K-Digital Training 스마트 팩토리 3기

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 ) { }
```

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 포인터의 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 n 의 값을 20으로 다시 할당 한다면, p 와 *p 의 값은 어떻게 될까?

(2) p를 이용하여 n의 값을 변경할 수 있을까? 어떻게 하면 될까?

동적 배열 – 선언과 할당

동적 배열을 만들기 위해선 포인터 변수와 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;
```

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 백터 생성 후 크기를 4로 할당(모든 백터요소 0으로 초기화)
vector<int> v(5, 1); //int형 백터 생성 후 크기를 5로 할당(모든 백터요소 1로 초기화)

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

vector 사용하기

- .at(인덱스) // 인덱스에 해당하는 값 반환
- .front() // 백터의 첫번째 요소 접근
- .back() // 백터의 마지막 요소 접근
- .size(); //v의 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() // 백터의 모든 요소를 지움