3.1 배열

배열의 개념 배열 거의 모든 프로그래밍 언어에서 기본적으로제공되는 자료형

동일한 타입의 데이터를 한번에 여러 개 만들 때 사용

연속적인 메모리 공간 할당 🡪 인덱스 번호를 사용해 쉽게 접근 가능

배열ADT 객체: <인덱스, 값> 쌍의 집합

연산:

create(size) ::= size개의 요소를 저장할 수 있는 배열 생성

get(A, i) ::= 배열 A의 i번째 요소 반환

set(A, i, v) ::= 배열 A의 i번째 위치에 값 v 저장

C언어에서의 1차원 배열 배열선언 형식 자료형 배열이름[크기]

ex) int list[6]; //create

요소저장 형식 배열이름[인덱스] = 값

ex) list[0] = 100; //set

요소가져오기 형식 변수이름 = 배열이름[인덱스]

ex) int value = list[0]; //get

메모리주소 배열의 요소를 메모리의 연속된 위치에 할당

[0], 첫번째 요소 기본주소 = base

[1], 두번째 요소 base+1\*sizeof(요소자료형의 크기)

[2], 세번째 요소 base+2\*sizeof(요소자료형의 크기)

[3], 네번째 요소 base+3\*sizeof(요소자료형의 크기)

…

2차원 배열 요소들이 2차원형태로 나열된 배열

가로줄 행

세로줄 열

형식 자료형 리스트변수[행][열];

ex) int list[3][5];

Quiz 1. int list[6]; //create

list[0] = 100; //set

int value = list[0]; //get

2. base+5\*sizeof(int)

3.2 구조체

구조체의 개념 타입이 다른 데이터를 묶는 방법

struct 키워드 사용

구조체 태그 형식 struct 구조체이름{

항목1;

항목2;

…

};

ex) struct studentTag {

char name[10];

int age;

double gpa;

};

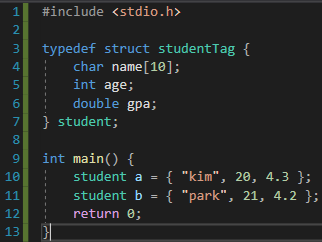
구조체 변수 선언 형식 struct 구조체이름 구조체변수

ex) struct studentTag s;

멤버연산자 구조체의 멤버사용

형식 구조체변수.항목

ex) s.age = 10;

프로그램3.1 

Quiz 1. typedef struct Point {

double x;

double y;

}point;

2. point p1, p2;

3. p1.x = 1;

p1.y = 2;

p2.x = 9;

p2.y = 8;

4. double get\_distance(point p1, point p2) {

return sqrt((p1.x - p2.x) \* (p1.x - p2.x) + (p1.y - p2.y) \* (p1.y - p2.y));

