

민선생코딩학원 시작반

---

# 수업노트 LV-04



# 배우는 내용

## 배열 다루기

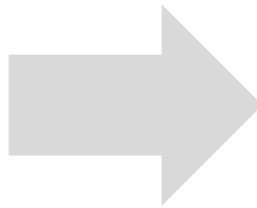
1. 배열의 이해
2. 배열 하드코딩
3. index 개념
4. for문을 활용하여 배열 다루기

# 배열이란?

다량의 변수들을 한꺼번에 만들 수 있는 기능이다.

15개의 변수를 각각 만들기 번거롭다.

```
int a1, a2, a3, a4, a5;  
int b1, b2, b3, b4, b5;  
int c1, c2, c3, c4, c5;
```



편리하게 변수 15개를 만들 수 있다.

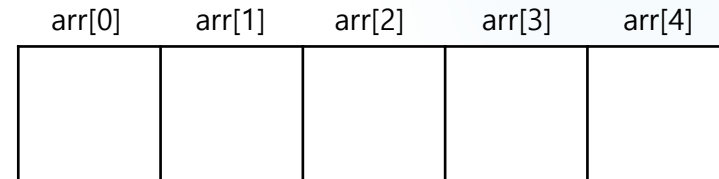
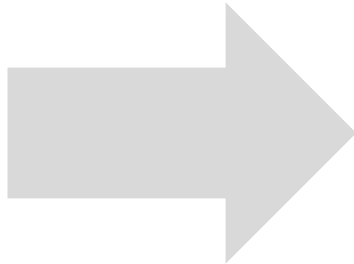
```
int a[15];
```

# 배열을 만드는 방법

배열 이름을 적고, 몇 칸을 만들 지 적어주면 된다.

int arr[5] 는 arr 배열의 0 ~ 4번 칸까지, 총 5칸을 생성하는 것을 의미한다.

```
int arr[5];
```



arr[0] 부터 arr[4] 까지 각 칸이 생성되었다.  
arr 배열의 각 칸들을 **마치 변수처럼** 사용하면 된다.

# 배열 활용 예제

배열의 각 칸들을, 각각의 **변수처럼** 사용한다.

```
int arr[2];

arr[0] = 1;
arr[1] = 2;

cout << arr[0] << " ";

arr[1] = 10;

cout << arr[1];
```

실행결과 : 1 10

숫자 3개를 입력 받고, 합을 출력하는 소스코드

```
int vect[5];

cin >> vect[0];
cin >> vect[1];
cin >> vect[2];

cout << vect[0] + vect[1] + vect[2];
```

# 변수에서 “초기화” 란?

변수를 선언하고, 처음으로 변수에 값을 넣는 것을 “초기화” 라고 한다.

변수 x 선언 후 3으로 초기화

```
int x = 3;
```

배열 3칸 선언 후,  
모든 칸을 0으로 초기화

```
int arr[3];
```

```
arr[0] = 0;
```

```
arr[1] = 0;
```

```
arr[2] = 0;
```

# “하드코딩”이란?

“하드코딩”이란 소스코드 안에 직접 값을 넣는 것을 말한다.

(프로그램 실행하고 키보드로 값을 넣는 것은, 하드코딩이 아니라 **입력**이다. 혼동하지 말자.)

초기화와 하드 코딩은 엄밀히 다른 용어이지만, 같은 의미로 쓰인다. 혼용해서 써도 된다.

- ▶ 초기화 : 변수 선언 후, 처음 값을 세팅
- ▶ 하드 코딩 : 소스코드 안에 값을 미리 넣어두는 것

# 배열 하드코딩하기

arr 배열에 1 ~ 5로 초기화를 시키는 두가지 방법

방법 1

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

방법 2

```
int arr[5];  
  
arr[0] = 1;  
arr[1] = 2;  
arr[2] = 3;  
arr[3] = 4;  
arr[4] = 5;
```

배열을 선언하면, 마치 변수처럼 쓰레기 값으로 채워져 있다.

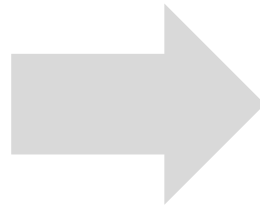


# 배열에서 index 의미

0번 부터 시작하는 **칸 번호**를 의미한다.

아래 vect 배열은 0번 index부터 6번 index까지 사용할 수 있다.

