

# 자료구조 및 알고리즘 3주차 레포트

임베디드시스템공학과

2015146003

김기덕

## 1. 배열 값의 위치를 90도 이동 시키는 코드

### 1) 코드설명

```
1  #define _CRT_SECURE_NO_WARNINGS
2  #include <stdio.h>
3  #include <Windows.h>
4
5  int main( )
6  {
7      system("title Kimkideok");
8
9      int A[5][5];
10     int x, y, k = 0;
11     for (y = 0; y < 5; y++)
12     {
13         for (x = 0; x < 5; x++)
14         {
15             k = k + 2;
16             A[x][y] = k;
17             printf("%3d ", A[x][y]);
18         }
19         printf("\n");
20     }
21
22     int B[5][5];
23     int rotate, h;
24     for (rotate = 1; rotate < 3; rotate++)
25     {
26         for (y = 0; y < 5; y++)
27         {
28             for (x = 0; x < 5; x++)
29             {
30                 h = 4 - x;
31                 if (rotate == 1)
32                     B[y][x] = A[h][y];
33                 else
34                     A[y][x] = B[h][y];
35             }
36         }
37     }
38     printf("===== 90도 1번째 =====\n");
39     for (y = 0; y < 5; y++)
40     {
41         for (x = 0; x < 5; x++)
42         {
43             printf("%3d ", B[y][x]);
44         }
45         printf("\n");
46     }
47     printf("===== 90도 2번째 =====\n");
48     for (y = 0; y < 5; y++)
49     {
50         for (x = 0; x < 5; x++)
51         {
52             printf("%3d ", A[y][x]);
53         }
54         printf("\n");
55     }
56
57     return 0;
58 }
```

강의자료에 있는 코드

배열A와 변수 선언.

for문을 이용하여  
배열의 초기 값을 설정해준다.

배열B와 변수 선언.

rotate변수로 몇 번 회전시킬 것 인지 정할 수 있  
다. 2번으로 정했다.

배열을 90도 회전시키는 수식

처음 회전할 때는 행렬 B에 회전 값을 설정.

두 번째 회전할 때는 행렬 A에 회전 값을 설정.

처음 회전한 배열 출력

두 번째 회전한 배열 출력

## 2) 실행화면

```

2   4   6   8  10
12  14  16  18  20
22  24  26  28  30
32  34  36  38  40
42  44  46  48  50
===== 90도 1번째=====
10   8   6   4   2
20  18  16  14  12
30  28  26  24  22
40  38  36  34  32
50  48  46  44  42
===== 90도 2번째=====
50  40  30  20  10
48  38  28  18   8
46  36  26  16   6
44  34  24  14   4
42  32  22  12   2
계속하려면 아무 키나 누르
  
```

초기 배열

처음 회전시킨 배열

두 번째 회전시킨 배열

## 3) 분석

두 번째에는 90도 회전이 잘 이뤄졌지만 첫 번째에는 회전이 잘 이뤄지지 않았다.

회전을 시켜주는 수식에는 문제가 없었다.

하지만 초기 배열을 설정하는 과정에서 x먼저 설정을 해주는데 실질적으로 들어 가는건 y에 해당하는 값이 들어가게 된다. 출력은 들어가는데로 나오기 때문에 문제가 없어 보이지만 실질적으로 배열에 들어가 있는 값은 좌우가 반대로 된 꼬인 값이 들어가게 된다.

따라서 다음과 같이 배열설정의 x, y순서를 바꿔보았다.

```

for (y = 0; y < 5; y++)
{
    for (x = 0; x < 5; x++)
    {
        k = k + 2;
        A[y][x] = k;
        printf("%3d ", A[y][x]);
    }
    printf("\n");
}
  
```

```

2   4   6   8  10
12  14  16  18  20
22  24  26  28  30
32  34  36  38  40
42  44  46  48  50
===== 90도 1번째=====
42  32  22  12   2
44  34  24  14   4
46  36  26  16   6
48  38  28  18   8
50  40  30  20  10
===== 90도 2번째=====
50  48  46  44  42
40  38  36  34  32
30  28  26  24  22
20  18  16  14  12
10   8   6   4   2
계속하려면 아무 키나 누르십
  
```

초기 배열 값, 처음 회전시킨 값, 두 번째 회전시킨 값 모두 잘 나오는 것을 확인할 수 있다.

## 2. 2차원 배열의 논리적·물리적 순서 확인하기 프로그램

### 1) 코드설명

```
1  #define _CRT_SECURE_NO_WARNINGS
2  #include <stdio.h>
3  #include <Windows.h>
4
5  void main()
6  {
7      system("title Kimkideok");
8
9      int i, n = 0, *ptr;
10     int sale[2][4] = { { 63, 84, 140, 130 },
11                       { 157, 209, 251, 312 } }; // 2차원 배열의 초기화
12     ptr = &sale[0][0];
13     for (i = 0; i < 8; i++)
14     {
15         printf("\n addresss : %u sale%d= %d", ptr, i, *ptr);
16         ptr++;
17     }
18     getch();
19 }
```

변수와 포인터 선언  
2x4의 이차원 배열 선언 및  
초기화  
ptr포인터의 주소를  
sale배열의 첫 번째 주소로  
설정.  
ptr포인터의 주소를 증가시  
키며 그 값을 보여준다.

