

| 10주차 과제 | | | | | |
|---------|---|----|------------|----|-----|
| 학년 | 2 | 학번 | 2016707079 | 이름 | 하상천 |

1. 과제 설명

과제1

2차원 배열을 통해 7X7 Map을 만들고, (0,0)에서 시작하여 달팽이 모양으로 회전하며 X가 있는 위치까지의 최단 거리를 계산하여 출력하는 프로그램을 구현한다.

단, main 문에서는 map 만을 선언하고, 최단거리를 계산하는 부분은 다른 함수를 만들어서 구현하여 최종 결과값을 다시 main문에서 확인할 수 있도록 한다.

2행당 하나의 X를 가지는 map으로 한정하고, 결과화면에 본인의 Map을 꼭 출력할 수 있도록 한다.

다른 Map을 적용해도 최단거리 계산이 정확히 나오는지 여부를 보고서에 작성하여 확인할 수 있도록 한다.

과제2

과제 1과 방식은 동일하나, 과제 2에서는 대각선 루트를 포함하는 최단거리를 계산한다.

과제 1과 마찬가지로 main문에서는 7X7 Map 선언과 결과 화면을 출력하도록 구현하고, 나머지 구현은 함수를 사용하도록 한다.

2행당 하나의 X를 가지는 map으로 한정하고, 결과화면에 본인의 Map을 꼭 출력할 수 있도록 한다.

다른 Map을 적용해도 마찬가지로 최단 거리 계산이 정확히 나오는지 여부를 보고서에 작성하여 확인할 수 있도록 한다.

2. 과제 이론

If문

조건을 주어 그 조건을 만족할 때에 실행하도록 한다.

Ex) if(i==1) 이면 i의 값이 1일 때만 실행하도록 한다.

그 외에는 다른 실행을 하고 싶다면 똑같은 형식으로 else if 또는 else로 조건을 주면 된다.

For문

일정한 조건을 주어서 원하는 만큼 반복할 수 있게 해주는 반복문

Ex) for(i=0;i<10;i++); ---> i=0으로 시작하고 한번 반복 할 때마다 i값이 1씩 증가한다. 그리고 i<10이라는 조건이 만족 할 때 까지만 반복한다. 즉 i=10이되면 반복하지 않는다.

Sqrt 함수

매개변수 x로 들어온 숫자에 루트를 씌워서 계산한 값을 반환해주는 함수
즉, 루트x를 구해주는 함수이다. Float형과 double형만 가능하다.

이 함수를 사용하기 위해서는 <math.h> 라이브러리가 필요하다.

Ex) float x = 25;

```
printf("sqrt(x) = %.2f", sqrt(x));
```

→sqrt(x) = 5.00

Abs 함수

정수(int형)의 절댓값을 구해주는 함수

이 함수를 사용하기 위해서는 <math.h> 라이브러리가 필요하다.

Ex) abs(-4) = 4

Continue 함수

반복문에서만 사용이 가능한 함수

반복문에서 continue를 만나면 더 이상 작업하지 않고 다시 반복문으로 돌아갑니다.

3. 주요 소스 설명

과제 1

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

```
#include<math.h> //절댓값을 구하는 함수 abs를 이용하기 위한 라이브러리
```

```
int calculator(char map[7][7]) //반환형이 int인 함수
```

```
{
```

```
    int i = 0;
```

```
    int j = 0;
```

```
    int before_x = 0;
```

```
    int before_y = 0;
```

```
    int after_x = 0;
```

```
    int after_y = 0;
```

```
    int count = 0;
```

```
    int length = 0;
```

```
    int all_length = 0; //총 최단거리
```

```
    for (int k = 0; k < 4; k++) //'X' 네개를 찾기 위해서 네번 반복한다.
```

```
    { if (map[0][0] == 'X') //(0,0)에 'X'가 있는 경우
```

```
        {
```

```
            map[0][0] = '0';
```

```
            continue;
```

```
        }
```

```
        for (int x = 1; x <= 11; x++) // map[i][j]라고 할때 달팽이모양으로 회전  
        하면 j+, i+, j-, j-, i-, i-, j+, j+, j+, i+, i+, i+, j-, j-, j-, j-, i-, i-, i-, i- 와 같은 규칙을 갖  
        는다. 그때 최대로 돌수있는경우는 j가 순차적으로 11번 +되는 경우이다.
```

```
        {
```

```
            if (x % 2 != 0) //홀수일때는 플러스
```

```
            {
```

```
                for (count = 0; count < x; count++) //1이면 한번만
```

