

민선생코딩학원 시작반

---

# 수업노트 LV-07



# 배우는 내용

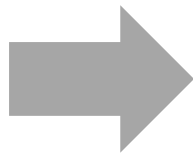
1. else if 와 %연산자
2. counting
3. Trace : Ctrl + F10
4. 2차 배열의 시작
5. 2차배열 + 2중 for문 체험하기

# else if

## ▶ if ~ else 문 뜻

→ 만약 [조건] 이 참이라면 [소스코드1]을 수행하고 그렇지 않으면 [소스코드2]를 수행

```
if (조건)
{
    소스코드1
}
else
{
    소스코드2
}
```



else if 문  
→ if문 에다가 **추가 조건**을 다는 명령어

## ▶ else if문 뜻

→ 만약 [조건1] 이 참이라면 [소스코드1]을 수행하고 그렇지 않고 [조건2]가 참이라면 [소스코드2]를 수행하고 그렇지 않고 [조건3]가 참이라면 [소스코드3]을 수행하고 그렇지 않으면 [소스코드4]를 수행한다.

```
if (조건)
{
    소스코드1
}
else if (조건2)
{
    소스코드2
}
else if (조건3)
{
    소스코드3
}
else
{
    소스코드4
}
```

# else if vs if문 여러 개

- ▶ 다음과 같은 문제를 풀어보자.  
input변수에 숫자 하나를 입력 받고,  
만약  $1 \leq \text{input} < 3$  이면 "#" 출력  
만약  $3 \leq \text{input} < 5$  이면 "@" 출력  
둘다 아니면 "\$" 출력

```
if (1 <= input && input < 3)
{
    cout << "#";
}

if (3 <= input && input < 5)
{
    cout << "@";
}
else
{
    cout << "$";
}
```

## 버그발생#1

input이 1 일때  
#\$ 출력되는 버그 발생

```
if (1 <= input && input < 3)
{
    cout << "#";
}
else
{
    cout << "$";
}

if (3 <= input && input < 5)
{
    cout << "@";
}
else
{
    cout << "$";
}
```

## 버그발생#2

input이 7일때  
\$\$ 출력되는 버그 발생

```
if (1 <= input && input < 3)
{
    cout << "#";
}
else if (3 <= input && input < 5)
{
    cout << "@";
}
else
{
    cout << "$";
}
```

## 정상동작

→ 여러 조건이 있을 때는 else if를 사용할 것

# else if 의 특징1

- ▶ 먼저 if문이 있어야 else if문을 쓸 수 있다

```
Input = 3;  
  
else if (input == 3)  
{  
    cout << "#";  
}  
else if (input == 4)  
{  
    cout << "$";  
}
```

## 버그발생

else if로 시작할 수 없음  
If로 시작하고, else if문으로  
추가 조건을 붙여야 합니다

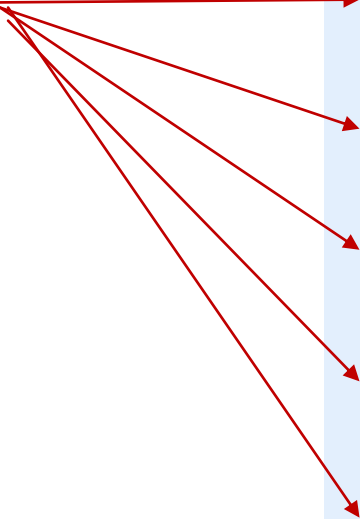
버그수정

```
Input = 3;  
  
if (input == 3)  
{  
    cout << "#";  
}  
else if (input == 4)  
{  
    cout << "$";  
}
```

# else if 의 특징2

- ▶ else if는 조건들을 계속 달 수 있음
- ▶ 소스코드는  
여러 조건 중에 한 곳만 실행하게 됨

```
Input = 3;  
  
if (input == 3)  
{  
    cout << "#";  
}  
  
else if (input == 4)  
{  
    cout << "$";  
}  
  
else if (input == 5)  
{  
    cout << "D";  
}  
  
else if (input == 6)  
{  
    cout << "T";  
}  
  
else  
{  
    cout << "%";  
}
```



