Chapter 04 - 실행 흐름 제어

생성일	@2025년 9월 16일 오전 10:02
태그	

01. 조건문으로 흐름 제어

if 문으로 분기하기

```
if (조건식)
{
    // 실행문
}
```

- 조건이 참이면 중괄호 안의 코드 블럭이 실행, 조건이 거짓이면 코드 블록은 무시

else문을 포함하는 if문

```
if (조건식)
{
    // 실행문 1
}
else
{
    // 실행문 2
}
```

if문에 중괄호 생략

```
#include <iostream>
using namespace std;
```

```
int main()
{
    int input_number;

    cout << "정수 입력 : ";
    cin >> input_number;

    if (input_number > 0)
    {
        cout << "입력한 수는 양수 입니다." << endl;
    }
    else if (input_number < 0)
    {
        cout << "입력한 수는 음수 입니다." << endl;
    }
    else
    {
        cout << "입력한 수는 0 입니다." << endl;
    }

    return 0;
}
```

switch 문으로 분기하기

```
#include <iostream>
using namespace std;

int main()
{
    int input_number;

    cout << "1~5 정수 입력: ";
    cin >> input_number;

    switch (input_number)
```

```
    {
    case 1:
        cout << "입력한 수는 1 입니다." << endl;
        break;
    case 2:
        cout << "입력한 수는 2 입니다." << endl;
        break;
    case 3:
        cout << "입력한 수는 3 입니다." << endl;
        break;
    case 4:
        cout << "입력한 수는 4 입니다." << endl;
        break;
    case 5:
        cout << "입력한 수는 5 입니다." << endl;
        break;
    default:
        cout << "입력한 수는 1 ~ 5 범위 밖 입니다." << endl;
        break;
    }

    return 0;
}
```

case 문에서 break 문 생략

```
#include <iostream>
using namespace std;

int main()
{
    int input_number;

    cout << "1~5 정수 입력: ";
    cin >> input_number;

    switch (input_number)
```

```
    {
    case 1:
        cout << "입력한 수는 1 입니다." << endl;
    case 2:
        cout << "입력한 수는 2 입니다." << endl;
    case 3:
        cout << "입력한 수는 3 입니다." << endl;
    case 4:
        cout << "입력한 수는 4 입니다." << endl;
    case 5:
        cout << "입력한 수는 5 입니다." << endl;
    default:
        cout << "입력한 수는 1 ~ 5 범위 밖 입니다." << endl;
    }

    return 0;
}
```

- break문을 생략하면 전체가 차례대로 출력됨

while 문으로 반복하기

```
#include <iostream>
using namespace std;

int main()
{
    int count = 0;
    while (count < 5) {
        cout << "cout : " << count << endl;
        count++;
    }
    return 0;
}
```

do ~ while 문의 동작 방식

- while문처럼 조건이 참인 동안 코드 블록을 반복

  → 무조건 한 번은 코드 블록이 실행됨.

```
#include <iostream>
using namespace std;

int main()
{
    {
        int i = 0;
        while (i < 0) { // 조건식이 거짓이므로 반복문은 실행 안됨
            cout << "i is less than 0" << endl;
            i++;
        }
    }
    int j = 0;
    do {
        cout << "i is less than 0" << endl;
        j++;
    } while (j < 0); // 조건식이 거짓이지만 반복문은 1회 실행된
다.

    return 0;
}
```

for 문으로 반복하기

- 일정 횟수 이하를 정확히 반복하고자 할 때 사용
- 형식

  ```
  for (1. 초기화; 2. 조건식; 3. 증감식) {
  // 반복 실행 코드
  }
  ```

```
#include <iostream>
using namespace std;
```

```
int main()
{
    for (int count = 0; count < 5; count++) {
        cout << "count : " << count << endl;
    }
    return 0;
}
```

- for문의 장점
  - 간결한 구문
  - 더 나은 반복 제어
  - 배열 같은 데이터 구조에서 사용하기 쉬움
  - 특정 횟수만큼 실행되는 반복을 더 쉽게 만들 수 있음
  - 무한 반복 방지
  - 초기화, 조건식, 증감식을 한 곳에서 파악할 수 있음

- for문 : 반복 횟수를 미리 알 수 있고, 배열을 순회&특정 범위의 값에 연속해서 접근할 때
- while문 : 반복 횟수를 미리 알 수 없고 특정 조건이 충족되는 한 계속 반복해야 할 때
- break : 반복문 종료
- continue : 반복문 계속 진행

03. 표현식과 구문의 차이

- 표현식 : 하나 이상의 변수, 연산자, 리터럴을 조합해 값을 평가하고 결과를 반환
  - 수학에서 수식과 같음, 항상 결괏값이 나옴
- 구문 : 하나 이상의 연산, 동작을 실행하는 명령문의 집합, 값을 할당하거나 실행 흐름을 제어
  - 여러 표현식을 포함할 수 있으며, 항상 ;로 끝남
  - 컴파일러가 실행할 수 있는 최소의 독립적인 코드

→ 한 개 이상의 표현식과 키워드를 포함함

→ **표현식** : 값을 생성하거나 계산하는 요소,
구문 : 프로그램의 실행 흐름을 제어하거나 작업을 실행하는 명령문의 집합