



K-Digital Training 스마트 팩토리 4기

배열

배열 ??

- 한 번에 여러 개의 값들을 저장할 수 있는 변수
- 배열 변수를 이용하면 비슷한 값들을 쉽게 처리할 수 있음.
- ex) 친구들 이름을 저장한다고 했을 때,
friend1, friend2, 이렇게 변수를 여러 개 만들지 않고,
friend 라는 하나의 변수 안에 여러 명의 정보를 저장!

배열의 선언 및 초기화

방법 1 – 선언과 초기화가 분리되어 있는 경우

// 배열의 선언

```
std::string fruit[3];
```

//배열의 초기화

```
fruit[0] = "apple";
```

```
fruit[1] = "banana";
```

```
fruit[2] = "orange";
```

배열의 선언 및 초기화

방법 2 – 선언과 초기화를 동시에 하는 경우

// 배열의 선언 & 초기화

```
std::string fruit[] = { "apple", "banana", "orange" };
```

배열의 인덱싱

인덱스 : 배열 안 원소의 순서. [숫자] 형태 사용.

주의 📌 인덱스는 1이 아닌 **0부터 시작**

예시)

```
std::string fruit[] = { "apple", "banana", "orange" };
```

```
std::cout << fruit[1] << std::endl; // "apple"이 아닌 "banana"가 출력됨
```

다차원 배열

```
std::string fruit[2][2] = {  
    { "apple", "banana" },  
    { "orange", "strawberry" }  
};
```

```
std::cout << fruit[0][0] << std::endl; //apple 출력  
std::cout << fruit[0][1] << std::endl; //banana 출력  
std::cout << fruit[1][0] << std::endl; //orange 출력  
std::cout << fruit[1][1] << std::endl; //strawberry 출력
```

배열과 for문

```
std::string fruit[3] = { "apple", "banana", "orange" };  
  
for (int i = 0; i < sizeof(fruit) / sizeof(fruit[0]); i++) {  
    std::cout << fruit[i] << std::endl;  
}
```

* sizeof : 자료형 또는 변수의 크기를 byte단위로 반환

배열과 for-each문

```
std::string fruit[3] = { "apple", "banana", "orange" };
```

```
for (std::string fr : fruit) {  
    std::cout << fr << std::endl;  
}
```

**** 문법 ****

```
for ( 자료형 변수이름 : 배열 ) { }
```

실습1 배열 사용해보기

- (1) city라는 배열 변수를 만들고, 5개의 나라 이름을 값으로 지정
- (2) for문 혹은 for-each문을 활용하여 city 안의 모든 원소를 출력

실습2 배열 사용해보기

- (1) city2 라는 배열 변수를 선언하고, 5개의 나라 이름은 사용자가 콘솔에 입력한 값으로 지정
- (2) for문 혹은 for-each문을 활용하여 city2의 모든 원소 출력

실습3 배열 활용하기

(1) grade 라는 배열 변수를 선언하고, 학생 5인의 성적 입력받기

(2) for문 혹은 for-each문을 활용하여 학생 5인의 성적 평균 산출

```
1번 학생의 성적을 입력하세요: 10
2번 학생의 성적을 입력하세요: 20
3번 학생의 성적을 입력하세요: 30
4번 학생의 성적을 입력하세요: 40
5번 학생의 성적을 입력하세요: 58
성적 평균 : 31.6 output input
```

물론! 배열에 저장하지 않아도 위 기능 구현 가능하나,
연습을 위해 입력 받은 수들을 배열에 한번 저장하고 평균 산출해보기

동적 배열과 포인터

동적 배열 ??

지금까지의 배열은 선언할 때 고정된 크기를 지정해야 했음!

하지만, 배열의 크기를 나중에 정하고 싶다면?

➡ 이럴 때 사용하는 것이 **동적 배열**

단, 동적 배열을 익히기 위해선 **포인터**의 개념을 먼저 알아야 함

포인터 ??

