro;
    int tot;
    double avg;
    char grade;
} Student;

void input_data(Student *pary);
void calc_data(Student *pary);
void print_data(Student *pary);

int main(void) {
    Student ary[5];
    input_data(ary);
    calc_data(ary);
    printf("결과값은?₩n");
    print_data(ary);
    return 0;
}
```

- Student ary[5]

num (4)	name (20)	mic (4)	lin (4)	pro (4)	tot (4)	avg (8)	grade (1)

- 조건1 : grade에서 90점 이상 A, 80점 이상 B, 70점 이상 C, 나머지 F 출력
- 조건2 : 입력이 num: 1, name: 이순신, mic: 90, lin: 80, pro:100 일 때 출력 형태는 아래와 같이 표시

출력값은?
1, 이순신, 90, 80, 100, 270, 90, A

```c
// 조건 1: grade에서 90점 이상 A, 80점 이상 B, 70점 이상 C, 나머지 F 출력
// 조건 2: 입력이 num: 1, name: 이순신, mic: 90, lin: 80, pro:100 일 때
// 출력-> 출력값은?
//       1, 이순신, 90, 80, 100, 270, 90, A
#include<stdio.h>
typedef struct student{
    int num;
    int mic, lin, pro;
    int tot;
    double avg;
    char grade;
}Student;

void input_data(Student *pary);
void calc_Data(Student *pary);
void print_data(Student *pray);
int main(){
    Student ary[2];
    input_data(ary);
    calc_Data(ary);
    printf("출력값은\n");
    print_data(ary);
}
void input_data(Student *pary){
    for(int i=0;i<2;i++){
        scanf("%d", &(pary+i)->num);
        scanf("%d", &(pary+i)->mic);
        scanf("%d", &(pary+i)->lin);
        scanf("%d", &(pary+i)->pro);
```