# 숫자 3개 중 가장 큰 값을 출력하기

▶ 다음 문제를 풀어보자

숫자 3개를 변수 a, b, c에 입력 받았습니다  
이중 MAX를 출력 해 주세요

if (**a** >= b && **a** >= c)      ➔ a는 MAX인 조건

else if (**b** >= a && **b** >= c)      ➔ b는 MAX인 조건

else → c가 MAX인 조건

# mod 연산자

- ▶ 수학에서는 연산자는 : + - \* / 등 연산이 있지만, 프로그래밍에서는 % 라는 연산자가 추가로 존재 함
- ▶ mod연산자는 어떤 수를 **나누었을 때 나머지 값을 뜻함**
- ▶ **모듈러(modular)** 연산자 라고 부름

```
int a = 5 % 3;
```

//5를 3으로 나누었을 때  
값은 1이 나오고  
나머지는 2이 된다

따라서 **a = 2**

```
int b = 9 % 3;
```

//9를 3으로 나누었을 때  
값은 3이 나오고  
나머지는 0이 된다

따라서 **b = 0**

```
int c = 5 % 2;
```

//5를 2으로 나누었을 때  
값은 2이 나오고  
나머지는 1이 된다

따라서 **c = 1**



# mod 연산자 어떻게 활용할까?

- ▶ 짝수인지 아닌지 구분 가능함

```
cin >> a;

if (a % 2 == 0)
{
    cout << "짝수";
}
else
{
    cout << "홀수";
}
```

- ▶ 어떤 숫자의 배수인지 구분 가능

```
cin >> g;

if (g % 5 == 0)
{
    cout << "5의 배수 입니다";
}
```

# Counting 방식

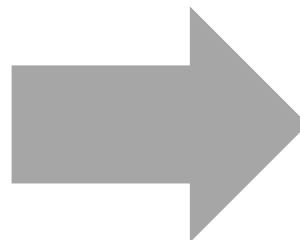
- ▶ Counting이란  
조건에 맞는 값이 몇 개인지 세는 것을 의미 함
- ▶ 바로 문제를 풀어보자  
하드코딩 되어있는 배열에서  
5보다 큰 숫자는 몇 개 있는지 Counting 해 주세요.  
(→ 정답 : 2개)
- ▶ Counting 방식을 써서 해결 가능

3	9	4	12	2
---	---	---	----	---

# Counting 방식

## 과정 설명

1. count라는 변수 만들고  
반드시 0으로 초기화 ( `int count = 0` )
2. for문으로 한 칸씩 탐색  
탐색하고있는 칸이 5보다 큰 수 이면 `count++`;



```
int arr[5] = {3, 5, 6, 12, 7};  
int x;  
int count = 0;  
  
for(x=0; x<5; x++)  
{  
    if(arr[x] > 5)  
    {  
        count++;  
    }  
}
```

count 변수 하나 만들고,  
특정 조건일때마다 count++ 해주는 것이  
counting 방식 코딩방법!

