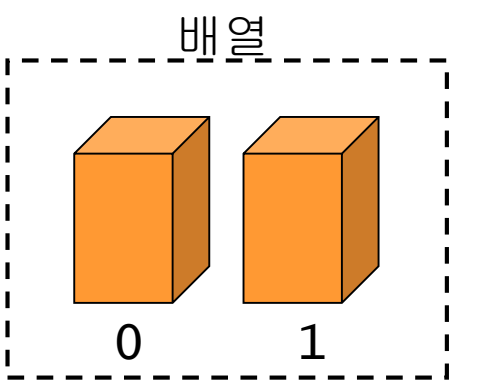
CSE2010 자료구조론

Week 2: Struct

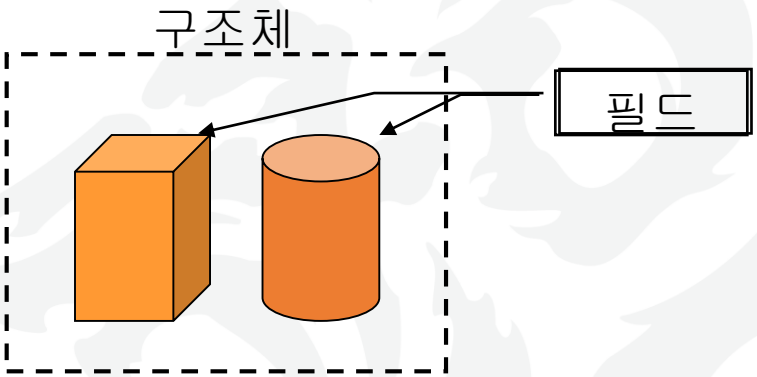
ICT융합학부 한진영

구조체

- 구조체(structure): 타입이 다른 데이터를 하나로 묶는 방법
  - cf. 배열(array): 타입이 같은 데이터들을 하나로 묶는 방법



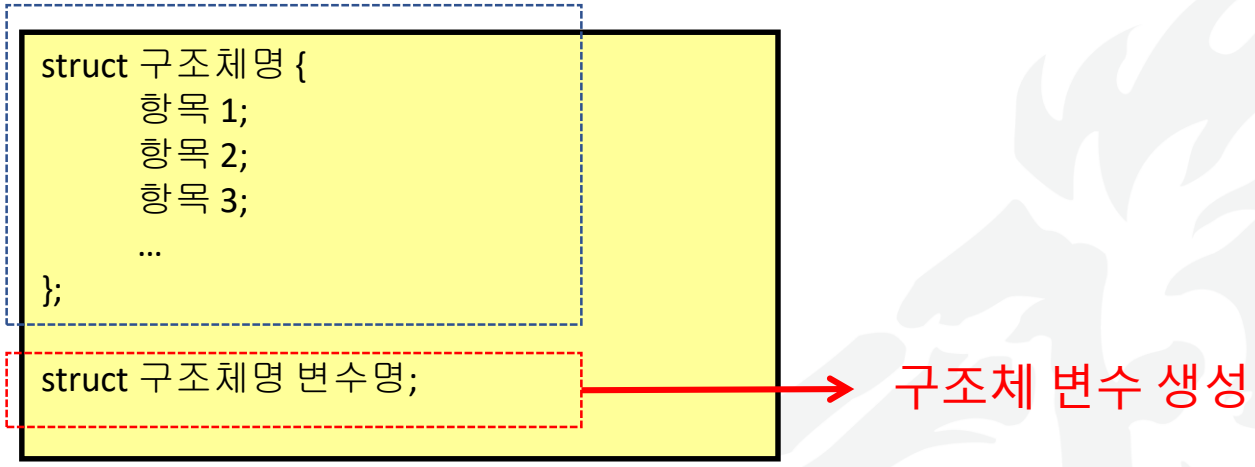

```
char carray[100];
```




```
struct example {
        char cfield;
        int ifield;
        float ffield;
        double dfield;
};
struct example s1;
```

구조체 형식

- 구조체 형식 정의



- typedef를 이용해 구조체를 만들면 기존 C의 자료형에는 없었던 새로운 타입으로 선언할 수 있음
  - 구조체 개념이 발전해 C++의 클래스(class)가 됨

구조체 사용 예

- 구조체의 선언과 구조체 변수의 생성

```
struct person {
        char name[10];      // 문자배열로 된 이름
        int age;            // 나이를 나타내는 정수값
        float height;  // 키를 나타내는 실수값
};
struct person a;            // 구조체 변수 선언
```

- typedef을 이용한 구조체의 선언과 구조체 변수의 생성

```
typedef struct person {
        char name[10];      // 문자배열로 된 이름
        int age;            // 나이를 나타내는 정수값
        float height;  // 키를 나타내는 실수값
} person;
person a;                   // 구조체 변수 선언
```

