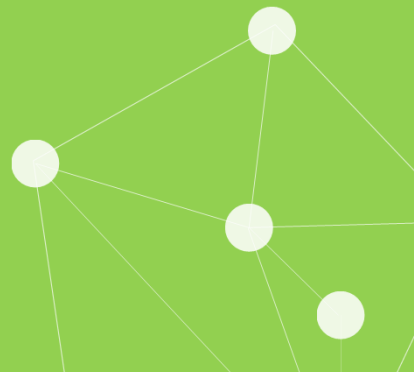


02 CHAPTER

배열과 구조체



많은 자료의 처리?



- 배열(array), 구조체(struct)
 - 성적 처리 프로그램에서 45명의 성적을 저장하는 방법
 - 주소록 프로그램에서 친구들의 다양한 정보(이름, 전화번호, 주소, 이메일 등)를 통합하여 저장하는 방법

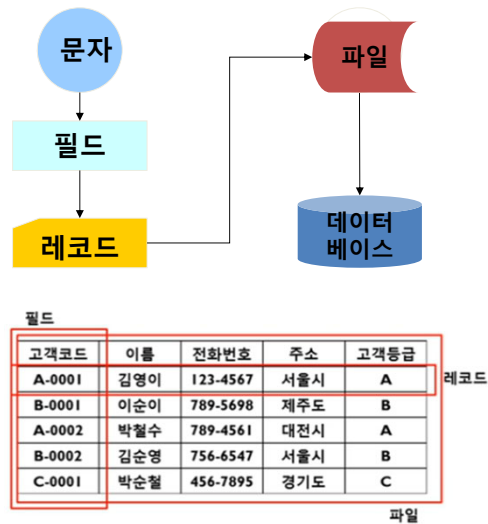
1번	2번	3번	...	45번
84	72	94		87

반 학생들의 성적 처리

홍길동
이름: 홍길동
전화: 010-1234-56XX
주소: 서울특별시 XX구 YY동...
이메일: kdhong@korea...

친구 주소록 만들기

정보의 표현



3

레코드(Record)의 정의

- 필드
 - 의미 있는 연속된 문자를 기억 시키는데 필요한 몇 개의 바이트들의 묶음
- 레코드
 - 서로 관계 있는 여러 필드를 한 개의 조로 묶어서 구성한 데이터 구조
 - 일반적으로 보조 기억 장치에서의 입출력 단위로 사용

4

배열과 레코드의 차이점

- 배열

- 데이터 원소의 크기와 데이터 형이 동일하다(Homogeneous)
- 첨자를 이용하여 원소를 참조(reference)
예) item[k]

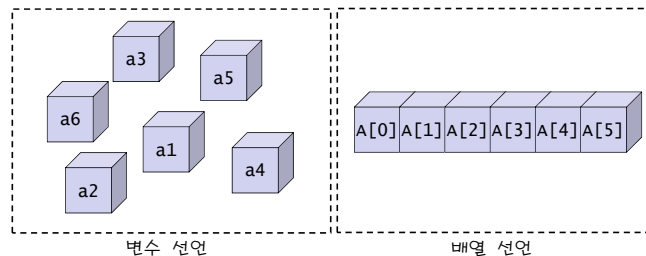
- 레코드

- 데이터 원소의 크기와 데이터 형이 서로 다르다(Heterogeneous)
- 필드를 이용하여 원소를 참조
예) Student.idNo

5

배열

- 같은 형의 변수를 여러 개 만드는 경우에 사용
 - 여러 개의 변수 선언: `int A0, A1, A2, A3, ..., A5;`
 - 하나의 배열 선언: `int A[6];`



- 만약 배열이 없다면?
 - 반복문을 사용할 수 없다!