# Counting 응용 - 짝수

- ▶ 배열 안에 짝수가 몇 개인지 세는 방법

4	3	6	2	1	0	4
---	---	---	---	---	---	---

```
int main()
{
    int vect[7] = { 4, 3, 6, 2, 1, 0, 4 };

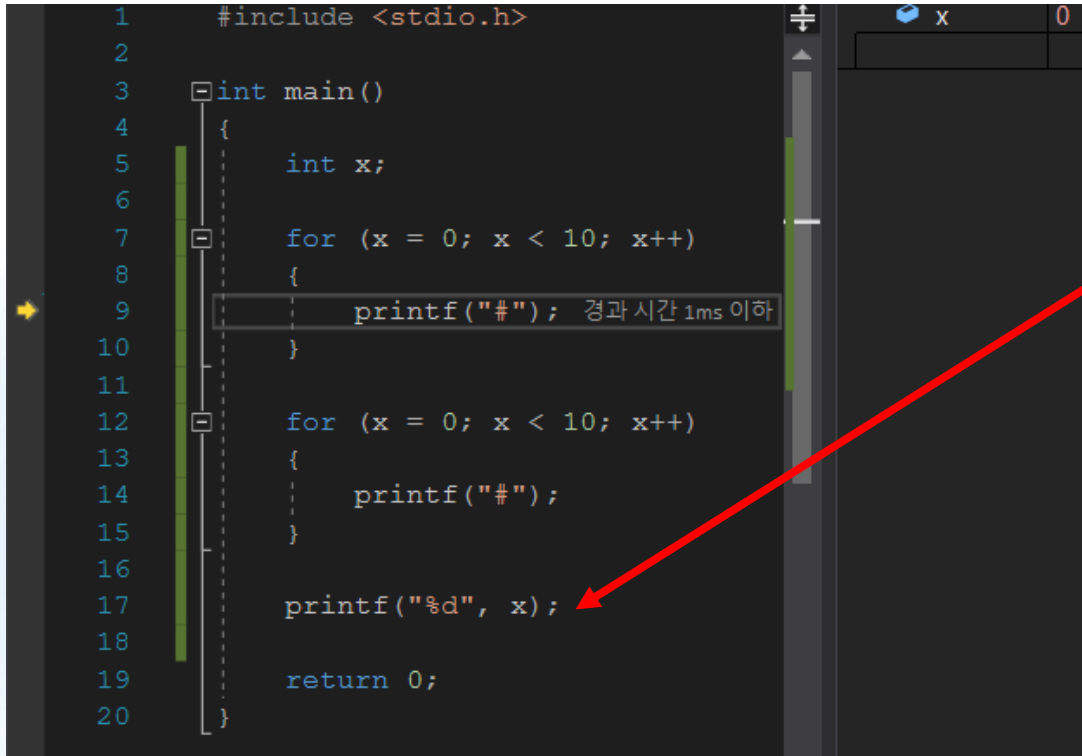
    int x;
    int count = 0;

    for (x = 0; x < 7; x++)
    {
        if (vect[x] % 2 == 0)
        {
            count++;
        }
    }

    cout << count;

    return 0;
}
```

# Trace 단축키 Ctrl + F10



```
1  #include <stdio.h>
2
3  int main()
4  {
5      int x;
6
7      for (x = 0; x < 10; x++)
8      {
9          printf("#"); 경과 시간 1ms 이하
10     }
11
12     for (x = 0; x < 10; x++)
13     {
14         printf("#");
15     }
16
17     printf("%d", x);
18
19     return 0;
20 }
```