- 항목연산자(membership operator): "."

```
   strcpy(a.name, "tom");
   a.age = 20;
   a.height = 180.5;
```

구조체의 대입과 비교 연산

- 구조체 변수의 대입: 가능

```
struct person {
        char name[10];      // 문자배열로 된 이름
        int age;            // 나이를 나타내는 정수값
        float height;  // 키를 나타내는 실수값
};
main()
{
        person a, b;
        b = a;          // 가능
}
```

- 구조체 변수끼리의 비교: 불가능

```
main()
{
        if( a > b )
                printf("a가 b보다 나이가 많음");       // 불가능
}
```

자체 참조 구조체

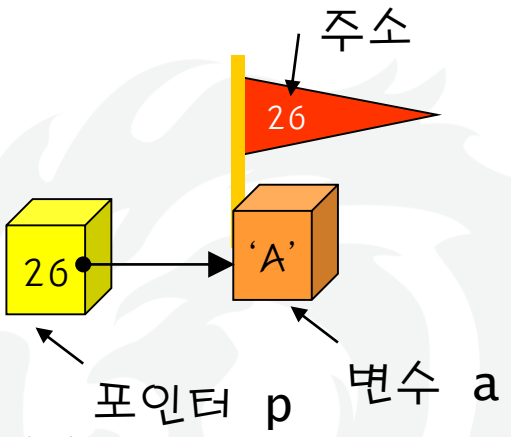
- 자체 참조 구조체(self-referential structure)
  - 항목 중에서 자기 자신을 가리키는 포인터가 한 개 이상 존재하는 구조체
  - 연결 리스트나 트리에 많이 등장

```
typedef struct ListNode {
        char    data[10];
        struct  ListNode *link;
} ListNode;
```

포인터

- 포인터 변수: 다른 변수의 주소를 가지고 있는 변수

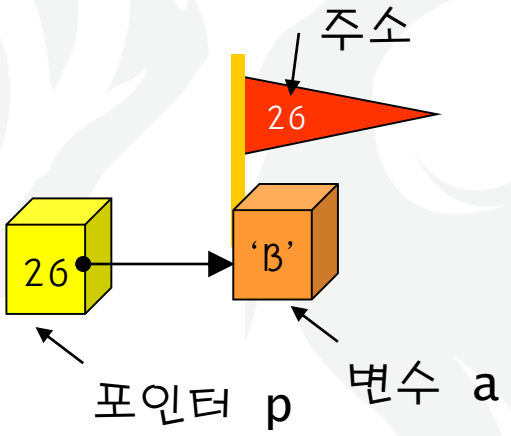
```
char a='A';
char *p;
p = &a;
```



- *p와 변수 a가 동일한 메모리 위치를 참조함
- *p와 변수 a는 동일한 객체를 가리키므로 전적으로 동일함
- *p값을 변경하면 변수 a값도 바뀜

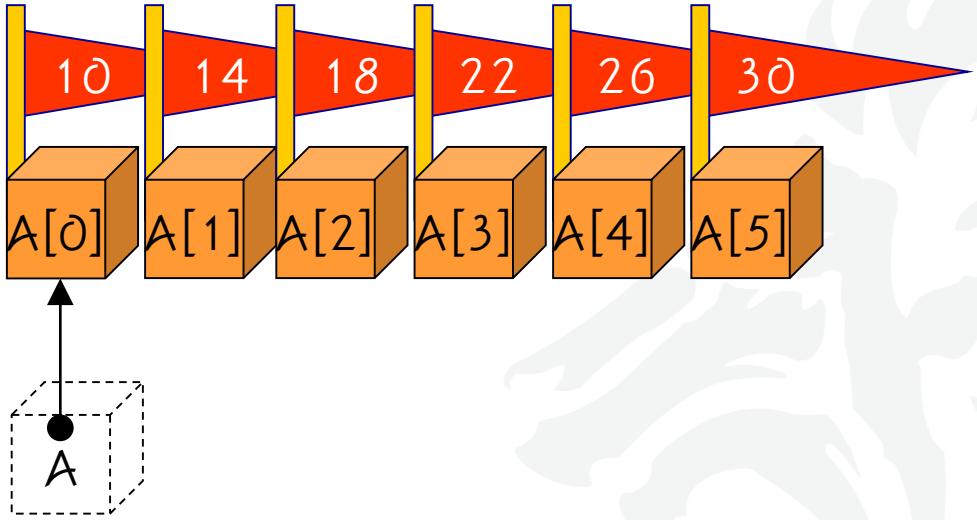
- 포인터가 가리키는 내용의 변경: * 연산자 사용

```
*p= 'B';
```