6

배열의 특징

- 배열: <인덱스, 요소> 쌍의 집합
 - 인덱스가 주어지면 해당되는 요소가 대응되는 구조
 - 배열1차원 배열의 선언

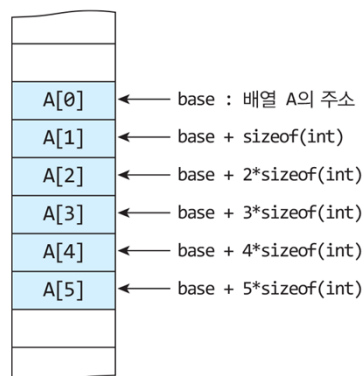
```
자료형 배열이름 [배열의 원소수];
```

- 직접 접근(direct access) 방식
 - 항목 접근의 시간 복잡도가 $O(1)$
 - 연결 리스트(5장)
 - 순차 접근(sequential access) 방식
 - 항목 접근의 시간 복잡도 $O(n)$

7

1차원 배열

- 자료형 배열이름[배열의_크기];
- `int A[6];`



8

배열의 복사

- 변수의 복사와 배열의 복사

```
#include <stdio.h>
void main()
{
    int A[5] = { 10, 20, 30 };
    int B[5], i, x = 2018, y = 0;
    y = x;
    for (i = 0; i < 5; i++)
        B[i] = A[i];
    printf("변수 복사 결과: x=%d, y=%d\n", x, y);
    printf("배열 복사 결과: \n");
    for (i = 0; i < 5; i++) {
        printf("A[%d] = %d\t", i, A[i]);
        printf("B[%d] = %d\n", i, B[i]);
    }
}
```

```
변수 복사 결과: x=2018, y=2018
배열 복사 결과:
A[0] = 10      B[0] = 10
A[1] = 20      B[1] = 20
A[2] = 30      B[2] = 30
A[3] = 0       B[3] = 0
A[4] = 0       B[4] = 0
```

9

문자열 : 특별한 1차원 배열

- `char s[12] = "game over";`

	s[0]	s[1]	s[2]	s[3]	s[4]	s[5]	s[6]	s[7]	s[8]	s[9]	s[10]	s[11]
s	'g'	'a'	'm'	'e'	' '	'o'	'v'	'e'	'r'	'\0'		

- 문자열 처리
 - 문자열의 복사나 비교를 위해 `=`나 `==` 또는 `<` 등의 연산자를 사용할 수 없다.
 - `strcmp()`, `strcpy()`, ...
 - `<string.h>` 포함

10

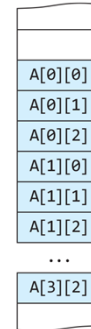
2차원 배열

- 자료형 배열이름[행의_크기][열의_크기];

자료형 배열이름 [행의 원소수][열의 원소수];

- `int A[4][3];`
- `int A[4][3] = { {1,2,3}, {4,5,6}, {7,8,9}, {10,11,12} };`

A[0][0]	A[0][1]	A[0][2]
A[1][0]	A[1][1]	A[1][2]
A[2][0]	A[2][1]	A[2][2]
A[3][0]	A[3][1]	A[3][2]



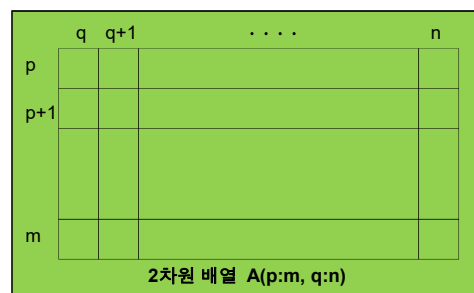
(a) 2차원 배열

(b) 실제 메모리 안에서의 위치

11

다차원 배열

- 2차원 배열 $A(p:m, q:n)$
 - $A(p:m, q:n) = \{ A(i, j) \mid i = p, \dots, m \text{ 과 } j = q, \dots, n \}$