}

3.3 배열의 응용: 구조체

다항식의 차수 가장 큰 차수

다항식 표현 자료구조

첫 번째 방법 모든 차수의 계수값을 배열에 저장

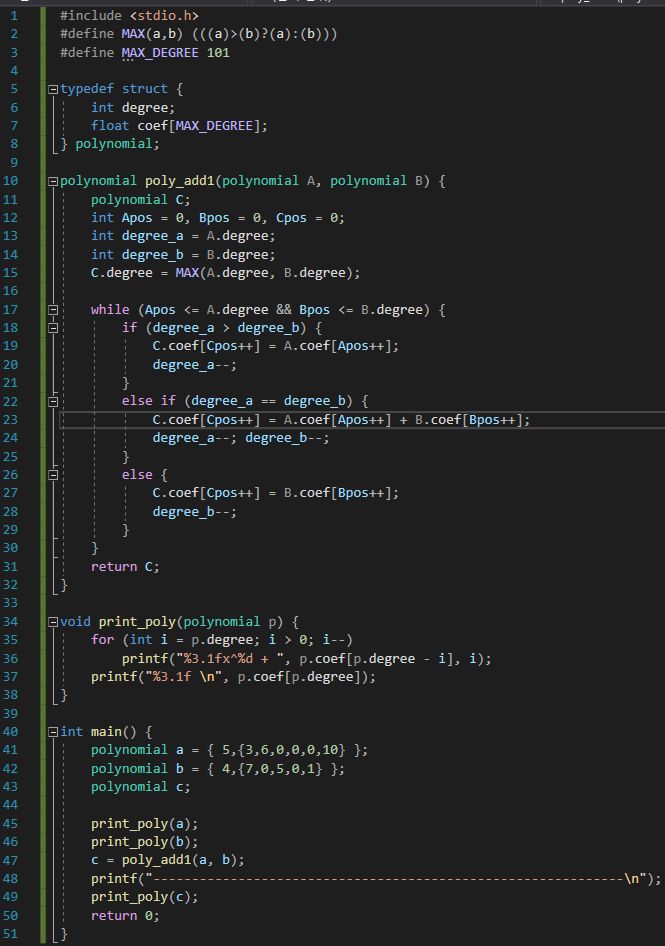
구조체에 다항식을 저장

구조체 coef 모든 차수의 계수값을 배열에 저장

degree 다항식의 차수

공간의 낭비가 매우 심함

간단한 알고리즘

프로그램 3.2 

두 번째 방법 다항식에서 0이 아닌 항을 구조체 전역 배열에 저장

0이 아닌 항들을 (계수, 차수)형식으로 표시

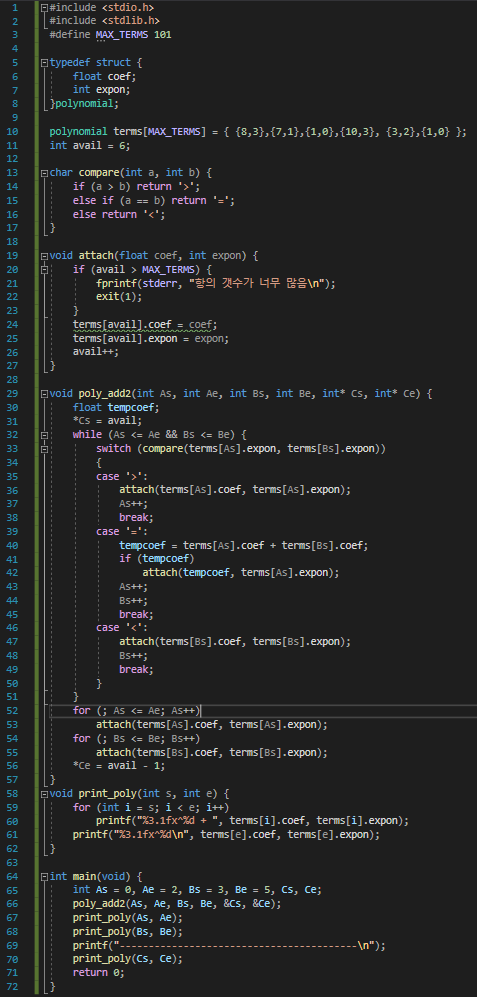
구조체 계수, 차수 쌍

구조체 배열 생성

단점 하나의 다항식이 시작되고 끝나는 위치를 가리키는 인덱스 변수 관리

연산의 구현이 어려움

더하기 알고리즘 각 항의 차수를 비교해 차수가 같으면 각 항의 계수를 더하여 결과로 옮김∙차수가 다르면 차수가 큰항을 결과로 옮김

프로그램 3.3 

Quiz 1. polynomial a = { 3,{6,8,0,9} };

2. polynomial terms[3] = { {6,3}, {8,2}, {9,0} };

3. 첫번째 방법

polynomial poly\_add1(polynomial A, polynomial B) {

polynomial C;

int Apos = 0, Bpos = 0, Cpos = 0;

int degree\_a = A.degree;

int degree\_b = B.degree;

