# 6장 계속 갈래길 앞에서 – 조건문과 순환문

**학습 목표**

C 언어 프로그램의 기본 구조를 알고 있고, 실행되는 순서를 확인할 수 있다.

지금 나의 상태는 \_\_\_\_점이고 이 장을 학습하고 난 뒤 나의 목표는 \_\_\_\_점이다.

1 전혀 모른다

2 이름만 들어봤다

3 어렴풋이 안다

4 설명하기는 어렵지만 사용할 수 있다

5 무엇인지 정확히 알고 설명할 수 있다

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

구문statement이란 표현식과 달리 어떤 동작을 수행하는 문장입니다. 지금까지 작성한 코드 구문은 위에서 아래로 진행하면서 일련의 구문을 실행합니다. 하지만 조건과 상황에 따라 다른 구문을 실행하고 싶을 수도 있고, 특정 구문 실행을 반복적으로 수행하고 싶은 경우가 있습니다. 이를 실행 흐름 제어control of execution flow이라고 부르며, 여기에는 조건문과 순환문으로 나눌 수 있습니다.

조건문은 분기문이라고도 불리는데,분기分岐란 갈림길이 나뉘어진다는 뜻입니다. 따라서 조건에 따라 실행 흐름이 달라지는 프로그래밍의 구조입니다. 주로 if 구문으로 구현되며, 동일한 변수의 여러 값을 비교해야 하는 경우에는 switch 구문을 사용합니다.

순환문은 반복문이라고도 불리는데, 반복해서 특정 동작을 실행한다는 뜻입니다. while, for, do-while 구문으로 구현됩니다. 덧붙여 break, continue를 사용하여 논리상 추가 부담을 줄일 수 있습니다.

goto 문도 실행 흐름 제어에 사용될 수 있습니다. 하지만 코드의 실행 흐름이 직관적이지 않고 디버깅 및 유지 보수가 어려운 코드를 만들어 내기 때문에 사용하지 않아야 하는 안티 패턴으로 간주됩니다.

이번 장에서는 실행 흐름 제어를 주제로 위에서 소개한 각 제어 구문을 하나씩 살펴봅니다.

## 6.1 if 구문

만약 특정 조건을 만족할 때만 수행하고 싶은 코드가 있다면 if를 사용합니다. if에는 조건을 만족하는지 판단하기 위하여 불린 표현식이 필요합니다. 불린 표현식은 괄호로 감싸 줍니다. if 구문 다음에는 이 조건을 만족할 경우 실행할 구문이 이어집니다.

실행해야할 구문이 여러 개라면 반드시 중괄호로 감싸주어야 합니다. 중괄호속의 각 구문은 세미콜론으로 끝나야 합니다. 중괄호로 감싼 코드를 코드 블록code block이라 부르기도 합니다.

위의 설명을 코드로 표현하면 다음과 같습니다.

1. if (조건식) /\* 세미콜론을 붙이지 않습니다. \*/
2. 참일때\_실행할\_구문;
3. if (조건식)
4. {
5. 참일때\_실행할\_구문\_1;
6. 참일때\_실행할\_구문\_2;
7. }

코드는 늘 살아숨쉬며 변화합니다. 코드를 작성할 때에는 참일 때 실행할 구문이 한 개여서 중괄호를 하지 않아도 되지만, 나중에 변경될 수 있으니 중괄호를 사용하는 것이 좋다고 알려져 있습니다. 물론 의도가 명확한 상황에서는 중괄호를 사용하지 않아도 됩니다.

조건식 괄호 뒤에는 세미콜론을 붙이지 않아요!

조건식 뒤에는 세미콜론이 없습니다! C 언어의 문법상 세미콜론만 있어도 하나의 구문이 됩니다. 따라서 if 구문뒤에 세미콜론이 붙게 되면 조건식은 아무일도 하지 않는 빈 구문을 실행합니다. 따라서 세미콜론이 붙은 아래 구문은 무조건 실행되므로 if 조건문의 의미가 사라집니다. 초보일 때 하게 되는 실수 중 하나입니다.

그럼 실제로 if 구문을 사용해 보겠습니다. 간단한 예제로 변수에 저장된 값이 3의 배수라면 별표를 표시해 봅시다. 3의 배수는 3으로 나누었을 때 나머지가 0이며, 우리는 이미 나머지를 구하는 연산자의 사용법을 알고 있습니다.

1. #include <stdio.h>
2. int main()
3. {
4. int val;
5. printf(“Enter the number:\n”);
6. scanf(“%d”, &val);
7. if ((val % 3) == 0)
8. {
9. printf(“\*”);
10. }
11. }

Enter the number:

2

Enter the number:

3

\*

줄번호 \*부터 \*까지는 프로그램을 실행했을 때 변수 val의 값을 사용자가 입력할 수 있도록 만들어줍니다. 입출력에 관한 함수의 사용법은 \*장에서 좀 더 상세히 살펴봅니다.