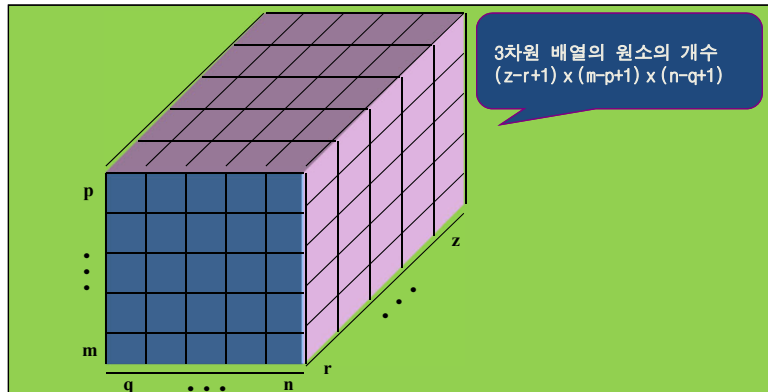


- 배열 A의 행의 개수 = $m - p + 1$, 열 A의 열의 개수 = $n - q + 1$
- 배열 A의 총 원소의 개수 = $(m - p + 1) \times (n - q + 1)$
- ▶ C 언어에서 2차원 배열의 기억 장소 배열 순서는 행우선 법칙을 사용

12

다차원 배열

- 3차원 배열 $B(r:z, p:m, q:n)$
 - $B(r:z, p:m, q:n) = \{ B(i, j, k) \mid i = r, \dots, z \text{ 과 } j = p, \dots, m \text{ 과 } k = q, \dots, n \}$



13

다차원 배열

- n 차원 배열 $C(L_1:U_1, L_2:U_2, \dots, L_n:U_n)$
 - $C(L_1:U_1, L_2:U_2, \dots, L_n:U_n)$
 $= \{ C(i_1, i_2, \dots, i_n) \mid k=1, 2, \dots, n, L_k < i_k < U_k \}$
- 배열 C 의 총 원소의 개수
 $= (U_1 - L_1 + 1) \times (U_2 - L_2 + 1) \times \dots \times (U_n - L_n + 1)$

14

다차원 배열

• 배열의 물리적 표현

- 기억장치에는 순차적으로 저장됨으로 인하여 2종류로 표현가능
- 배열 A(0:1, 0:2, 0:1)의 나열 순서

번지	열 방향 순서	행 방향 순서
101	A(0, 0, 0)	A(0, 0, 0)
102	A(0, 1, 0)	A(0, 0, 1)
103	A(0, 2, 0)	A(0, 1, 0)
104	A(0, 0, 1)	A(0, 1, 1)
105	A(0, 1, 1)	A(0, 2, 0)
106	A(0, 2, 1)	A(0, 2, 1)
107	A(1, 0, 0)	A(1, 0, 0)
108	A(1, 1, 0)	A(1, 0, 1)
109	A(1, 2, 0)	A(1, 1, 0)
110	A(1, 0, 1)	A(1, 1, 1)
111	A(1, 1, 1)	A(1, 2, 0)
112	A(1, 2, 1)	A(1, 2, 1)

첫
번째
째
면

A(0,0,0)	A(0,0,1)
A(0,1,0)	A(0,1,1)
A(0,2,0)	A(0,2,1)

두
번째
째
면

15

다차원 배열

• 배열의 물리적 표현 (계속)

- 열 방향 순서(column major order)
 - FORTRAN
 - 2차원 배열 A(0:U₁-1, 0:U₂-1)

$$A(i, j) \text{의 주소} = B + jU_1 + i$$
 - 3차원 배열 A(0:U₁-1, 0:U₂-1, 0:U₃-1)

$$A(i, j, k) \text{의 주소} = B + iU_2U_3 + kU_2 + j$$
- 행 방향 순서(row major order)
 - C, PASCAL, PL/1, COBOL
 - 2차원 배열 A(0:U₁-1, 0:U₂-1)

$$A(i, j) \text{의 주소} = B + iU_2 + j$$
 - 3차원 배열 A(0:U₁-1, 0:U₂-1, 0:U₃-1)