이 지점까지 Trace를 하려면  
F10을 굉장히 많이 눌러야 합니다

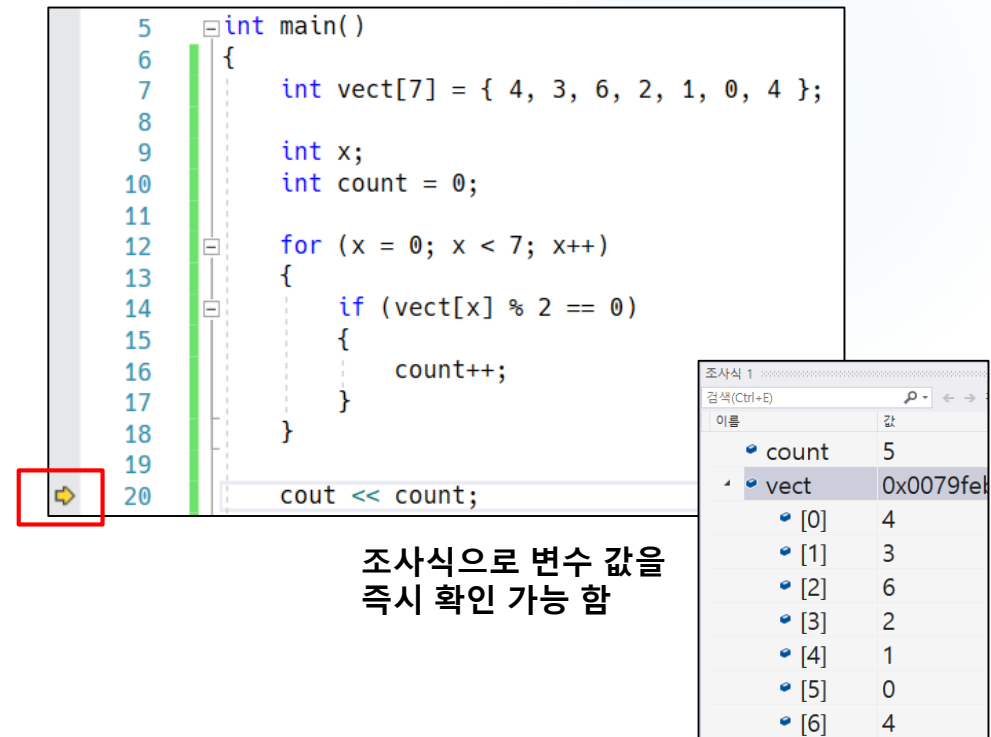
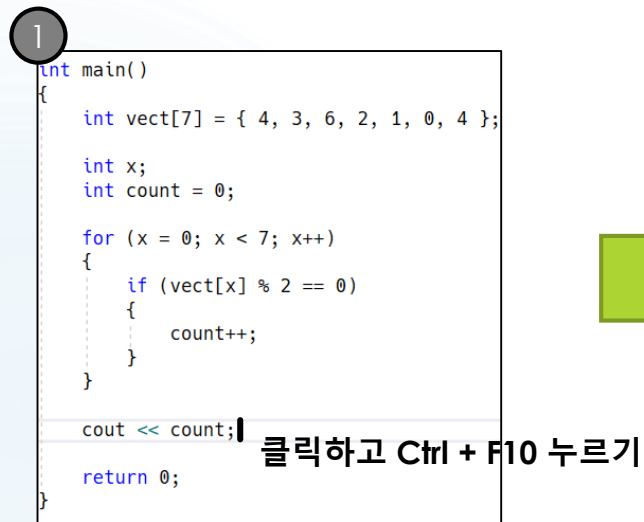
**한번에 Trace Pointer를 이동시키는 방법!**  
커서를 17번 Line에 두고  
**Ctrl + F10**을 누르면 됩니다

# Trace : Ctrl + F10 단축키

참고 : 암기해야 할 Trace 단축키

1. F10 : Trace 한줄실행 (함수 안으로 안들어감)
2. F11 : Trace 한줄실행 (함수 안으로 들어감)
3. Shift + F5 : Trace 종료
4. Ctrl + F10 : 커서 위치부터 Trace하기

▶ 커서가 있는 위치부터 Trace를 하는 단축키



# Ctrl + F10 : 변수 값을 즉시 확인가능1

- ▶ 버그를 찾아내는 첫 번째 방법 : Ctrl + F10 활용
- ▶ 소스코드에서 어디까지가 정상이고, 어디까지가 비정상인지 구분 가능

만약 오른쪽 소스코드에서 버그가 났다면  
Trace로 어느 범위에서 버그가 났는지  
Ctrl + F10으로 확인 해야 함

```
int main()
{
    int vect[10];

    int x;
    int count = 0;

    for (x = 0; x < 10; x++)
    {
        cin >> vect[x];
    }

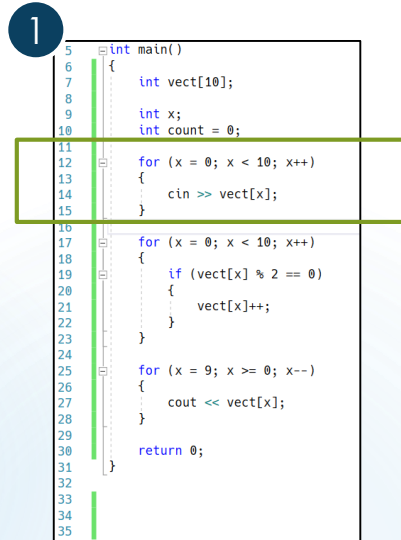
    for (x = 0; x < 10; x++)
    {
        if (vect[x] % 2 == 0)
        {
            vect[x]++;
        }
    }

    for (x = 9; x >= 0; x--)
    {
        cout << vect[x];
    }

    return 0;
}
```

