민코딩 수업노트 LV-14

반복문 Training

배우는 내용

1. 배열, 1중 For문 다루기
2. 2중 For 응용 훈련
3. 다중 Max
4. 다중 Counting
5. 선택정렬

복잡한 For 문제

For문은 손쉽게 쓸 수 있도록 이 과정이 필요하다.

최대한 정답코드 / 힌트코드를 보지 않고, 고민을 해가면서 풀어야 한다.

또한, 버그가 발생하면 디버깅을 통해 원인을 찾아내자.

[중요] 버그발생시?

안 좋은 방법 : 될 때까지 소스코드를 수정해보며 마구 찍어보기

프로그래밍을 생각하지 않고 코딩하는 것입니다.
추리 없이, 기억에 의존해서 코딩하면 안됩니다.

소스코드를 눈으로 버그 찾기

매우 간단한 소스코드는 눈으로 찾으셔도 괜찮습니다.
하지만 소스코드가 길어지고 복잡해 질 수록 버그 찾기 어려워집니다.
Trace없이는 디버깅하는 시간이 오래 걸리거나, 버그를 스스로 찾아내기 힘듭니다.

[필수] Trace로 디버깅하기

Trace 사용 경험이 많아질수록, 능숙해질수록 버그 찾는 디버깅시간이 줄어듭니다.
프로그래머에게 디버깅 능력이 매우 중요합니다.
버그가 발생하면 꼭 Trace로 버그를 찾아보시길 바랍니다.

문제 소개 : 각 줄의 MAX값 구하기

목표 : 배열의 각 줄의 MAX값을 출력하는 문제

4	1	3	**14**
5	**32**	1	5
6	3	**71**	2
43	2	1	6

출력결과 : 14 32 71 42

1중For를 쓰면 한 줄의 MAX값을 구할 수 있다.

2중 For를 쓰면 **여러 줄의 MAX값**을 구할 수 있다.

맨 윗줄의 MAX값 구하기

먼저 **가장 윗줄의 MAX값**을 구하는 소스코드

4	1	3	**14**
5	**32**	1	5
6	3	**71**	2
43	2	1	6

```
int vect[4][4] = {
    4, 1, 3, 14,
    5, 32, 1, 5,
    6, 3, 71, 2,
    43, 2, 1, 6,
};

int x, y;
int max;

max = 0;
for (x = 0; x < 4; x++)
{
    if (vect[0][x] > max)
    {
        max = vect[0][x];
    }
}

cout << max;
```

각 줄의 MAX값 구하기

각 줄의 MAX값을 구하는 소스코드

4	1	3	**14**
5	**32**	1	5
6	3	**71**	2
43	2	1	6

max 초기화 위치가 중요하다.

```
int vect[4][4] = {
    4, 1, 3, 14,
    5, 32, 1, 5,
    6, 3, 71, 2,
    43, 2, 1, 6,
};

int x, y;
int max;

for (y = 0; y < 4; y++)
{
    max = 0;
    for (x = 0; x < 4; x++)
    {
        if (vect[y][x] > max)
        {
            max = vect[y][x];
        }
    }

    cout << max << " ";
}
```

숫자 1 개수 찾기

숫자 1이 몇 개인지 찾기

간단히 Count 하면 됨


```
int vect[6] = { 5, 1, 3, 7, 2, 1 };
int count;
int x;

count = 0;
for (x = 0; x < 6; x++)
{
    if (vect[x] == 1) count++;
}

cout << count;
```

숫자 1 ~ 4 개수 찾기

숫자 1, 2, 3, 4가 각각 몇 개인지 찾기

2중 For문을 돌려 해결 가능

5	1	3	7	2	1

```
int vect[6] = { 5, 1, 3, 7, 2, 1 };
int count;
int x, y;

for (y = 1; y <= 4; y++)
{
    count = 0;
    for (x = 0; x < 6; x++)
    {
        if (vect[x] == y) count++;
    }

    cout << y << ":" << count << endl;
}
```

```
1:2
2:1
3:1
4:0
```

다중 count

입력받은 숫자가 vect배열에
몇 개 있는지 Count하는 문제

vect	1	5	1	5	3

input	1	2	3

실행결과
1 : 2개
2 : 0개
3 : 1개

```
for (y=0; y<3; y++)
{
    count = 0;
    for (x=0; x<5; x++)
    {
        if (vect[x] == input[y])
        {
            count++;
        }
    }

    cout << count;
}
```