$$A(i, j, k) \text{의 주소} = B + iU_2U_3 + jU_3 + k$$

16

함수의 매개변수로서의 배열

- 변수의 전달 → 값을 복사 (call by value)
- 배열의 전달 → 첫 번째 항목의 주소를 전달(주소를 복사)

```
void copy_array(int a[], int b[], int len) {
    int i;
    for (i = 0; i < len; i++)
        b[i] = a[i];
}

void copy_variable(int a, int b) {
    b = a;
}

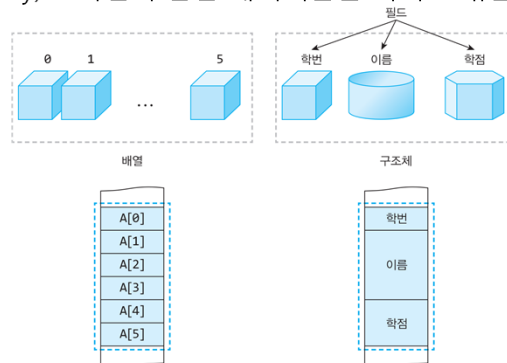
...
void main()
{
    int A[5] = { 10, 20, 30 };
    int B[5], i, x = 2018, y = 0;
    copy_variable(x, y);
    copy_array(A, B, 5);
}
```

C:\WINDOWS\system32\cmd.exe
 변수 복사 결과: x=2018, y=0
 배열 복사 결과:
 A[0] = 10 B[0] = 10
 A[1] = 20 B[1] = 20
 A[2] = 30 B[2] = 30
 A[3] = 0 B[3] = 0
 A[4] = 0 B[4] = 0
 계속하려면 아무 키나 누르십시오

17

구조체

- 기존의 자료형들을 조합해 새로운 자료형을 만드는 방법
- 배열과의 차이
 - 구조체(structure): 타입이 다른 데이터를 하나로 묶음
 - 배열(array): 타입이 같은 데이터들을 하나로 묶음



18

C 언어에서의 레코드



- 레코드의 선언
 - 구조체 데이터 형태를 선언

```
struct 구조체 선언명 {  
    member 1의 선언;  
    member 2의 선언;  
    ...  
    member n의 선언;  
}
```

19

C 언어에서의 레코드



- 레코드 필드의 참조
 - 참조 연산자 " ."를 사용

– 예)

```
person.number = 1;  
person.name = "Kim, SungSuk";  
person.address = "Seoul Sungbuk-Gu";  
person.tel = "123-3456";  
person.sex = 0;
```

20

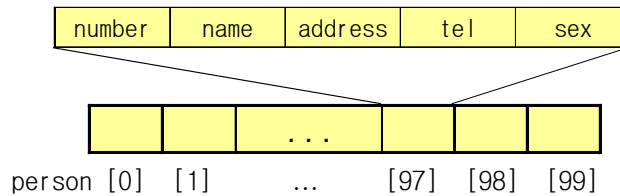
C 언어에서의 레코드

- 레코드 배열

- 배열의 각 원소가 레코드 원소로 구성된 경우

- 예)

```
struct Jusorok {  
    int number;  
    String name[20];  
    String address[100];  
    String tel[13];  
    int sex;  
}  
struct Jusorok person[100];
```



21

C 언어에서의 레코드

- 중첩된 레코드

- 레코드내 필드가 다른 레코드로 구성된 경우

- 예) 중첩된 레코드 alphaNumeric

```
struct r_char {  
    char cmem1;  
    char cmem2;  
};
```

```
struct r_number {  
    int nmem1;  
    int nmem2;  
};
```

```
struct nested_struct {  
    int mem0;  
    struct r_char c_field;  
    struct r_number n_field(2);  
} alpha_numeric(2);
```