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

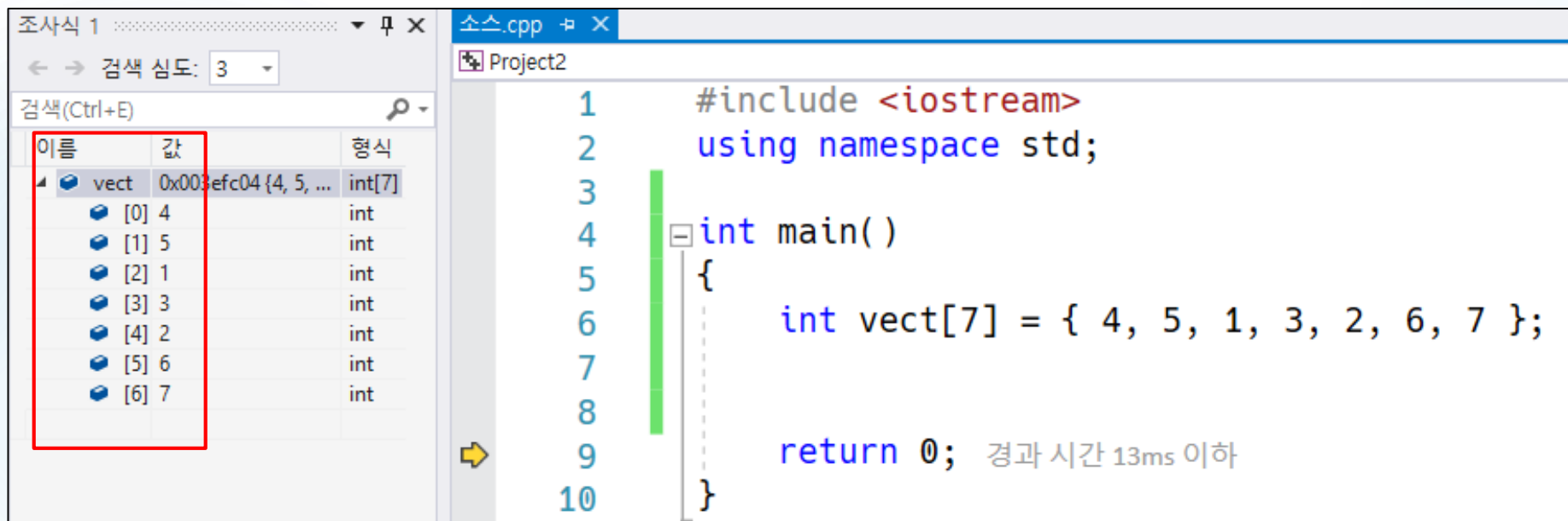


0	1	2	3	4	5	6
4	5	1	3	2	6	7

# 배열 Trace 하기

F10을 눌러 Trace를 시작 해보자.

조사식에 배열 이름을 입력한 뒤에, 배열 값을 확인할 수 있다.



The screenshot displays the Visual Studio IDE during a debug session. On the left, the '조사식 1' (Watch 1) window shows the state of a variable named 'vect'. The variable is of type 'int[7]' and its value is '0x003efc04 {4, 5, ...}'. A red rectangle highlights the array elements: [0] 4, [1] 5, [2] 1, [3] 3, [4] 2, [5] 6, and [6] 7. On the right, the '소스.cpp' (Source.cpp) window shows the C++ code for 'Project2'. The code includes <iostream>, uses namespace std, and defines a main function. Inside main, an array 'vect' of size 7 is initialized with the values {4, 5, 1, 3, 2, 6, 7}. The execution has reached the 'return 0;' statement, with a status message indicating '경과 시간 13ms 이하' (Execution time: 13ms or less). A green vertical bar in the margin indicates the current execution point.

이름	값	형식
vect	0x003efc04 {4, 5, ...}	int[7]
[0]	4	int
[1]	5	int
[2]	1	int
[3]	3	int
[4]	2	int
[5]	6	int
[6]	7	int

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int vect[7] = { 4, 5, 1, 3, 2, 6, 7 };
7
8
9      return 0; 경과 시간 13ms 이하
10 }
```

# index 를 활용한 문제 풀이

숫자 하나를 입력 받고, 그 Index 에 해당하는 칸에 100 넣는다.

```
#include <iostream>
using namespace std;
int main()
{
    int a;
    int vect[7] = { 4, 5, 1, 3, 2, 6, 7 };

    cin >> a;

    vect[a] = 100;

    return 0;
}
```

# 배열 모든 칸을 7로 채우기

for 문을 활용하면, 손쉽게 배열에 7을 채워 넣을 수 있다.

```
int vect[10];  
  
int x;  
  
for (x = 0; x < 10; x++)  
{  
    vect[x] = 7;  
}
```

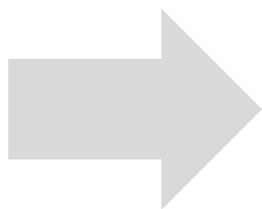
for 문이 반복되면서, x가 0, 1, 2, ..., 9 값으로 바뀐다.  
따라서 vect[0], vect[1], vect[2] .... vect[9] 에 각각 7이 들어간다.

# 배열에 증가되는 숫자 넣기

for 문으로 배열에 값을 넣는 코드를 반복 시킨다.

for 문을 이용하여, 각 배열 칸에 0 부터 4까지 값을 넣을 수 있다.