플러스, 3이면 세번만플러스 같은 형태로 반복되기 때문

```
{  
  
    if (i < 0 || j < 0) //배열에서 마이너스로 가면 j  
++하고 넘어가기  
  
    {  
  
        j++;  
  
    }  
  
    else if (map[i][j] == 'X') // 'X'발견하면 그때 좌  
표 i,j를 after_x , after_y에 대입 하고 break로 가까운 반복문 탈출  
  
    {  
  
        after_x = i;  
  
        after_y = j;  
  
        break;  
  
    }  
  
    else j++; // 나머지는 j 1증가  
  
}
```

for (count = 0; count < x; count++) //1이면 한번만 플
러스, 3이면 세번만 플러스 같은 형태로 반복되기 때문

```
{  
  
    if (i < 0 || j < 0) //배열에서 마이너스로 가면  
i++하고 넘어가기  
  
    {  
  
        i++;  
  
    }  
  
    else if (map[i][j] == 'X') // 'X'발견하면 그때
```

좌표 i,j를 after_x , after_y에 대입 하고 break로 가까운 반복문 탈출

```
{
```

```
    after_x = i;
```

```
    after_y = j;
```

```
    break;
```

```
}
```

```
else i++; //나머지는 i 1증가
```

```
}
```

```
}
```

```
else //짝수일때는 마이너스
```

```
{
```

for (count = 0; count < x; count++) //2이면 두번만 마이너스, 4이면 네번만 마이너스 형태로 반복되기 때문

```
{
```

if (i < 0 || j < 0) //배열에서 마이너스로 가면 j-하고 넘어가기

```
{
```

```
    j--;
```

```
}
```

else if (map[i][j] == 'X')// 'X'발견하면 그때 좌표 i,j를 after_x , after_y에 대입 하고 break로 가까운 반복문 탈출

```
{
```

```
    after_x = i;
```

```
    after_y = j;
```

```

        break;

    }

    else j--; //나머지는 j 1감소

}

for (count = 0; count < x; count++) //2이면 두번만 마
이너스, 4이면 네번만 마이너스 형태로 반복되기 때문

{

    if (i < 0 || j < 0) //배열에서 마이너스로 가면 i-
-하고 넘어가기

    {

        i--;

    }

    else if (map[i][j] == 'X') // 'X'발견하면 그때 좌
표 i,j를 after_x , after_y에 대입 하고 break로 가까운 반복문 탈출

    {

        after_x = i;

        after_y = j;

        break;

    }

    else i--; //나머지는 i 1감소

}

}

if (map[i][j] == 'X') //'X'를 발견하면 'O'로 바꾸어주고 break로
가까운 반복문 탈출

```

```

        {

            map[i][j] = 'O';

            break;

        }

    }

    if (k == 0) //첫번째 일때는 (0,0)과 발견 점과의 거리를 구한다.

    {

        length = (after_x - 0) + (after_y - 0);

    }

    else if (k != 0) //나머지의 경우는 절댓값을 구하는 함수 abs를 이용해서
    발견 점과 직전의 발견 점과의 거리를 구한다.

    {

        length = abs((after_x - before_x)) + abs((after_y - before_y));

    }

    before_x = after_x; //발견한 점의 좌표를 before_x에 대입

    before_y = after_y; //발견한 점의 좌표를 before_y에 대입

    all_length += length; //구한 최단거리를 all_length에 더해준다.

    }

    return all_length; // 총 최단거리 반환

}

int main()

{

    char map[7][7] = { 'O','O','O','X','O','O','O',

```

```

'O','O','O','O','O','O','O',
'X','O','O','O','O','O','O',
'O','O','O','O','O','O','O',
'O','O','O','X','O','O','O',
'O','O','O','O','O','O','O',
'O','O','O','O','O','O','X' }; //map선언

```

```

int shorttest = 0;

for (int x = 0; x < 7; x++)//map 출력
{
    for (int y = 0; y < 7; y++)
    {
        printf("%3c", map[x][y]);

    }

    printf("\n");
}

shorttest = calculator(map);

printf("최단거리 : %d", shorttest); //최단거리 출력
}

