

5장 프로그래밍 실습_C 코드

[프로그래밍 실습 2]

```
#include <stdio.h>
#include <Windows.h>

// scanf()로 인해 발생하는 경고 해제
#pragma warning(disable : 4996)

#define MAXITEM 10    // 관계행렬의 최대 크기를 위한 상수

void main()
{
    // 관계행렬의 정의 및 0으로 초기화
    int m[MAXITEM + 1][MAXITEM + 1] = { 0 };
    int i, j, x, y, max;    // 관계행렬의 연산을 위한 변수

    // 사용자로부터 관계행렬의 크기와 관계행렬을 입력받기
    printf("입력으로 넣을 관계행렬의 행의 크기는?\n");
    scanf("%d", &max);
    printf("\n");
    printf("1과 0으로 데이터를 입력하라.\n");
    for (i = 1; i <= max; i++)
    {
        for (j = 1; j <= max; j++)
        {
            scanf("%d", &m[i][j]);
        }
    }
    printf("\n");
    // 행렬 m에 대해 관계가 3인 경로를 계산하면서 출력
    for (i = 1; i <= max; i++)
    {
        for (j = 1; j <= max; j++)
        {
            if (m[i][j] == 1)    // 값이 1인 셀에 대해 계속 진행
            {
                for (x = 1; x <= max; x++)
                {
                    if (m[j][x] == 1)
                    {
                        for (y = 1; y <= max; y++)
                        {
                            if (m[x][y] == 1)
                            {
                                // 발견된 길이가 3인 경로를 출력
                                printf("(%2d%2d%2d%2d) => (%d..%d)\n", i, j, x, y, i, y);
                            }
                        }
                    }
                }
            }
        }
    }
    system("PAUSE");
}
```

[프로그래밍 실습 3]

```
#include<stdio.h>
#include<conio.h>
#include<Windows.h>

#pragma warning (disable : 4996)

#define domain 20    // 관계행렬의 크기를 위한 상수

// 함수 원형 정의
void readfile(char fn[13], char adjancy[domain + 1][domain + 1]);
void setout(int set_size);
void relationout(int** R, int size);

void main()
{
    int i, j, k, test;    // 관계 연산을 위한 변수 정의
    char fn[] = "pp2-1.DAT";    // 파일 이름
    char adjancymx[domain + 1][domain + 1] = { 0 };    // 관계행렬의 정의 및 초기화

    char reflexivity, symmetry, transitivity;    // 반사, 대칭, 추이관계 확인을 위한 변수

    readfile(fn, adjancymx);    // 파일에서 관계를 읽음
    setout(domain);    // 집합 A = {1, 2, 3, ..., 20}을 출력
    relationout(adjancymx, domain);    // 집합 A에 대한 관계 R을 출력

    // R이 반사관계인지 아닌지 확인하고 결과를 출력
    reflexivity = 1;
    for (i = 1; i <= domain; i++)
    {
        // iRi가 없으면 바로 반복문을 멈춤
        if (adjancymx[i][i] == 0)
        {
            reflexivity = 0;
            break;
        }
    }
    if (reflexivity == 1)
    {
        printf("%2cR은 반사관계이다.\n", ' ');
    }
    else
    {
        printf("%2cR은 반사관계가 아니다.\n", ' ');
    }

    // R이 대칭관계인지 아닌지 확인하고 결과를 출력
    symmetry = 1;
    test = 1;
    for (i = 1; i <= domain; i++)
    {
        for (j = 1; j <= domain; j++)
        {
            // iRj가 있을 때 jRi가 없으면 바로 반복문을 멈춤
            if (adjancymx[i][j] == 1 && adjancymx[j][i] == 0)
            {
                symmetry = 0;
                test = 0;
            }
        }
    }
}
```

```

        break;
    }
    }
    if (test == 0)
    {
        break;
    }
}
if (symmetry == 1)
{
    printf("%2cR은 대칭관계이다.\n", ' ');
}
else
{
    printf("%2cR은 대칭관계가 아니다.\n", ' ');
}

// R이 추이관계인지 아닌지 확인하고 결과를 출력
transitivity = 1;
test = 1;
for (i = 1; i <= domain; i++)
{
    for (j = 1; j <= domain; j++)
    {
        if (adjancymx[i][j] == 1)
        {
            for (k = 1; k <= 5; k++)
            {
                // iRj와 jRk가 있고, iRk가 없으면 반복문을 멈춤
                if (adjancymx[j][k] == 1 && adjancymx[i][k] == 0)
                {
                    transitivity = 0;
                    test = 0;
                    break;
                }
            }
            if (test == 0)
            {
                break;
            }
        }
    }
    if (test == 0)
    {
        break;
    }
}
if (transitivity == 1)
{
    printf("%2cR은 추이관계이다.\n\n", ' ');
}
else
{
    printf("%2cR은 추이관계가 아니다.\n", ' ');
}
system("PAUSE");
}