선택정렬

정렬하는 방법 중 가장 기본적인 알고리즘

* 알고리즘 : 어떤 문제를 해결하기 위한 절차 or 방법

1차원 배열을 숫자 순서대로 나열시키는 것이 목표

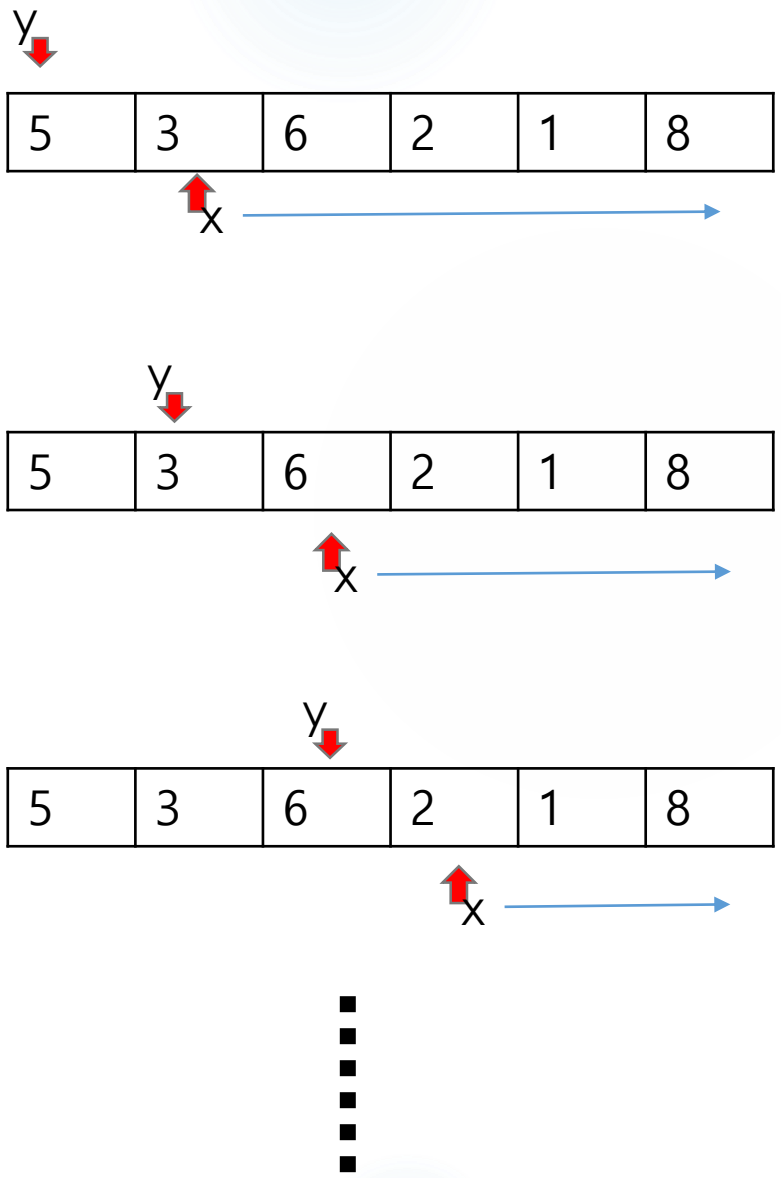
선택정렬 원리1

중첩 2중 for문을 돌린다

y for문은 [0]부터 [6]까지 탐색한다.
x for문은 [y + 1]번부터 [6]까지 탐색한다.

```
int y, x;

for (y=0; y<6; y++)
{
    for (x=y+1; y<6; y++)
    {
        //어떤 소스코드
    }
}
```



선택정렬 원리2

Y와 x가 선택한 숫자를 비교 함

y는 항상 x의 왼쪽, x는 항상 y의 오른쪽에 배치 됨

y가 선택한 값이, x가 선택한 값 보다 크다면 **SWAP**

```
int y, x;

for (y=0; y<6; y++)
{
   for (x=y+1; y<6; y++)
   {
      if ( vect[y] > vect[x])
      {
         temp = vect[y];
         vect[y] = vect[x];
         vect[x] = temp;
      }
   }
}
```