```

과제2


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

```
#include<math.h>//제곱근을 구하는 함수 sqrt를 이용하기 위한 라이브러리
```

```
float calculator(char map[7][7]) //반환형이 float인 함수
```

```
{
```

```
    int i = 0;
```

```
    int j = 0;
```

```
    int before_x = 0;
```

```
    int before_y = 0;
```

```
    int after_x = 0;
```

```
    int after_y = 0;
```

```
    int count = 0;
```

```
    float length = 0;
```

```
    float all_length = 0; //총 최단거리
```

```
    for (int k = 0; k < 4; k++) //'X' 네개를 찾기 위해서 네번 반복한다.
```

```
        { if( map[i][j] == 'X') //(0,0)에 'X'가 있는 경우
```

```
        {
```

```
            map[i][j] = '0';
```

```
            continue;}

            for (int x = 1; x <= 11; x++) // map[i][j]라고 할때 달팽이모양으로 회전
            하면 j+, i-,   j-, j-, i-, i-,   j+, j+, j+, i+, i+, i+,   j-, j-, j-, j-, i-, i-, i-, i- 와 같은 규칙을 갖
            는다. 그때 최대로 돌수있는경우는 j가 순차적으로 11번 +되는경우이다.
```

```
                {
```

```
                    if (x % 2 != 0) //홀수일때는 플러스
```

```
                    {
```

for (count = 0; count < x; count++) //1이면 한번만 플러스, 3이면 세번만플러스 같은 형태로 반복되기 때문

```
{  
  
    if (i < 0 || j < 0) //배열에서 마이너스로 가면 j  
    ++하고 넘어가기  
  
    {  
  
        j++;  
  
    }  
  
    else if (map[i][j] == 'X') // 'X'발견하면 그때 좌  
표 i,j를 after_x , after_y에 대입 하고 break로 가까운 반복문 탈출  
  
    {  
  
        after_x = i;  
  
        after_y = j;  
  
        break;  
  
    }  
  
    else j++; // 나머지는 j 1증가  
  
}
```

for (count = 0; count < x; count++) //1이면 한번만 플러스, 3이면 세번만 플러스 같은 형태로 반복되기 때문

```
{  
  
    if (i < 0 || j < 0) //배열에서 마이너스로 가면  
i++하고 넘어가기  
  
    {  
  
        i++;  
  
    }  
  
}
```

else if (map[i][j] == 'X') // 'X'발견하면 그때 좌
표 ij를 after_x , after_y에 대입 하고 break로 가까운 반복문 탈출

```
{  
  
    after_x = i;  
  
    after_y = j;  
  
    break;
```

```
}  
  
else i++; //나머지는 i 1증가
```

```
}
```

```
}
```

else //짝수일때는 마이너스

```
{
```

for (count = 0; count < x; count++) //2이면 두번만
마이너스, 4이면 네번만 마이너스 형태로 반복되기 때문

```
{
```

if (i < 0 || j < 0) //배열에서 마이너스로 가면 j-
-하고 넘어가기

```
{  
  
    j--;  
  
}
```

else if (map[i][j] == 'X') // 'X'발견하면 그때 좌
표 ij를 after_x , after_y에 대입 하고 break로 가까운 반복문 탈출

```
{  
  
    after_x = i;
```

```

        after_y = j;

        break;

    }

    else j--; //나머지는 j 1감소

}

for (count = 0; count < x; count++) //2이면 두번만 마
이너스, 4이면 네번만 마이너스 형태로 반복되기 때문

{

    if (i < 0 || j < 0) //배열에서 마이너스로 가면 i-
-하고 넘어가기

    {

        i--;

    }

    else if (map[i][j] == 'X') // 'X'발견하면 그때 좌
표 i,j를 after_x , after_y에 대입 하고 break로 가까운 반복문 탈출

    {

        after_x = i;

        after_y = j;

        break;

    }

    else i--; //나머지는 i 1감소

}

}

if (map[i][j] == 'X') //'X'를 발견하면 'O'로 바꾸어주고 break로

```

가까운 반복문 탈출

```
        {

            map[i][j] = 'O';

            break;

        }

    }

    if (k == 0) //첫번째 일때는 (0,0)과 발견 점과의 거리(대각선포함)를 sqrt
함수를 이용해서 구한다.

    {

        length = sqrt((after_x - 0)*(after_x - 0) + (after_y - 0)*(after_y -
0));

    }

    else if (k != 0) //나머지의 경우는 제곱근을 구하는 함수 sqrt를 이용해서
발견 점과 직전의 발견 점과의 거리(대각선포함)를 구한다.

    {

        length = sqrt((after_x - before_x)*(after_x - before_x) + (after_y -
before_y)*(after_y - before_y));

    }

    before_x = after_x; //발견한 점의 좌표를 before_x에 대입

    before_y = after_y; //발견한 점의 좌표를 before_y에 대입

    all_length += length; //구한 최단거리를 all_length에 더해준다.

}

return all_length; // 총 최단거리 반환

}
```

```

int main()

{

    char map[7][7] = { 'O','O','O','X','O','O','O',

        'O','O','O','O','O','O','O',

        'X','O','O','O','O','O','O',

        'O','O','O','O','O','O','O',

        'O','O','O','X','O','O','O',

        'O','O','O','O','O','O','O',

        'O','O','O','O','O','O','X' };    //map선언

    float shorttest = 0;

    for (int x = 0; x < 7; x++)    //map 출력

    {

        for (int y = 0; y < 7; y++)

        {

            printf("%3c", map[x][y]);

        }

        printf("\n");

    }


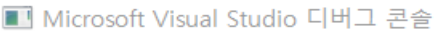
    shorttest = calculator(map);

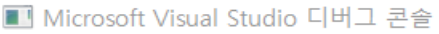
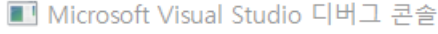
    printf("최단거리 : %f", shorttest);    //최단거리 출력

}