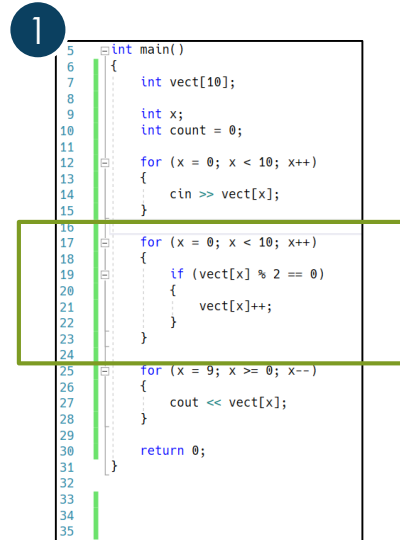
이 3개의 코드 덩어리 중  
어느 범위에서 버그가 났는지  
Ctrl + F10으로 확인해봐야 함

# Ctrl + F10 : 변수 값을 즉시 확인가능2



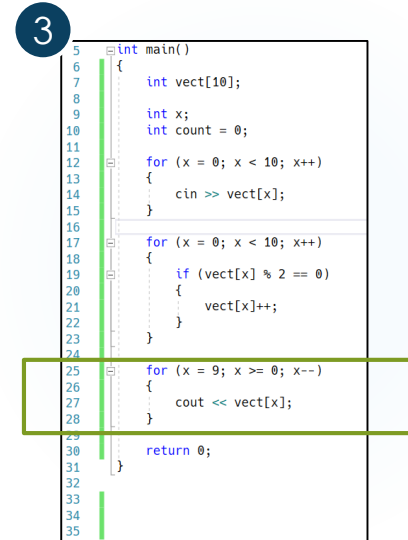
이곳에서 버그가 있는지 확인해보기 위해  
16번 Line 클릭 후 Ctrl + F10

그리고 조사식 vect에  
값이 잘 들어왔는지 확인



이곳에서 버그가 있는지 확인해보기 위해  
24번 Line 클릭 후 Ctrl + F10

그리고 조사식 vect에  
의도대로 값이 다 바뀌었는지 확인



vect값은 잘 채워졌는데  
출력만 잘못된 것인지  
조사식과 콘솔창을 보면서 확인



# Ctrl + F10을 이용한 디버깅

- ▶ 긴 소스코드를 눈으로 하나씩 잘못되는 부분을 찾기에는 시간이 너무 오래 걸린다.
- ▶ **Ctrl + F10을 활용해서 어느 범위에서 버그가 발생하는지 버그의 범위를 점점 좁혀가야 한다.**
- ▶ 버그가 발생하는 범위를 찾아 냈으면 천천히 F10 / F11을 통해 Trace 필요

참고 : 암기해야 할 Trace 단축키

1. F10 : Trace 한줄실행 (함수 안으로 안들어감)
2. F11 : Trace 한줄실행 (함수 안으로 들어감)
3. Shift + F5 : Trace 종료
4. Ctrl + F10 : 커서 위치부터 Trace하기

```
int main()
{
    int vect[10];

    int x;
    int count = 0;

    for (x = 0; x < 10; x++)
    {
        cin >> vect[x];
    }

    for (x = 0; x < 10; x++)
    {
        if (vect[x] % 2 == 0)
        {
            vect[x]++;
        }
    }

    for (x = 9; x >= 0; x--)
    {
        cout << vect[x];
    }

    for (x = 0; x < 10; x++)
    {
        if (vect[x] % 2 == 0)
        {
            vect[x]++;
        }
    }

    for (x = 9; x >= 0; x--)
    {
        cout << vect[x];
    }

    for (x = 0; x < 10; x++)
    {
        if (vect[x] % 2 == 0)
        {
            vect[x]++;
        }
    }

    for (x = 9; x >= 0; x--)
    {
        cout << vect[x];
    }

    return 0;
}
```

# 만약 버그가 발생했다면?

- ▶ 코딩을 하고 난 뒤, 버그가 발생한다면  
눈으로 버그를 찾는 것 보다 Trace를 통해 버그를 찾아주세요.
- ▶ 프로그래머에게는 디버깅 실력이 중요하기 때문에  
충분한 연습이 필요합니다.
- ▶ ctrl + F10과 F10으로  
버그가 어느 범위에서 발생했는지 알아내서 버그를 찾아주세요.