### 2) 실행화면

```
Kimkideok
addresss : 10222136 sale0= 63
addresss : 10222140 sale1= 84
addresss : 10222144 sale2= 140
addresss : 10222148 sale3= 130
addresss : 10222152 sale4= 157
addresss : 10222156 sale5= 209
addresss : 10222160 sale6= 251
addresss : 10222164 sale7= 312
```

### 3) 분석

이 코드는 2차원 배열에서 행과 열 중 어디를 우선적으로 저장하는지를 볼 수 있는 코드이다. 배열의 첫 번째 주소부터 포인터(시작주소 : 10222136)에 저장을 했는데 그 값을 순서대로 확인을 해 보았을 때 C컴파일러는 행 우선 순서 방법으로 2차원 배열을 저장한다는 것을 알 수 있다.

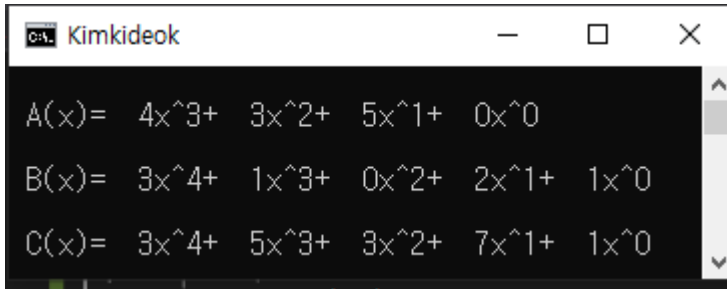
### 3. 다항식의 덧셈 알고리즘 : $A(x)+B(x) = C(x)$

#### 1) 코드설명

```
1  #define _CRT_SECURE_NO_WARNINGS
2  #include <stdio.h>
3  #include <Windows.h>
4  #define MAX(a, b) ((a>b) ? a : b)
5  #define MAX_DEGREE 50
6
7  typedef struct {
8      int degree;
9      float coef[MAX_DEGREE];
10 } polynomial;
11
12 polynomial addPoly(polynomial A, polynomial B)
13 {
14     polynomial C;
15     int A_index = 0, B_index = 0, C_index = 0;
16     int A_degree = A.degree, B_degree = B.degree;
17     C.degree = MAX(A.degree, B.degree);
18     while (A_index <= A.degree && B_index <= B.degree) {
19         if (A_degree > B_degree) {
20             C.coef[C_index++] = A.coef[A_index++];
21             A_degree--;
22         }
23         else if (A_degree == B_degree) {
24             C.coef[C_index++] = A.coef[A_index++] + B.coef[B_index++];
25             A_degree--;
26             B_degree--;
27         }
28         else {
29             C.coef[C_index++] = B.coef[B_index++];
30             B_degree--;
31         }
32     }
33     return C;
34 }
35
36 void printPoly(polynomial P)
37 {
38     int i, degree;
39     degree = P.degree;
40     for (i = 0; i <= P.degree; i++) {
41         printf("%3.0fx^%d", P.coef[i], degree--);
42         if (i < P.degree)
43             printf("+");
44     }
45     printf("\n");
46 }
47
48 void main()
49 {
50     system("title Kimkideok");
51
52     polynomial A = { 3, { 4, 3, 5, 0 } };
53     polynomial B = { 4, { 3, 1, 0, 2, 1 } };
54     polynomial C;
55     C = addPoly(A, B);
56     printf("\n A(x)="); printPoly(A);
57     printf("\n B(x)="); printPoly(B);
58     printf("\n C(x)="); printPoly(C);
59     getchar();
60 }
```

MAX 정의(a와 b의 크기를 비교했을 때 a가 크면 a, 작으면 b)  
MAX\_DEGREE정의  
polynomial 구조체 선언  
addPoly 다항식 A와 B를 입력받아 두 다항식을 더한 값을 C에 저장해 C를 반환하는 함수.  
변수 선언  
while문의 조건은 A와 B의 차수가 0이 될 때 까지.  
A, B 두 변수의 차수가 같다면 두 계수를 더해주고 다르다면 큰 것의 계수만 쓴다.  
C를 반환.  
다항식으로 보이게 출력해주는 printPoly함수 선언.  
3차수이며 계수가 내림차순으로 4, 3, 5, 0인 다항식 A선언.  
4차수이며 계수가 내림차순으로 3, 1, 0, 2, 1인 다항식 B선언.  
A와 B를 더한 C 설정.  
세 다항식 모두 출력.