포인터 : 메모리의 주소를 가진 변수

포인터 변수는 * 을 이용하여 선언

```
int *p;
```

```
int n = 10;
```

```
p = &n; // p에 n의 메모리 주소를 저장함.
```

포인터 문법

```
int n = 10;  
int *p = &n;  
std::cout << p << std::endl;  
// p에는 n의 주소값이 저장되어 있음. => 00000000 형태의 주소값 출력  
std::cout << *p << std::endl;  
// *p는 주소값에 저장되어 있는 실제 값에 접근 => 10 출력
```

생각해보기

- (1) 만약 n의 값을 20으로 다시 할당 한다면, p와 *p의 값은 어떻게 될까?
- (2) p를 이용하여 n의 값을 변경할 수 있을까? 어떻게 하면 될까?

동적 배열 – 선언과 할당

동적 배열을 만들기 위해선 **포인터 변수**와 **new** 키워드 사용.

```
int n2;  
std::cout << "숫자를 입력하세요: ";  
std::cin >> n2;
```

```
int *arr = new int[n2];  
// 1. 동적 메모리를 가리키는 포인터 선언  
// 2. new라는 키워드를 사용하여, 동적 배열 할당.
```

동적 배열 – 사용과 해제(반납)

동적 배열을 해제(반납)하기 위해선 **delete** 키워드 사용.

```
int *arr = new int[n2]; // 동적 배열 선언 및 할당
```

```
for (int i = 0; i < n2; i++) {  
    arr[i] = i+1; // 동적 배열 사용  
}
```

```
delete[] arr; // 동적 배열 해제(반납). 동적 메모리는 사용 후 꼭 해제하기
```

2차원 동적 배열

```
int n3;  
std::cout << "숫자를 입력하세요: ";  
std::cin >> n3;
```

// 동적 배열 선언 & 할당

```
int **arr2 = new int *[n3];
```

```
for (int i = 0; i < n3; i++) {  
    arr2[i] = new int[n3];  
}
```

```
for (int i = 0; i < n3; i++) {  
    for (int j = 0; j < n3; j++) {  
        arr2[i][j] = 0; // 동적 배열 사용  
    }  
}
```

// 동적 배열 해제(반납)

```
for (int i = 0; i < n3; i++){  
    delete[] arr2[i];  
}  
delete[] arr2;
```

실습1 동적 배열 사용해보기

- (1) 사용자로부터 x, y 2개의 자연수를 입력 받기
- (2) 만약 사용자가 x 와 y 에 0 이나 음수를 입력한다면 에러 메시지를 출력 후 다시 입력 받기.
- (3) $x * y$ 의 크기를 갖는 이차원 동적 배열 `arr` 을 선언
- (4) `arr[0][0]` 부터 `arr[x][y]` 까지 순서대로 1 부터 $x * y$ 저장
- (5) `arr[0][0]` 부터 `arr[x][y]` 까지 저장된 값 출력

```
x를 입력하세요: 6
y를 입력하세요: -1
x와 y모두 양수를 입력해주세요.
x를 입력하세요: 3
y를 입력하세요: 4
```

```
x를 입력하세요: 3
y를 입력하세요: 4
1 2 3 4
5 6 7 8
9 10 11 12
```

input

output

[번외] 실습. 배열 실습2 응용!

- (1) 몇명의 학생 성적 평균을 구할 지 입력 받기
- (2) 입력받은 학생 수 만큼 성적을 입력받기
- (3) 학생들의 성적 평균 산출하기

```
1번 학생의 성적을 입력하세요: 10
2번 학생의 성적을 입력하세요: 20
3번 학생의 성적을 입력하세요: 30
4번 학생의 성적을 입력하세요: 40
5번 학생의 성적을 입력하세요: 58
성적 평균 : 31.6
```

물론! 배열에 저장하지 않아도 위 기능 구현 가능하나,
동적 배열 연습을 위해 입력 받은 수들을 배열에 한번 저장하고 평균 산출해
보기

vector

vector ??