C.degree = MAX(A.degree, B.degree);

while (Apos <= A.degree && Bpos <= B.degree) {

if (degree\_a > degree\_b) {

C.coef[Cpos++] = A.coef[Apos++];

degree\_a--;

}

else if (degree\_a == degree\_b) {

C.coef[Cpos++] = A.coef[Apos++] + B.coef[Bpos++]; //C.coef[Cpos++] = A.coef[Apos++] - B.coef[Bpos++];

degree\_a--; degree\_b--;

}

else {

C.coef[Cpos++] = B.coef[Bpos++];

degree\_b--;

}

}

return C;

}

두번째방법

void poly\_add2(int As, int Ae, int Bs, int Be, int\* Cs, int\* Ce) {

float tempcoef;

\*Cs = avail;

while (As <= Ae && Bs <= Be) {

switch (compare(terms[As].expon, terms[Bs].expon))

{

case '>':

attach(terms[As].coef, terms[As].expon);

As++;

break;

case '=':

tempcoef = terms[As].coef + terms[Bs].coef; //tempcoef = terms[As].coef - terms[Bs].coef;

if (tempcoef)

attach(tempcoef, terms[As].expon);

As++;

Bs++;

break;

case '<':

attach(terms[Bs].coef, terms[Bs].expon); //attach((-1)\*terms[Bs].coef, terms[Bs].expon);

Bs++;

break;

}

}

for (; As <= Ae; As++)

attach(terms[As].coef, terms[As].expon);

for (; Bs <= Be; Bs++)

attach(terms[Bs].coef, terms[Bs].expon);

\*Ce = avail - 1;

}

4. void poly\_read(polynomial \*a) {

FILE\* fp;

fp = fopen("text.txt", "rb");

fread(a, sizeof(a), 1, fp);

}

5. 첫번째 방법

polynomial poly\_eval(polynomial a, polynomial b, int e) {

polynomial r = { 0,{0} };

if (a.degree > e && b.degree > e) fprintf(stderr, "계산불가");

else if (a.degree<e && b.degree > e) {

r.degree = e;

r.coef[0] = a.coef[e];

}

else if (b.degree<e && a.degree > e) {

r.degree = e;

r.coef[0] = b.coef[e];

}

else {

r.degree = e;

r.coef[0] =a.coef[e]+b.coef[e];

}

return r;

}

두번째 방법

polynomial poly\_eval(int As, int Ae, int Bs, int Be, int e) {

polynomial r = { 0,e };

for (; As <= Ae; As++) {

if (terms[As].expon == e) r.coef += terms[As].coef;

}

for (; Bs <= Be; Bs++) {

if (terms[Bs].expon == e) r.coef += terms[Bs].coef;

}

return r;

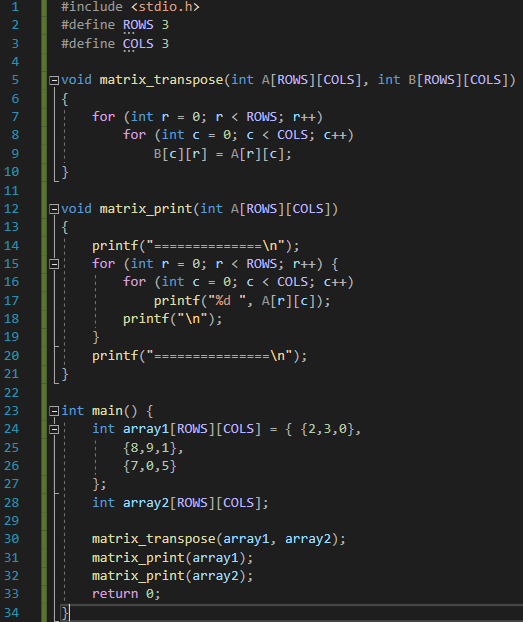
}

3.4 배열의 응용:희소행렬

행렬 표현 1. 2차원배열

2. 0이 아닌 노드만을 (행, 열, 값)으로 표시

전치행렬 계산하기1 방법1로 구현

프로그램3.4 

전치행렬 계산하기2 방법2로 구현

하나의 요소 (row, col, value)로 표현

구조체 element로 정의

알고리즘 3.1 전치행렬

새로운 구조체 b를 생성한다.