22

C 언어에서의 레코드

- alphaNumeric 구조
 - alphaNumeric[0]
 - mem0(1)
 - cField(2)
 - cMem1
 - cMem2
 - nField[0]
 - nMem1
 - nMem2
 - nField[1]
 - nMem1
 - nMem2
 - alphaNumeric[1]
 - mem0
 - cField
 - cMem1
 - cMem2
 - nField[0]
 - nMem1 (3)
 - nMem2
 - nField[1]
 - nMem1
 - nMem2(4)
- 필드 참조 예
 - (1) alphaNumeric[0].mem0
 - (2) alphaNumeric[0].cField.cMem1
 - (3) alphaNumeric[1].nField[0].nMem1
 - (4) alphaNumeric[1].nField[1].nMem2

23

구조체의 정의와 선언

- 정의

<pre>struct Student { int id; char name[20]; double score; };</pre>	<pre>typedef struct Student_t { int id; char name[20]; double score; } Student;</pre>
---	---
- 선언

<pre>struct Student a;</pre>	<pre>Student a;</pre>
------------------------------	-----------------------

```
Student a = { 201803156, "홍길동", 96.3 };
```
- 멤버 접근: 항목 연산자(membership operator) '.'


```
a.id = 30830;
a.score = 92.3;
strcpy(a.name, "Jinyoung");
// a.name = "Jinyoung";은 오류 발생
```

24

구조체와 연산자

- 대입 연산자만 가능

```
int x, y=10;
Student a, b={ 201803156, "홍길동", 96.3 };
x = y;           // OK: int 변수의 복사
a = b;           // OK: 구조체 변수의 복사
```

- 다른 연산자 사용 불가

```
if( a > b )       // 오류: 구조체의 비교연산 불가
    a += b;       // 오류: 구조체의 다른 대입 연산도 불가

int compare(Student a, Student b) {
    return a.id - b.id;
}
```

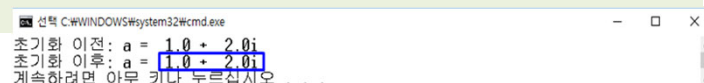
25

구조체와 함수

- 함수의 매개 변수나 반환형으로 사용할 수 있음.
 - Call by value
- 다음 함수의 동작은?

```
void print_complex(Complex c) {
    printf("%4.1f + %4.1fi\n", c.real, c.imag);
}

void reset_complex(Complex c) {
    c.real = c.imag = 0.0;
}
```



```
선택 C:\WINDOWS\system32\cmd.exe
초기화 이전: a = 1.0 + 2.0i
초기화 이후: a = 1.0 + 2.0i
계속하려면 아무 키나 누르십시오 . . .
```

26

배열과 구조체의 응용 : 다항식

- 다항식의 일반적인 형태

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$$

- 처리를 위한 다항식의 자료구조가 필요
- 어떤 자료구조가 다항식의 연산을 편리하게 할까?

- 다항식을 위한 자료구조?

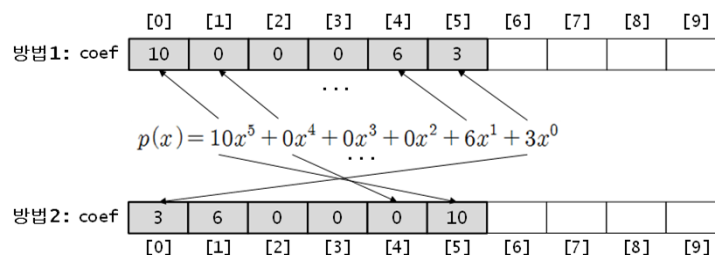
- 배열을 사용하는 방법

- 모든 항의 계수를 배열에 저장
- 다항식의 0이 아닌 항만을 배열에 저장: **희소 다항식**

27

다항식 구조체

```
#define MAX_DEGREE    101    // 다항식의 최고 차수 + 1
typedef struct {
    int degree;
    float coef[MAX_DEGREE];
} Polynomial ;
```