- 자동으로 메모리를 할당해주는 배열
- 배열은 한 번 크기가 정해지면 그 크기가 고정되어 바뀌지 X
- vector를 사용하면 포인터를 쓰지 않아도 크기를 추후에 지정할 수 있음.
- 뿐만 아니라, 중간에 배열의 크기를 바꿀 수도 있음!

vector 사용하기

```
#include <vector> // vector 헤더파일을 추가해야 사용 가능
```

```
vector<int> v = { 1,2,3,4,5 };
```

```
vector<int> v(4); //int형 벡터 생성 후 크기를 4로 할당(모든 벡터요소 0으로 초기화)
```

```
vector<int> v(5, 1); //int형 벡터 생성 후 크기를 5로 할당(모든 벡터요소 1로 초기화)
```

```
v.assign(5, 1); //0~4인덱스의 값을 1로 초기화
```


vector 사용하기

- `.at(인덱스)` // 인덱스에 해당하는 값 반환
- `.front()` // 벡터의 첫번째 요소 접근
- `.back()` // 벡터의 마지막 요소 접근
- `.size();` // `v`의 길이 반환

vector 사용하기

- `.push_back(원소)` //배열의 제일 마지막에 원소를 삽입
- `.pop_back()` //백터의 마지막 부분 제거
- `.begin()` // 백터 시작점의 iterator 반환
- `.insert(v.begin()+3, 원소);` //3번 인덱스에 원소를 삽입

vector 사용하기

- `.erase(시작 iterator, 마지막 iterator)` // 시작 위치부터 마지막 위치 전까지의 원소를 삭제

`v.erase(v.begin() + 1, v.begin() + 3);` // 1 ~ 2번 인덱스 삭제

- `.clear()` // 벡터의 모든 요소를 지움

실습1 vector 사용해보기

(1) 벡터를 이용하여 이전 실습 해결해보기

```
x를 입력하세요: 6
y를 입력하세요: -1
x와 y 모두 양수를 입력해주세요.
x를 입력하세요: 3
y를 입력하세요: 4
```

```
x를 입력하세요: 3
y를 입력하세요: 4
1 2 3 4
5 6 7 8
9 10 11 12
```

input

output

(번외) 실습 vector 사용해보기

(1) 중복 요소 삭제

```
std::vector<int> myVector = {10, 20, 30, 20, 40, 10, 50};
```

결과 : {10, 20, 30, 40, 50};

Array와 Vector의 비교

	Array	Vector
타입	기본적 자료구조	STL 컨테이너
메모리	선언시 메모리 고정됨	메모리 재할당 용이
접근방식	Index기반 랜덤접근	Iterator 기반 랜덤 접근
접근속도	빠름	느림(포인터를 통한 접근)

* 랜덤접근 (Random Access)

무작위가 아닌 어디로든 똑같은 시간으로 접근 가능하다는 의미로서, 메모리의 주소만 알고 그곳을 지정하면 별다른 지연 없이 바로 접근할 수 있다는 의미

(2차원 배열 복습) 행렬 합 구하기

입력 :

```
행과 열의 수를 입력하세요: 3 3
행렬 원소를 입력하세요:
1 2 3
4 5 6
7 8 9
```

결과 :

```
각 행의 합:
행 1: 6
행 2: 15
행 3: 24

각 열의 합:
열 1: 12
열 2: 15
열 3: 18
```

(vector 복습) vector 조작하기

1. Vector를 사용하여 정수를 저장하는 빈 벡터 선언
2. 사용자로부터 5개의 정수를 입력받아 벡터에 추가
3. 벡터의 크기 출력
4. 벡터의 모든 원소 출력
5. 가장 큰 값을 찾아 출력
6. 벡터의 모든 원소를 두배로
7. 인덱스를 입력받아 해당 인덱스에 있는 원소 제거
8. 인덱스를 입력받아 해당 인덱스에 있는 새로운 원소 삽입