```

```

// readfile 함수를 통해 텍스트 파일에 있는 관계를 읽음
void readfile(char fn[13], char adjancy[domain + 1][domain + 1])
{
    FILE* fp;
    int x, y;
    fp = fopen(fn, "r");
    while (!feof(fp))
    {
        fscanf(fp, "%d %d", &x, &y);
        adjancy[x][y] = 1;
    }
    fclose(fp);
}

// setout 함수가 행렬 mx를 출력
void setout(int set_size) {
    printf("\nA = { ");
    for (int j = 1; j < set_size + 1; j++)
    {
        printf("%d,", j);
    }
    printf("\b }\n\n");
}

// relationout 함수를 통해 관계 R을 출력
void relationout(int R[domain + 1][domain + 1], int size) {
    printf("\nR = { ");
    for (int i = 1; i < size + 1; i++) {
        for (int j = 1; j < size + 1; j++) {
            if (R[i][j] == 1) {
                printf("(%d, %d),", i, j);
            }
        }
    }
    printf("\b }\n\n");
}

```

[프로그래밍 실습 4]

```

#include<stdio.h>
#include<conio.h>
#include<windows.h>
#include<process.h>

#pragma warning(disable : 4996)

void main()
{
    // 와샬 알고리즘을 진행하기 위한 변수 및 이중 포인터 변수 정의
    int i, j, k, m, n, s, t, domain, ** Mrs;

    // 관계행렬의 크기를 사용자로부터 입력받음
    printf("\n 집합 A는 1부터 n까지의 자연수를 갖는다. 자연수 n을 입력하라. : ");
    scanf("%d", &domain);

    // 입력될 관계를 저장하기 위한 이중 포인터로 이루어진 2차원 배열의 할당 및 초기화
    Mrs = (int**)malloc(sizeof(int*) * domain);
    for (i = 0; i < domain; i++)

```

```

{
    *(Mrs + i) = (int*)calloc(domain,sizeof(int));
}
// - 1 - 10이 입력될 때까지 사용자로부터 관계를 입력받으면서 Mrs에 대입
printf("\n집합 A에 대한 관계들을 (멈추기 위해 -1 -1)을 입력하라.:\n");
do
{
    scanf("%d %d", &i, &j);
    if (i == -1)
    {
        break;
    }
    else
    {
        (*(Mrs + i - 1) + j - 1) = 1;
    }
} while (1);

// 입력된 관계행렬을 출력
printf("\n[ 관계행렬 ]\n");
for (i = 0; i < domain; i++)
{
    for (j = 0; j < domain; j++)
    {
        printf("%d ", (*(Mrs + i) + j));
    }
    printf("\n");
}
printf("\n<<아무 키나 입력하라.>>");
getch();
printf("\n");
// 해당 제일 바깥쪽 for 문으로 관계행렬(Mrs)을 계산해서 단계별로 출력 진행
for (i = 0; i < domain; i++)
{
    for (j = i + 1; j < i + domain; j++)
    {
        for (k = i + 1; k < i + domain; k++)
        {
            m = j % domain;
            n = k % domain;
            if (Mrs[i][n] == 1 && Mrs[m][i] == 1)
            {
                Mrs[m][n] = 1;
            }
        }
    }
    // 한 단계가 끝나면 수정된 관계행렬을 출력
    printf("      [W%d]      \n", i + 1);
    for (s = 0; s < domain; s++)
    {
        for (t = 0; t < domain; t++)
        {
            printf("%d ", (*(Mrs + s) + t));
        }
        printf("\n");
    }
    printf("\n");
    // 한 단계가 끝난 후에 관계를 출력
    for (s = 0; s < domain; s++)

```

```
{
    for (t = 0; t < domain; t++)
    {
        if (*(Mrs + s) + t == 1)
        {
            printf("(%d, %d) ", s + 1, t + 1);
        }
    }
    // 다시 진행하거나 for 문을 멈춤
    printf("\n<<아무 키나 입력하라.>>");
    getch();
    printf("\n\n");
}
// 할당된 메모리 공간을 해제
for (i = 0; i < domain; i++)
{
    free(*(Mrs + i));
}
free(Mrs);
system("PAUSE");
}
```