# 2차 배열

- ▶ 1차 배열을 여러 개 만드는 것을 2차배열이라고 함 (=2차원 배열 이라고도 함)
- ▶ 물리 세상의 1차원 2차원 3차원 개념이 아님

배열은 1차 / 2차 / 3차 / 4차 / 5차 ...배열이 존재함

1차배열 예시 : `int arr[7];`

2차배열 예시 : `int arr[5][2];`

3차배열 예시 : `int arr[3][3][2];`

4차배열 예시 : `int arr[2][3][2][4];`

...

1	2	3	4	5
---	---	---	---	---

//1차배열 선언 및 하드코딩

```
int arr[5] = {1, 2, 3, 4, 5};
```

1	2	3	4	5
2	3	4	5	6
3	4	5	6	7

//2차배열 선언 및 하드코딩

```
int arr[3][5] = {{1, 2, 3, 4, 5}, {2, 3, 4, 5, 6}, {3, 4, 5, 6, 7}};
```

# 2차 배열의 좌표 개념

1	2	3	4	5
2	3	4	5	6
3	4	5	6	7

- ▶ 2차원 배열에서 좌표를 표현할 때 (y축, x축) 이렇게 표기 함
- ▶ (0, 0)의 값 : 1
- ▶ (1, 2)의 값 : 4
- ▶ (2, 4)의 값 : 7
- ▶ 좌표는 (0,0) 부터 시작함  
왼쪽 2차 배열에서의 맨 마지막 좌표는 (2,4)

# 2차 배열을 만들고 (2,0)에 값 넣기

- ▶ 다음 소스코드를 이해 해보자

(좌표계에서는 항상 y축이 먼저임을 기억하자 → [y][x])

```
int main( )  
{  
    int vect[3][2] = { 0 };  
    //3x2 배열 만들고, 전체 칸을 0으로 초기화하기  
  
    vect[2][0] = 7;  
  
    return 0;  
}
```



0	0
0	0
7	0

## [어렵지만 필수] 2차 배열에 값 3 채우기

- ▶ 2중 for문을 돌아 배열 전체에 값을 채울 수 있음
- ▶ Trace를 통해 이 소스코드를 이해 해 보자

```
int arr[3][5];
int x, y;

for (y=0; y<3; y++)
{
    for (x=0; x<5; x++)
    {
        arr[y][x] = 3;
    }
}
```

3	3	3	3	3
3	3	3	3	3
3	3	3	3	3

# [어렵지만 필수] 2차 배열에 1부터 증가하는 숫자 채우기

- ▶ Trace를 통해 아래 소스코드를 이해 해 보자

```
int arr[3][5];
int x, y;
int t;

t = 1;
for (y=0; y<3; y++)
{
    for (x=0; x<5; x++)
    {
        arr[y][x] = t;
        t++;
    }
}
```

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15

# [어렵지만 필수] 2차원 배열 모두 출력하기

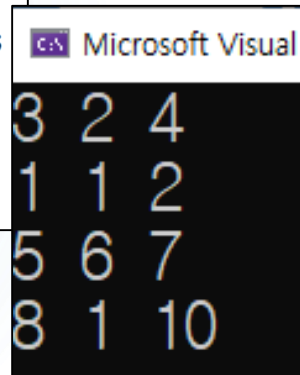
- ▶ 배열 모양 그대로 화면에 출력하는 소스코드를 이해하자

```
int main()
{
    int vect[4][3] =
    {
        {3, 2, 4},
        {1, 1, 2},
        {5, 6, 7},
        {8, 1, 10}
    };

    int x, y;

    for (y = 0; y < 4; y++)
    {
        for (x = 0; x < 3; x++)
        {
            cout << vect[y][x];
        }
        cout << endl;
    }

    return 0;
}
```



Microsoft Visual

```
3 2 4
1 1 2
5 6 7
8 1 10
```

- ▶ 2중 for문이 돌면서 변수 y와 x의 변화는 다음과 같기때문에 배열 모습 그대로 출력이 가능하다

**y, x**

- 00
- 01
- 02
- 10
- 11
- 12
- 20
- 21
- 22