```
int vect[5];  
  
int x;  
  
for (x = 0; x <= 4; x++)  
{  
    vect[x] = x;  
}
```

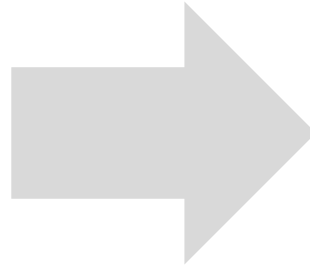


0	1	2	3	4
<b>0</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>

# 배열의 모든 칸을 0 으로 초기화

for 문으로 모든 칸을 0으로 초기화 한다.

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



더 간단한 방법!

```
int arr[5] = { 0 };
```

간단한 방법으로는 0 이외 숫자로는 초기화 할 수 없다.  
다른 숫자로 초기화 하려면 왼쪽처럼 for 문을 돌려야 한다.

# 배열 하드코딩 할 때 주의사항 1

여러 개 값을 동시에 하드코딩 할 때는 **배열 선언 할 때만 가능하다.**

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

```
int arr[5];
```

```
arr = {1, 2, 3, 4, 5}; // 버그, 불가능한 문법
```

```
int arr[5];
```

```
arr[5] = {1, 2, 3, 4, 5}; // 버그, 불가능 문법
```

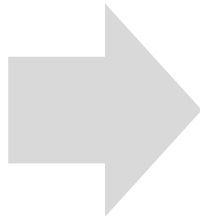
배열 선언이 끝나면 {} 로 여러 값을 넣는 것은 불가능하다.

선언이 끝난 후에는 값 하나씩 넣어야 한다.

# 배열 하드코딩 할 때 주의사항 2

만약, 선언 후 값을 넣고 싶다면?

```
int bbq[5];  
bbq = {1, 2, 3, 4, 5}; // 버그, 불가능한 문법  
bbq[5] = {1, 2, 3, 4, 5}; // 버그, 불가능 문법
```



```
int bbq[5];  
  
bbq[0] = 1;  
bbq[1] = 2;  
bbq[2] = 3;  
bbq[3] = 4;  
bbq[4] = 5;
```

OR

```
int bbq[5];  
int x;  
  
for (x=0; x<5; x++)  
{  
    bbq[x] = x + 1;  
}
```



# 배열에 있는 값을 모두 출력 하기

하드 코딩 되어있는 배열을 for 문으로 출력한다.

```
int vect[5] = { 3, 1, 5, 1, 2 };  
  
int x;  
  
for (x = 0; x < 5; x++)  
{  
    cout << vect[x];  
}
```

맨 뒤에 있는 숫자부터 for 문으로 거꾸로 출력한다.  
4번 index가 가장 마지막 칸임을 유의하자.

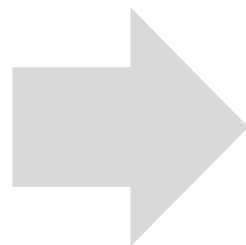
```
int vect[5] = { 3, 1, 5, 1, 2 };  
  
int x;  
  
for (x = 4; x >= 0; x--)  
{  
    cout << vect[x];  
}
```

# 응용 문제

소스코드를 이해 해 보자.

[문제]

배열 일곱 칸에 숫자 7개 입력 받는다.  
배열의 모든 칸에 1씩 더한다.  
이후, 모든 배열의 값을 출력한다.



```
int vect[7];  
int x;  
  
for (x = 0; x < 7; x++)  
{  
    cin >> vect[x];  
}  
  
for (x = 0; x < 7; x++)  
{  
    vect[x]++;  
}  
  
for (x = 0; x < 7; x++)  
{  
    cout << vect[x];  
}
```

코딩은 한 가지 방법이 있는 것이 아니라 여러가지 방법이 있다.  
위 코드는 한가지 예시 일 뿐이다.

# Boss 문제 : Sum 구하기

배열에 모든 숫자들을 모두 더하자.

sum 변수 하나 만든다.

for 문을 돌려 배열의 모든 값을 누적하여 더한다.

```
int num[4] = { 3, -1, 5, 12 };  
int sum;  
int x;  
  
sum = 0; //0부터 시작  
  
for (x = 0; x < 4; x++)  
{  
    //sum에 누적해서 숫자를 더한다  
    sum += num[x];  
}  
  
cout << sum;
```

sum = sum + num[x] 를 줄여서  
sum += num[x] 로 표현할 수 있다