K-Digital Training 스마트 팩토리 3기

반복문

반복문 ??

똑같은 명령을 일정 횟수만큼 반복해 수행하도록 하는 실행문

for

while

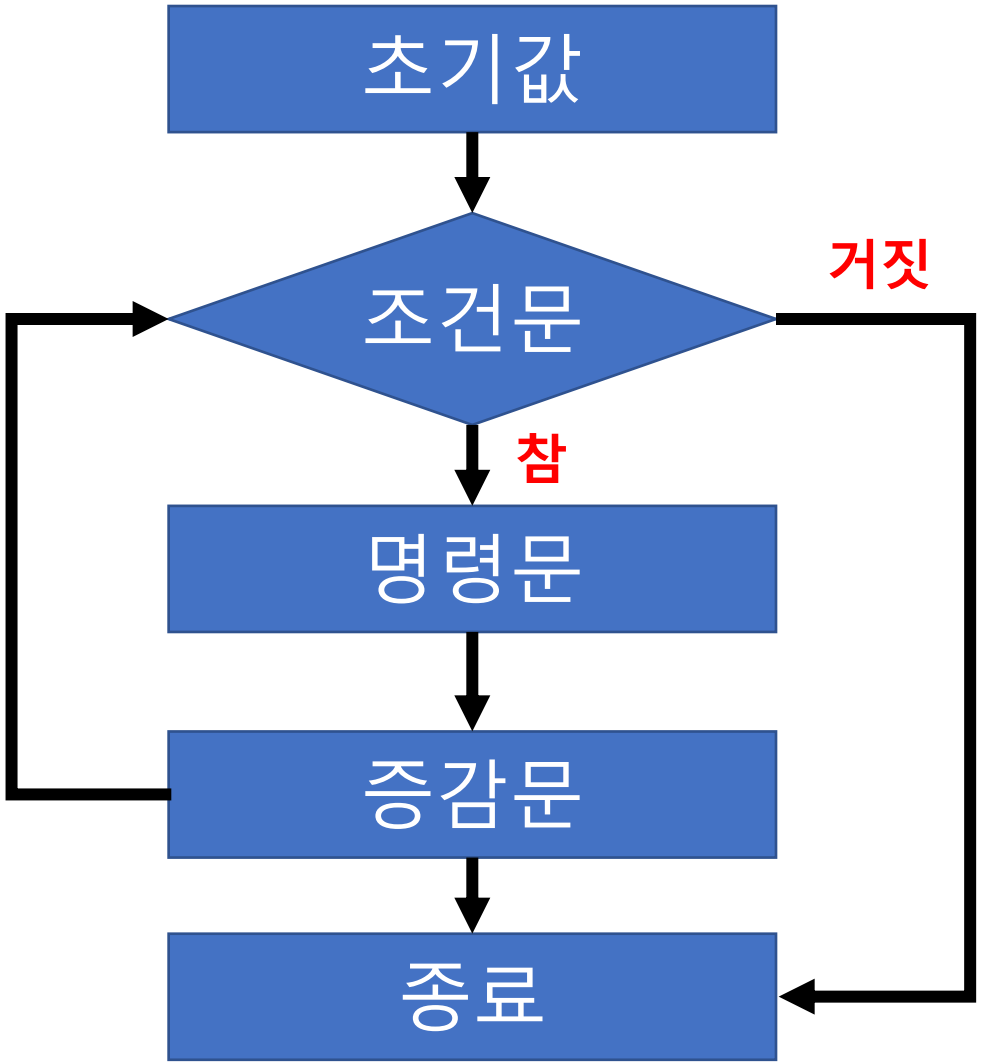
for-each

do / while

기본 For문

```
for ( 초기값; 조건문; 증감문 ){
        // 조건문의 결과가 참인 경우 실행
        // 실행할 코드
}
```

```cpp
for (i = 1; i <= 5; i++) {
    std::cout << "Iteration " << i << std::endl;
}
```



For-each문

배열의 원소에 대한 반복을 수행할 때 사용함.

(추후 배열에 대해 배운 후 알아볼 예정!)

While문

while (조건문) {

// 조건문이 참인 경우 반복적으로 실행하는 명령문

}

```
i = 1;  // Reset i
while (i <= 5) {
    std::cout << "Iteration " << i << std::endl;
    i++;
}
```

※ 주의사항

- while문의 경우 명령문에서 조건문의 결과를 바꾸지 않으면 무한 루프에 빠질 수 있다.

do/while문

do {

	// 조건문이 참인 경우 반복적으로 실행하는 명령문

	// 단, 처음 한 번은 무조건 실행됨!

}

while (조건문);

```
do {
    std::cout << "Iteration " << i << std::endl;
    i++;
} while (i <= 5);
```

함수

함수 ??

어떤 일을 수행하는 코드의 묶음. 즉, 기능을 따로 빼서 묶는 것

함수를 사용하는 이유 ??

- 필요할 때마다 호출이 가능하다.
  - 반복적으로 수행해야 하는 기능을 한번 만들어 놓으면 필요할 때 마다 호출해서 사용할 수 있음.

- 논리적인 단위로 분할이 가능하다.
  - 코드를 기능에 따라 나눠서 볼 수 있음.
  - 코드를 분석할 때 함수로 구분이 되어있으면 분석이 쉬워짐.

함수 문법

```
리턴 타입 함수 이름( 인수 목록 )
{
        // 함수의 본문
}
```

함수 문법 – 리턴 타입

```
리턴 타입 함수 이름( 매개 변수 )
{
        // 함수의 본문
}
```

리턴 타입 : 이 함수가 결과로 어떤 유형의 값을 리턴(반환)할 지 선언

함수 문법 – 함수 이름

```
리턴 타입 함수 이름( 매개 변수 )
{
        // 함수의 본문
}
```

함수 이름 : 함수의 이름을 결정. 추후 호출 할 때 사용될 이름.

* 함수의 이름은 상대방이 이해하기 쉽도록 합리적으로 작성
    ex) 숫자들의 합을 구하는 함수를 작성 할 땐,
        aaa → X , sum → O

함수 문법 – 매개 변수

```
리턴 타입 함수 이름( 매개 변수 )
{
        // 함수의 본문
}
```

매개 변수 : 함수를 호출할 때 전달된 값을 함수 내부에서 사용할 수 있게 해주는 변수

함수 문법

```
리턴 타입 함수 이름( 매개 변수 ) {
        // 함수의 본문
}
```

함수의 선언

```cpp
int funcEx_1() {
    return 10;
}

std::string funcEx_2() {
    return "hello";
}

void funcEx_3() {
    std::cout << "리턴 타입이 void인 함수" << std::endl;
}
```

함수의 호출

```cpp
int a = funcEx_1();
std::cout << a << std::endl;

std::string str = funcEx_2();
std::cout << str << std::endl;

funcEx_3();
```

함수 문법

```
리턴 타입 함수 이름( 매개 변수 ) {
        // 함수의 본문
}
```

함수의 선언 (매개변수 有)

```
int funcEx_sum(int n1, int n2) {
    return n1 + n2;
}
```

함수의 호출

```
std::cout << funcEx_sum(2,3) << std::endl;
```