구조체 a에 저장된 모든 요소에 대하여 다음을 반복

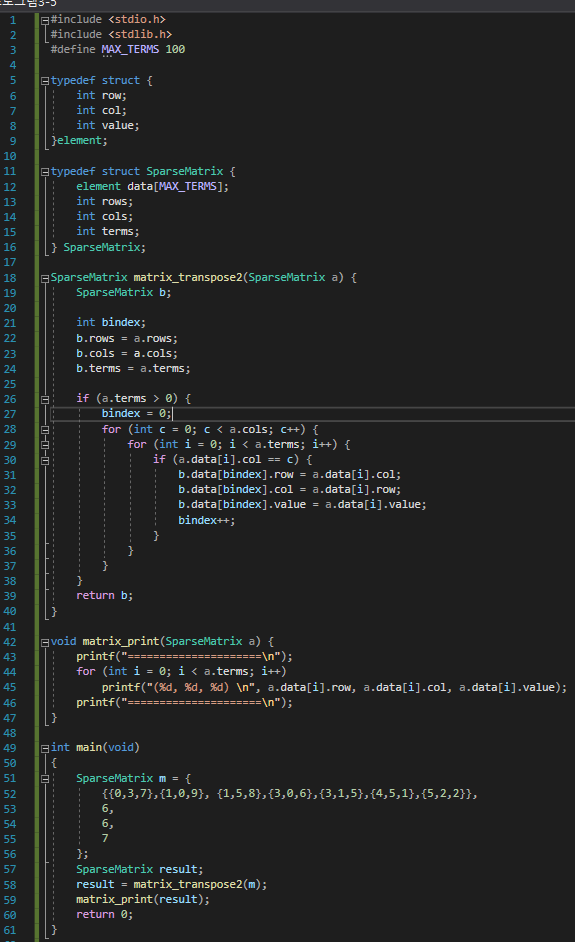
{

b.data[bindex].row = a.data[i].col;

b.data[bindex].col = a.data[i].row;

b.data[bindex].value = a.data[i].value;

}

프로그램3.5 

Quiz 1. void add\_matrix(int A[ROWS][COLS], int B[ROWS][COLS], int C[ROWS][COLS])

{

for (int r = 0; r < ROWS; r++)

for (int c = 0; c < COLS; c++)

C[r][c] = B[r][c] + A[r][c];

}

3.5 포인터

포인터의 개념 포인터 다른 변수의 주소를 가지고 있는 변수

ex) int a = 10;

int\* p = &a;

포인터와 관련된 연산자 &연산자 주소 연산자

변수의 주소를 추출하는 연산자

\*연산자 간접참조 연산자

포인터가 가리키는 곳에 값을 할당하는 연산자

다양한 포인터 여러가지 자료형에 포인터 사용가능

ex) int\* ip;

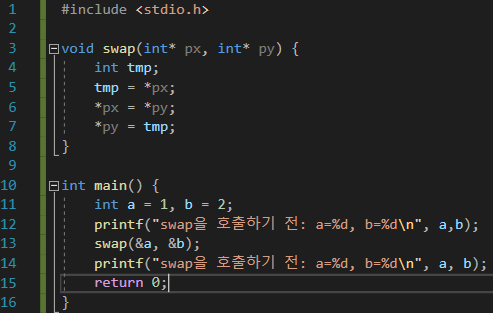
float\* fp;

char\* cp;