컴파일할 때 오류가 발생합니다!

위 코드를 컴파일하면 비주얼 C 컴파일러에서는 보안 경고(C4996 경고)가 발생합니다. scanf와 같은 C표준 함수에서 입력데이터의 길이나 범위에 대하여 보안 문제가 발생할 수 있기 때문입니다. 경고 메시지를 보면 scanf\_s함수를 사용하거나 \_CRT\_SECURE\_NO\_WARNINGS를 정의하라고 안내하고 있습니다.

scanf\_s 함수는 마이크로소프트 비주얼C 컴파일러에서 제공하는 보안 강화 함수로, 다른 컴파일러를 사용할 경우에는 지원되지 않는 함수이므로 컴파일 오류가 발생합니다. 정수와 실수를 입력받을 때에는 scanf함수와 똑같이 사용할 수 있습니다.

이 경고를 해소하려면 크게 두가지 방법, 경고를 무시하도록 컴파일러 옵션을 설정하거나 경고를 받아들여 보안이 강화된 입출력함수를 사용하는 방법이 있습니다. 이것도 조건문이네요.

경고를 무시하도록 컴파일러 옵션을 설정하려면 \_CRT\_SECURE\_NO\_WARNINGS를 정의해 주거나 프로젝트 속성에 추가하면 됩니다.

코드 수준에서 컴파일러 옵션을 설정하려면 파일의 맨 처음에 다음 코드를 추가합니다.

#define \_CRT\_SECURE\_NO\_WARNINGS

그러면 경고는 사라집니다. 다만 이 옵션을 설정해야하는 모든 파일에 위 코드를 추가해 주어야 합니다. 또는 프로젝트 속성에서 C/C++의 SDL 검사 속성을 아니오(/sdl-)로 선택할 수 있습니다. 프로젝트 속성이므로 프로젝트에 포함된 모든 파일에 동일한 컴파일러 옵션이 적용됩니다.

텍스트, 스크린샷, 디스플레이, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

일관된 스타일 유지가 중요합니다!

중괄호로 감쌀 때 코드 작성 관례에 대해 크게 두가지 주장이 있습니다. 어떤 것을 선택하든 코드의 동작은 동일하며, 코드 작성에서는 일관성있게 사용하는 것이 중요합니다. 팀을 이뤄 개발중이라면 일관성있게 코드를 작성하는 것이 가독성을 향상시키는데 도움이 됩니다.

1. /\* 1번 스타일 \*/
2. if (조건식) {
3. 구문\_1;
4. 구문\_2;
5. }
6. /\* 2번 스타일\*/
7. if (조건식)
8. {
9. 구문\_1;
10. 구문\_2;
11. }

이전 코드를 확장해 보겠습니다. 이번에는 3의 배수이면 \*를 표시하고, 3의 배수가 아니면 값을 그대로 출력해 보겠습니다. % 연산자의 결과값이 0이 아니면 값을 출력하는 조건 구문을 추가해 봅니다.

1. #include <stdio.h>
2. int main()
3. {
4. int val;
5. printf(“Enter the number:\n”);
6. scanf(“%d”, &val);
7. if ((val % 3) == 0)
8. printf(“\* ”);
9. if ( (val % 3) != 0)
10. print(“%d “, val);
11. }

원하는대로 잘 동작하나요? 그런데 한번 더 생각해 봅시다. 첫번째 if 문이 참이면 \*를 표시하는 구문을 실행하는데, 따로 거짓인 경우에 대한 조건문을 작성하지 않고 거짓일 경우에 수행할 구문 블록을 추가하면 코드의 의도가 더 명확해 질 것입니다. 이를 위해 if… else… 구조를 사용합니다. 위 코드를 if… else…로 다시 작성하면 다음과 같습니다.

1. if ((val % 3) == 0) /\* 3의 배수인 경우 \*/
2. printf(“\*”);
3. else /\* 3의 배수가 아닌 경우 \*/
4. printf(“%d”, val);

조금 더 확장시켜볼까요? 예를 들어 3의 배수면 \*문자를, 5의 배수면 # 문자를, 그 외의 숫자일 경우에는 값을 그대로 출력하는 코드를 작성해 봅시다. 어떻게 구현할지 잠시 생각하는 시간을 가진 다음 계속 코드를 살펴봅시다.

조건식이 거짓일 경우 조건문은 else 블록을 실행합니다. 이때 또다른 조건식을 다음과 같이 추가할 수 있습니다.

if ((val % 3) == 0) /\* 3의 배수인 경우 \*/

printf(“\*”);

else if ((val % 5) == 0) /\* 5의 배수인 경우 \*/

printf(“#”);

else /\* 3의 배수도, 5의 배수도 아닌 경우 \*/

printf(“%d”, val);

이와 같이 조건문을 계속 이어서 실행 흐름을 변경할 수 있습니다. 다만, 주의해야할 점이 있습니다. 만약 val 변수의 값이 15이면 어떻게 될까요? 15는 3의 배수이기도 하지만 5의 배수이기도 합니다. 과연 어떤 값이 출력될까요? 먼저 실행해 봅시다.