배열과 포인터

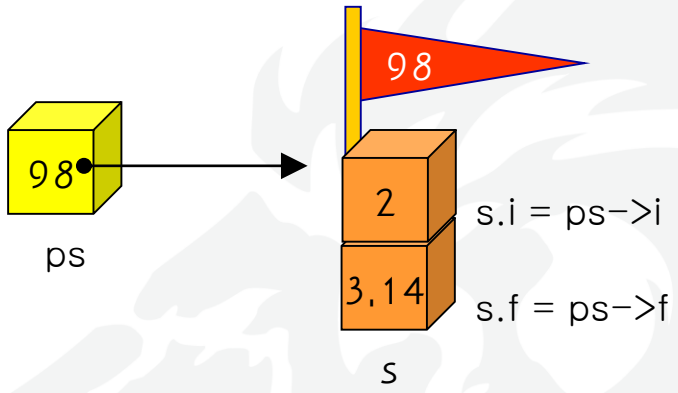
- 배열의 이름: 사실상의 포인터와 같은 역할



컴파일러가 배열의 이름을 배열의 첫 번째 주소로 대치

구조체와 포인터

- 구조체의 요소(멤버)에 접근하는 연산자: ->




```
main()
{
     struct {
          int i;
          float f;
     } s, *ps;
     ps = &s;
     ps->i = 2;
     ps->f = 3.14;
}
```

ps->i 는 (*ps).i와 동일한 효과를 가짐

포인터 사용시 주의점

- 포인터가 아무것도 가리키고 있지 않을 때는 NULL로 설정
  - int *pi=NULL;
- 초기화가 안된 상태에서 사용 금지

```
main()
{
        char *pc;     // 포인터 pi는 초기화가 안되어 있음
        *pc = 'E';    // 위험한 코드
}
```

- 포인터 타입간의 변환시에는 명시적인 타입 변환 사용

```
int *pi;
float *pf;
pf = (float *)pi;
```

메모리 할당

- 프로그램이 메모리를 할당받는 방법
  - 정적 메모리
  - 동적 메모리 할당
- 정적 메모리 할당
  - 메모리의 크기는 프로그램이 시작하기 전에 결정
  - 프로그램의 수행 도중에 그 크기가 변경될 수는 없음
  - 만약 처음에 결정된 크기보다 더 큰 입력이 들어온다면 처리하지 못할 것이고 더 작은 입력이 들어온다면 남은 메모리 공간은 낭비될 것임
  - 예: 변수나 배열의 선언
    - int buffer[100]; char name[] = "data structure";
- 동적 메모리 할당
  - 프로그램의 실행 도중에 메모리를 할당 받는 것
  - 필요한 만큼만 할당을 받고 또 필요한 때에 사용하고 반납
  - 메모리를 매우 효율적으로 사용가능

동적 메모리 할당

- 동적 메모리 할당(dynamic memory allocation)
  - 프로그램이 실행 도중에 동적으로 메모리를 할당 받는 것
- 전형적인 동적 메모리 할당 코드

```
main()
{
   int *pi;
   pi = (int *)malloc(sizeof(int));        // 동적 메모리 할당
   …
   …                                       // 동적 메모리 사용
   …
   free(pi);                               // 동적 메모리 반납
}
```

- 동적 메모리 할당 관련 함수
  - malloc(size): 메모리 할당
  - free(ptr): 메모리 해제
  - sizeof(var): 변수나 타입의 크기 반환(바이트단위)

동적 메모리 할당 관련 함수

- malloc(int size): size만큼 메모리 블록 할당

```
(char *)malloc(100) ;    /* 100 바이트 할당 */
(int *)malloc(sizeof(int));/* 정수 1개를 저장할 메모리 확보*/
(struct Book *)malloc(sizeof(struct Book))/* Book 구조체 메모리 할당 */
```

- free(ptr): 메모리 해제: ptr이 가리키는 할당된 메모리 블록을 해제
- sizeof(var): 변수나 타입의 크기 반환(바이트단위)

```
size_t  i = sizeof( int );                          // 4
struct AlignDepends {
    char c;
    int i;
};
size_t size = sizeof(struct AlignDepends);          // 5? 8?
int  array[] = { 1, 2, 3, 4, 5 };
size_t  sizearr = sizeof( array ) / sizeof( array[0] );  // 20/4=5
```

동적 메모리 할당 예제

```
struct Example {
        int number;
        char name[10];
};
void main()
{
        struct Example *p;

        p=(struct Example *)malloc(2*sizeof(struct Example));
        if(p==NULL){
          fprintf(stderr, "can't allocate memory\n") ;
          exit(1) ;
        }
        p->number=1;
        strcpy(p->name,"Park");
        (p+1)->number=2;
        strcpy((p+1)->name,"Kim");
        free(p);
}
```

Week 2: Struct