## 2) 실행화면



```
Kimkideok
A(x)= 4x^3+ 3x^2+ 5x^1+ 0x^0
B(x)= 3x^4+ 1x^3+ 0x^2+ 2x^1+ 1x^0
C(x)= 3x^4+ 5x^3+ 3x^2+ 7x^1+ 1x^0
```

## 3) 분석

이 코드는 최대 50차수의 두 다항식을 더한 후 출력해주는 코드이다.

다항식을 만드는 구조체를 먼저 선언을 해주었고 구조체 변수를 초기화 해줄 때 는 차수 먼저 초기화해준 후 계수는 배열로 초기화해준다.

같은 차수의 계수끼리 더해주고 다른차수는 그대로 사용한다.

두 다항식과 더한 다항식을 높은차수부터 순서대로 다항식형식으로 출력해준다.

### 3. 희소 행렬 만들기

#### 1) 코드설명

```
1  #define _CRT_SECURE_NO_WARNINGS
2  #include <stdio.h>
3  #include <stdlib.h>
4  #include <Windows.h>
5
6  typedef struct {
7      int row;
8      int col;
9      int value;
10 } SprsMtrx;
11
12 SprsMtrx *smCreate(int A[][7], int row, int col, int *cnt)
13 {
14     int i, j;
15     SprsMtrx *S;
16     for (j = 0; j < row; j++)
17     {
18         for (i = 0; i < col; i++)
19         {
20             if (A[j][i]) // 0이 아닌 데이터 개수 구하기;
21                 (*cnt)++;
22         }
23     }
24     // 초기값 때문에 공간이 1개 더 필요함
25     S = (SprsMtrx*)malloc(sizeof(SprsMtrx)*((*cnt) + 1));
26     // 시작 값 행, 열, 데이터 개수 넣기
27     S[0].row = row;
28     S[0].col = col;
29     S[0].value = *cnt;
30     // 희소행렬의 각 원소에 값 할당하기
31     *cnt = 1;
32     for (j = 0; j < row; j++)
33     {
34         for (i = 0; i < col; i++)
35         {
36             if (A[j][i])
37             {
38                 S[*cnt].row = j;
39                 S[*cnt].col = i;
40                 S[*cnt++].value = A[j][i];
41             }
42         }
43     }
44     return S;
45 }
46
47 int main()
48 {
49     system("title Kimkideok");
50
51     int i, j, cnt = 0;
52     // 변경 전 행렬 값
53     int A[][7] = { { 0, 0, 2, 0, 0, 0, 12 },
54                   { 0, 0, 0, 0, 7, 0, 0 },
55                   { 23, 0, 0, 0, 0, 0, 0 },
56                   { 0, 0, 0, 31, 0, 0, 0 },
57                   { 0, 14, 0, 0, 0, 25, 0 },
58                   { 0, 0, 0, 0, 0, 0, 6 },
59                   { 52, 0, 0, 0, 0, 0, 0 },
60                   { 0, 0, 0, 0, 11, 0, 0 } };
61
62     for (j = 0; j < 8; j++)
63     {
64         for (i = 0; i < 7; i++)
65         {
66             printf("%4d", A[j][i]);
67         }
68         printf("\n");
69     }
70     SprsMtrx *S = smCreate(A, sizeof(A) / (sizeof(int) * 7), 7, &cnt);
71     printf("====희소 행렬 출력====\n"); // 출력 하기
72     for (i = 0; i < cnt; i++)
73     {
74         printf("%4d %4d %4d\n", S[i].row, S[i].col, S[i].value);
75     }
76     return 0;
77 }
```

SprsMtrx구조체 선언

포인터변수 선언

행과 열을 바꿔가며 0이 아닌 데이터를 선별

동적메모리 추가

0이 아닌 데이터의 배열 생성

배열 초기화

배열 출력

0이 아닌 데이터(희소 행렬) 출력

## 2) 실행화면

```
0 0 2 0 0 0 12
0 0 0 0 7 0 0
23 0 0 0 0 0 0
0 0 0 31 0 0 0
0 14 0 0 0 25 0
0 0 0 0 0 0 6
52 0 0 0 0 0 0
0 0 0 0 11 0 0
=====희소 행렬 출력=====
8 7 10
0 2 2
0 6 12
1 4 7
2 0 23
3 3 31
4 1 14
4 5 25
5 6 6
6 0 52
7 4 11
```

## 3) 분석

이 코드는 어떤 배열에서 0이 아닌 데이터를 선별해주는 코드이다.

희소 행렬 출력을 보면 초기 값 8, 7, 10 이 출력된 후 각 데이터의 행과 열, 데이터 값이 출력되는 것을 확인할 수 있다.

0이 아닌 값을 선별하는 방법은 행과 열을 순서대로 바꿔가면서 모든 값을 0과 비교하여 0 이라면 다음 값으로 넘어가고 0이 아니라면 그 데이터의 행값, 열값, value값을 새로운 행렬에 각 행마다 저장해준다. 그리고 마지막으로 그 저장한 새로운 행렬을 출력해보면 0이 아닌 각 데이터의 행, 열, value를 한번에 확인할 수 있다.