- 교제 p62-64 프로그램 참조

28

다항식 프로그램

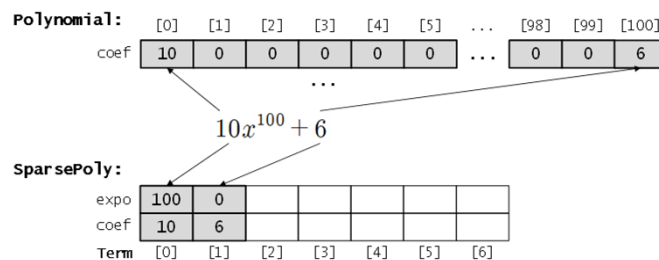
```
void main()
{
    Polynomial a, b, c;
    a = read_poly();
    b = read_poly();
    c = add_poly(a, b);
    print_poly(a, "A = ");
    print_poly(b, "B = ");
    print_poly(c, "A+B = ");
}
```

```
C:\WINDOWS\system32\cmd.exe
다항식의 최고 차수를 입력하시오: 3
각 항의 계수를 입력하시오 (총 4개): 1 2 3 4
다항식의 최고 차수를 입력하시오: 5
각 항의 계수를 입력하시오 (총 6개): 1 2 3 4 5 6
A = 1.0 x^3 + 2.0 x^2 + 3.0 x^1 + 4.0
B = 1.0 x^5 + 2.0 x^4 + 3.0 x^3 + 4.0 x^2 + 5.0 x^1 + 6.0
A+B= 1.0 x^5 + 2.0 x^4 + 4.0 x^3 + 6.0 x^2 + 8.0 x^1 + 10.0
계속하려면 아무 키나 누르십시오 . . .
```

29

희소 다항식의 표현

- 희소 다항식(Sparse Polynomial) 이란?
 - 대부분 항의 계수가 0인 다항식



```
typedef struct {
    int      nTerms;
    Term     term[MAX_TERMS];
} SparsePoly;
```

```
typedef struct {
    int      expon;
    float     coef;
} Term;
```

30

특별한 행렬

- 희소 행렬(sparse matrix)

- 행렬의 많은 원소들이 0으로 구성되어 있는 행렬

	열0	열1	열2	열3	열4	열5	열6
행0	20	0	0	0	0	5	-1
행1	0	0	28	0	0	0	0
행2	0	0	-6	0	11	0	0
행3	0	0	0	3	0	0	0
행4	9	0	0	0	0	0	0
행5	0	0	7	0	0	0	-8
행6	0	-3	0	0	0	0	0



7(행)	7(열)	11(값)
0	0	20
0	5	5
..
6	1	-3

31

특별한 행렬

- 희소 행렬(sparse matrix)

- 기억 장소의 낭비를 줄이기 위한 표현 방법
 - 0 인 아닌 원소들에 대하여 (행, 열, 값) 의 형태로 표현.

행\열	0	1	2
0	7	7	11
1	0	0	20
2	0	5	5
3	0	6	-1
4	1	2	28
5	2	2	-6
6	2	4	11
7	3	3	3
8	4	0	9
9	5	2	7
10	5	6	-8
11	6	1	-3

32

특별한 행렬

- 전치 행렬(transpose matrix)

- 행렬에서 (i, j) 에 위치한 원소를 (j, i) 로 옮기는 행렬

행\열	0	1	2
0	7	7	11
1	0	0	20
2	0	4	9
3	1	6	-3
4	2	1	28
5	2	2	-6
6	2	5	7
7	3	3	3
8	4	2	11
9	5	0	5
10	6	0	-1
11	6	5	-8

→

행\열	열0	열1	열2	열3	열4	열5	열6
행0	20	0	0	0	9	0	0
행1	0	0	0	0	0	0	-3
행2	0	28	-6	0	0	7	0
행3	0	0	0	3	0	0	0
행4	0	0	11	0	0	0	0
행5	5	0	0	0	0	0	0
행6	-1	0	0	0	0	-8	0

2장 정리

- 배열

- 1차원 -> 다차원
- 행우선(row major order) - C
- 열우선(column major order)
- 선언 및 활용

- 구조체란?

- 배열과의 차이
- 레코드(cf. 구조체)

- C언어 - call by value, call by reference 구분

- 함수를 사용한 다항식 프로그램
- 희소다항식과 희소행렬