Enter the number:

15

\*

3의 배수이므로 \*가 표시되었습니다. 잠시만요. 그러면 5의 배수인지 여부를 확인하는 조건문은 어떻게 된 것일까요? 그렇습니다. else if는 그 앞에 나온 if 조건문이 거짓일 경우에만 실행되는 조건문 입니다. if… else if… 구조를 사용할 때에는 조건문의 순서에 따라 다른 결과가 나올 수 있다는 점에 주의를 기울여야 합니다.

위 조각 코드를 글로 설명하면 다음과 같습니다.

만약 3의 배수이면 \* 글자를 표시한다.

그렇지 않고(즉, 3의 배수는 아니지만) 5의 배수이면 #글자를 표시한다.

그외 경우는 값을 그대로 표시한다.

조건문은 if로 반드시 시작해야 하며, else if 또는 else는 선택사항입니다. else if는 원하는 대로 조건을 추가할 수 있습니다. 그렇지만 ‘그 외 경우에는’을 의미하는 else는 반드시 마지막에 배치되어야 합니다. 그래서 if..else if…else.. 구문에 사용되는 조건식은 논리적으로 순차적인 경우가 많습니다.

조건문에서 참일 때 수행될 코드블록과 거짓일 때 수행될 코드블록에는 또다시 조건문을 포함할 수 있습니다. 보통 특정 조건일 경우 다시 추가 조건에 따라 실행해야 할 동작이 달라지는, 즉 조건이 계층적으로 연결되는 경우에 사용됩니다. 이런 형태를 중첩 조건문nested conditional statement이라 부릅니다.

if (조건식1)

{

if (조건식2)

구문1;

else

구문2;

구문3;

}

else

{

if (조건식3)

구문4;

else

구문5;

구문6;

}

보통 조건문이 추가될 때마다 들여쓰기를 통해 논리적 수준을 구별해 줍니다. 또한 조건문 블럭이 끝나면 빈 줄을 추가하여 조건문이 끝났음을 구별해 주는 것이 좋습니다. 위 조각 코드에서 구문1은 조건식1과 조건식2가 모두 참일 경우에만 실행되며, 구문2는 조건식1은 참이지만 조건식 2는 거짓일 경우에만 실행됩니다. 구문3은 조건식2와 무관하게 조건식1이 참이면 실행됩니다. 구문4는 조건식1은 거짓이고 조건식3은 참일 경우에만 실행되며, 구문5는 조건식 1과 조건식 2가 모두 거짓일 때에만 실행됩니다. 구문6은 조건식3과 무관하게 조건식1이 거짓이면 실행됩니다.

물론 구문1에도 다시 중첩 조건문을 사용할 수 있습니다. 하지만 중첩 조건문이 많아지면 코드의 가독성이 떨어집니다. 이럴 경우 함수를 분리하거나 조건 논리를 간소화하는 방법을 살펴보아야 합니다.

리팩토링refactoring

코드의 외부 동작은 바꾸지 않고 그대로 유지하면서 더 읽기 쉽고, 이해하기 쉽고, 유지관리하기 좋게 내부 구조를 바꾸는 과정을 리팩토링이라고 합니다.

올림픽에서는 시합 성적에 따라 다른 메달을 수여합니다. 1등에게는 금메달을, 2등에게는 은메달을, 3등에게는 동메달을, 그 외 등수에는 메달을 수여하지 않습니다. 사용자가 등수를 입력하면 메달 종류를 표시하는 코드를 작성해 봅시다.

1. #include <stdio.h>
2. int main()
3. {
4. int rate;
5. printf(“Enter your rate:\n”);
6. scanf(“%d”, &rate);
8. /\* 사용자가 입력한 등수에 따라 메달의 색상을 표시하도록 코드를 작성해 봅시다. \*/
9. return 0;
10. }

이번 장에서 배울 내용은

* 조건문
  + if / if else / if else if else
  + 중첩
  + switch와 break
* 순환문
  + for / while / do.. while
  + continue, break

—--

If else의 예: 홀짝.

Switch: 윤년 판단, 계산기?

If else 중첩.. 이어지는 if..else if 와 중첩 if {if}

성적 처리,

if에서 세미콜론 붙인 실수.

루프

for는 미리 정의한 횟수만큼 반복하거나, 특정 조건을 만족할 때 반복하거나, for반복문을 시작하는 시점에 계산된 값을 기준으로 반복.

팩토리얼 구하기

continue 현재 블록에 대한 실행을 멈추고, 다시 for의 첫문장으로 넘어가 반복 실행.