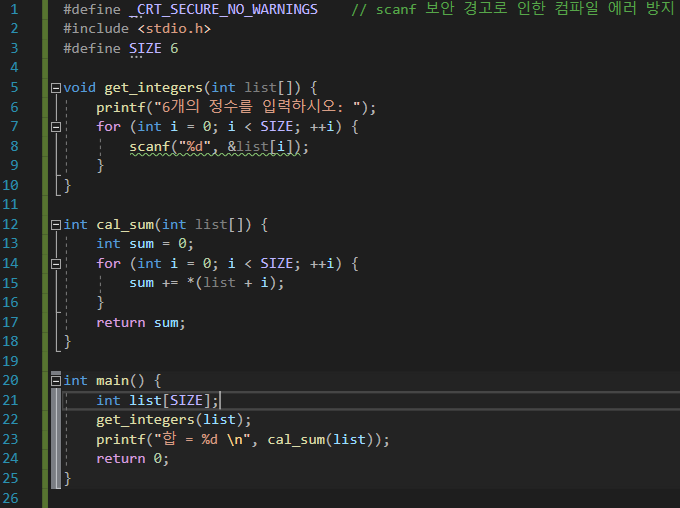
널 포인터 어떤 객체도 가리키지 않는 포인터

포인터가 아무 것도 가리키지 않을 경우에는 널 포인터 상태로 만들어두는 것이 좋음 🡪 오류 예방

함수 매개변수로 포인터 사용하기 함수 안에서 외부 변수의 값 변경

프로그램3.6 

배열과 포인터 배열의 이름 = 배열의 시작위치를 가리키는 포인터 = 배열의 첫 번째 원소를 가리키는 포인터

프로그램3.7 

Quiz 1. 첫 번째 함수에서는 매개변수로 받은 구조체 변수 변경 불가, 두 번쨰 함수에서는 매개변수로 받은 구조체 변수 변경 가능

2. 첫 번째 함수에선 포인터를 매개변수로 받음, 두 번째 함수에선 인수의 이중포인터를 매개변수로 받음

2번째

3.6 동적 메모리 할당

배열의 고정된 크기 문제 발생

더 큰 입력 처리 불가

더 작은 입력 메모리 낭비

동적 메모리 할당 필요한 만큼에 메모리를 운영체제로부터 받아 할당, 사용이 끝나면 반환 🡪 효율적인 메모리 사용 가능

ex) int\* p;

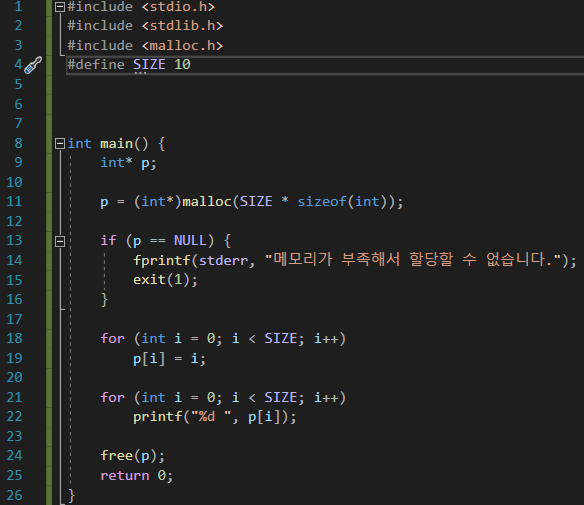
p = (int\*)malloc(sizeof(int));

\*p = 100;