```

4. 실행화면

| | |
|--|---|
|  <pre> 0 0 0 X 0 0 0 0 0 0 0 0 0 0 X 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 X 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 X 최단거리 : 16 C:\Users\seomk\source\repos\Project34 이 창을 닫으려면 아무 키나 누르세요. </pre> |  <pre> 0 X 0 0 0 0 0 0 0 0 0 0 0 0 0 0 X 0 0 0 0 0 0 0 0 0 0 0 X 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 X 0 최단거리 : 15 C:\Users\seomk\source\repos\Project34 이 창을 닫으려면 아무 키나 누르세요. </pre> |
| 과제1 실행화면1 | 과제1 실행화면2 |

| | |
|---|--|
|  <pre> 0 0 0 X 0 0 0 0 0 0 0 0 0 0 X 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 X 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 X 최단거리 : 13.211102 C:\Users\seomk\source\repos\Project34 이 창을 닫으려면 아무 키나 누르세요. </pre> |  <pre> 0 X 0 0 0 0 0 0 0 0 0 0 0 0 0 0 X 0 0 0 0 0 0 0 0 0 0 0 X 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 X 0 최단거리 : 11.449659 C:\Users\seomk\source\repos\Project34 이 창을 닫으려면 아무 키나 누르세요. </pre> |
| 과제2 실행화면1 | 과제2 실행화면2 |

5. 고찰

이번주 과제는 저번주에 비해 엄청 어려워진 것 같다. 처음에 엄두가 나지 않아서 달팽이모양으로 회전하는 것이 어떻게 이동하는 것인지 노트에 적어보았다. 적어 보니까 나름의 규칙이 있었다. Map[i][j] 라고 했을 때 j+, i+, j-, j-, i-, i-, j+, j+, j+, i+, i+, i+, j-, j-, j-, j-, i-, i-, i-, i- 와 같은 규칙으로 이동한다. 과제1 같은 경우는 for문과 if문을 이용해서 구했고, 'X'를 발견하면 'O'로 바꾸어 주었다. (0,0)에 'X'가 있는 경우는 최단거리를 계산할 필요가 없기 때문에 continue함수를 사용하였다. 그런 경우, k가 1이되면 length = abs((after_x - before_x)) + abs((after_y - before_y))로 최단거리를 구하는데, before_x, before_y는 원점이어야 하므로 0으로 초기화 해주었다. continue함수는 반복문안에서만 사용이 가능하다. (0,0)에 'X'가 없는 경우는 처음에 'X'가 발견된 좌표를 after_x, after_y에 대입해주고 원점과의 거리를 구했다. 그 값을 all_length에 더해주었다. 그리고 after_x, after_y를 before_x, before_y에 다시 대입 해 놓는다. for문을 다시 반복하여 달팽이모양의

로 회전시키고 'X'가 발견된 좌표를 after_x, after_y로 대입한다. 점(after_x, after_y)과 점(before_x, before_y)사이의 거리를 구한다. 거리는 양수이기 때문에 절댓값을 구하는 함수 abs를 이용했고 <math.h>라이브러리를 이용했다. 그 값을 all_length에 더해주었다. for문을 많이 사용해서 'X'를 발견했을 때 break; 문을 많이 사용 할 수밖에 없다는게 조금 아쉽다. 과제2는 과제1에서 피타고라스의 정리만 사용하니까 쉽게 나왔다. 과제1과 같은 방법으로 구하고 제곱근을 구하는 함수 sqrt를 이용해서 두 점 사이의 거리를 구했다. 마찬가지로 <math.h> 라이브러리를 이용했다.