

민선생코딩학원 시작반

수업노트 LV-15



배우는 내용

1. 복잡한 2중 For문 훈련
2. 두 문장이 같은지 비교하기 - Flag
3. 두 문장이 같은지 비교하기 - 함수 사용

복잡한 2중 For

- ▶ For문은 손쉽게 쓸 수 있도록 이 과정이 필요
- ▶ 최대한 정답코드 / 힌트코드를 보지 않고,
고민을 해가면서 풀어야 함

[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.
Trace없이 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.

세 문장 길이 구하기

```
char vect[3][10] = { "FIRST", "SECOND", "TH" };  
  
int n1, n2, n3;  
int x;  
  
//첫번째 문장  
for (x = 0; x < 10; x++)  
{  
    if (vect[0][x] == '\\0')  
    {  
        n1 = x;  
        break;  
    }  
}  
//두번째 문장  
for (x = 0; x < 10; x++)  
{  
    if (vect[1][x] == '\\0')  
    {  
        n2 = x;  
        break;  
    }  
}  
//세번째 문장  
for (x = 0; x < 10; x++)  
{  
    if (vect[2][x] == '\\0')  
    {  
        n3 = x;  
        break;  
    }  
}  
  
cout << n1 << " " << n2 << " " << n3 << endl;
```

- ▶ 왼쪽 소스코드는 기존에 배웠던 방식
- ▶ for문을 세 번 사용하여 문장 길이를 각각 구한다.

**For문으로
한번에 묶을 수 있다**

세 문장 길이구하기

```
char vect[3][10] = { "FIRST", "SECOND", "TH" };
int n1, n2, n3;
int x;

//첫번째 문장
for (x = 0; x < 10; x++)
{
    if (vect[0][x] == '\0')
    {
        n1 = x;
        break;
    }
}
//두번째 문장
for (x = 0; x < 10; x++)
{
    if (vect[1][x] == '\0')
    {
        n2 = x;
        break;
    }
}
//세번째 문장
for (x = 0; x < 10; x++)
{
    if (vect[2][x] == '\0')
    {
        n3 = x;
        break;
    }
}

cout << n1 << " " << n2 << " " << n3 << endl;
```



배열로 만들어야
For문을 쓸 수 있다

y가 0일때, n[0]에 첫번째 문장 길이를
y가 1일때, n[1]에 두번째 문장 길이를
y가 2일때, n[2]에 세번째 문장 길이를
넣는다.

```
char vect[3][10] = { "FIRST", "SECOND", "TH" };
int n[3];
int x, y;

//첫번째 문장
for (y = 0; y < 3; y++)
{
    for (x = 0; x < 10; x++)
    {
        if (vect[y][x] == '\0')
        {
            n[y] = x;
            break;
        }
    }

    cout << n[y] << " ";
}

return 0;
```

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 MAX값을 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

5	1	3	7	2	1
---	---	---	---	---	---

```
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
---	---	---	---	---	---

[주의]

- 소스코드를 이해하는 것으로 끝내면 안됩니다.
- 힌트없이 직접 짤 수 있어야 합니다

```
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;
}
```

g++ Mi

```
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

[주의]

- 소스코드를 이해하는 것으로 끝내면 안됩니다.
- 힌트없이 직접 짤 수 있어야 합니다

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

두 숫자배열이 완전히 동일한지 비교

v1

5	1	3	7
---	---	---	---

v2

5	1	4	3
---	---	---	---

- ▶ 1중 for문을 돌며
같은 글자가 있는지 찾는 것이 아니라
다른 글자가 있는지 찾는다

두 숫자배열이 완전히 동일한지 비교

v1

5	1	3	7
---	---	---	---

v2

5	1	4	3
---	---	---	---

```
int v1[4] = { 5, 1, 3, 7 };
int v2[4] = { 5, 1, 4, 3 };

int x;
int flag = 0;

for (x = 0; x < 4; x++)
{
    if (v1[x] != v2[x])
    {
        flag = 1;
        break;
    }
}

if (flag == 0) cout << "동일";
else cout << "다른배열";
```

한 글자라도 다르면
flag = 1로 세팅한다

같은 글자 수인 문자열 비교하기 (버그발생 소스코드)

```
#include <iostream>
using namespace std;

int main()
{
    char name1[10] = "MIN";
    char name2[10] = "KFC";

    if (name1 == name2)
    {
        cout << "같은문장";
    }
    else
    {
        cout << "다른문장";
    }

    return 0;
}
```

▶ 왼쪽 소스코드는 버그 발생 소스코드

▶ **버그이유 : 배열은 비교가 되지 않는다.**
→ 한글자씩 비교를 해 주는 소스코드를 작성해야 함

```
char name1[10] = "MIN";
char name2[10] = "KFC";

int x;
int flag = 0;

for (x = 0; x < 3; x++)
{
    if (name1[x] != name2[x])
    {
        flag = 1;
        break;
    }
}

if (flag == 0) cout << "같음";
else cout << "다름";
```