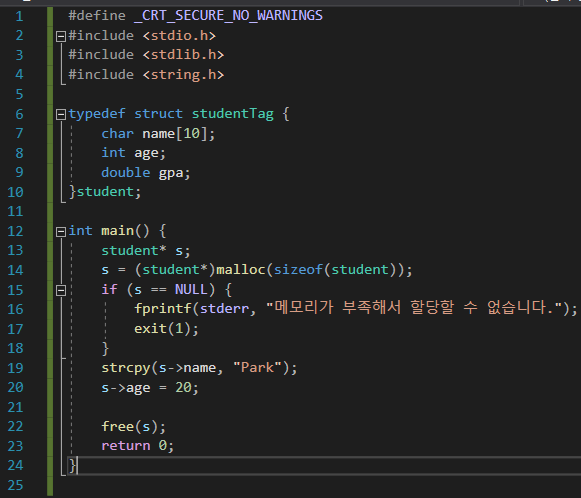
free(p);

히프 동적메모리가 할당되는 공간

운영체제가 사용되지 않는 메모리 공간을 모아 둔 것

프로그램3.8 

구조체와 포인터 ->연산자 구조체 포인터를 통해 구조체멤버 접근

프로그램3.9 

Quiz 1. double\* p1;

p1 = (int\*)malloc(double);

p1=23.92

🡪

double\* p1;

p1 = (double\*)malloc(sizeof(double));

\*p1 = 23.92;

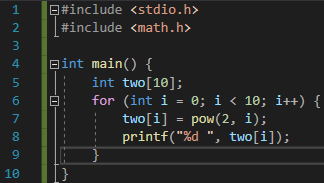
free(p1);

연습문제

1. 4

2. 4

3. 2

4. 

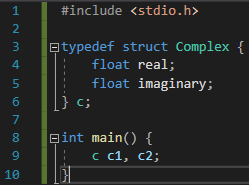
5. typedef struct person {

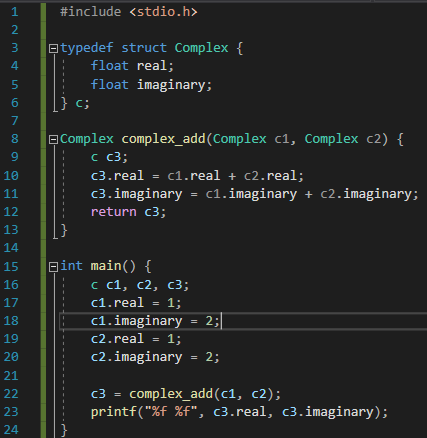
char name[10];

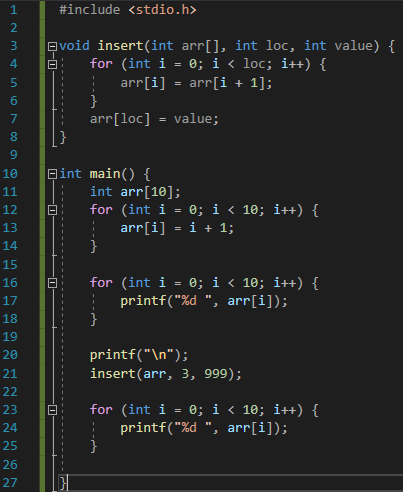
int age;

float sal;

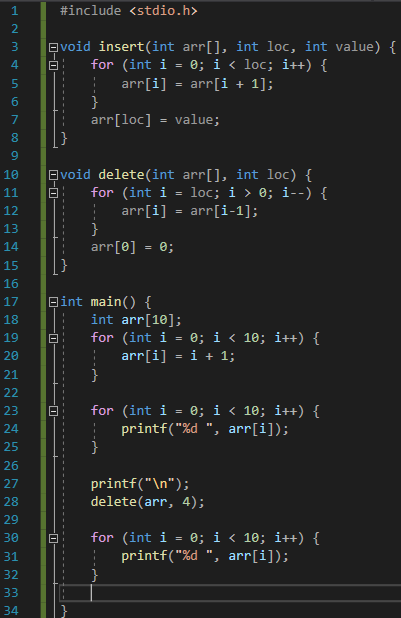
};

6. 

7. 

8. 

9. O(n)

10. 

11. O(n)

12. 