1. 문자열

C 언어에서 문자열은 `char` 자료형의 배열로 표현되며, 이는 연속된 문자들을 순서대로 저장하는 구조이다. 문자열은 마지막에 `'\0'`(NULL 문자)이 붙어 있어 문자열의 끝을 나타낸다. 이 `'\0'` 문자 덕분에 컴퓨터는 배열의 끝까지 가지 않고도 문자열의 끝을 쉽게 알 수 있다. 예를 들어, "Hello"라는 문자열은 메모리에 `H`, `e`, `l`, `l`, `o`, `\0`로 저장되어 총 6바이트를 차지한다.

문자열 초기화 및 길이

문자열을 초기화하는 방법은 크게 두 가지로 나뉜다. 첫째는 `char str[] = {'H', 'e', 'l', 'l', 'o', '\0'}`와 같이 개별 문자와 `'\0'`을 직접 배열에 넣는 방법이다. 둘째는 `char str[] = "Hello"`처럼 큰따옴표로 문자열을 한 번에 선언하는 방법이다. 두 번째 방식은 `'\0'`이 자동으로 추가되기 때문에 코드가 간결해지는 장점이 있다.

문자열의 길이는 `'\0'` 이전까지의 문자 개수로 정의된다. 예를 들어 "Hello"의 길이는 5이며, `strlen()` 함수를 사용하면 이 문자열 길이를 구할 수 있다. 배열의 크기는 문자열의 길이보다 하나 더 커야 하므로, "Hello" 문자열을 저장할 배열의 크기는 6이어야 한다. 이처럼 문자열과 배열의 크기를 잘 구분해야 하며, `strlen` 함수는 문자열 길이를 구할 때 `'\0'`을 포함하지 않는다는 점을 기억해야 한다.

문자열 입출력

문자열을 입출력할 때는 `printf`와 `scanf`에서 `%s` 형식을 사용한다. `scanf`는 문자열을 입력받을 때 공백 문자가 나오면 입력을 종료한다. 공백을 포함하는 문자열을 입력받고자 할 때는 `fgets` 같은 함수를 사용해야 한다. 또한, 문자열의 각 문자에 접근하여 대문자 변환, 소문자 변환, 또는 숫자 여부 확인 등의 연산을 수행할 수 있다.

2. 연산자와 수식

연산자는 데이터를 처리하는 데 필요한 다양한 연산을 수행하며, C 언어에는 수많은 연산자가 제공된다. 이를 통해 데이터를 계산, 비교, 논리 판단 등을 할 수 있으며, 각 연산자는 우선순위와 결합 순서를 가지고 있다.

대입 연산자

대입 연산자는 변수에 값을 할당하는 역할을 한다. 기본적으로 `=` 연산자가 있으며, 이 외에도 `+=`, `-=`, `\*=`, `/=`, `%=`와 같은 복합 대입 연산자들이 있다. 예를 들어 `a += 2`는 `a = a + 2`와 같은 의미이며, 연산을 간결하게 표현할 수 있다. 이러한 복합 대입 연산자는 코드 가독성을 높이고, 연산 과정을 줄이는 장점이 있다.

산술 연산자

산술 연산자는 덧셈(`+`), 뺄셈(`-`), 곱셈(`\*`), 나눗셈(`/`), 나머지(`%`) 연산을 수행한다. 이러한 산술 연산자는 기본적인 수치 계산을 가능하게 하며, 피연산자들이 정수형인지 실수형인지에 따라 결과도 달라진다. 나머지 연산자인 `%`는 두 정수 사이의 나머지를 반환하므로, 실수형에서는 사용할 수 없다.

증감 연산자

증감 연산자는 변수의 값을 1씩 증가(`++`)시키거나 감소(`--`)시킨다. 전위 증감 연산자(`++a`, `--a`)는 먼저 변수의 값을 증가 또는 감소시킨 후 반환하는 반면, 후위 증감 연산자(`a++`, `a--`)는 현재 값을 반환한 뒤에 증가 또는 감소시킨다. 예를 들어, `++a`는 변수를 먼저 1 증가시킨 후 그 값을 사용하고, `a++`는 현재 값을 사용한 후 다음에 1을 증가시키는 차이가 있다.

비트 연산자

비트 연산자는 `&`, `|`, `^`, `~`, `<<`, `>>`로 이루어져 있으며, 비트 단위의 연산을 수행한다. 이는 시스템 프로그래밍에서 주로 사용되며, 하드웨어를 제어하거나 데이터의 특정 비트를 조작할 때 유용하다. `&`는 AND 연산, `|`는 OR 연산, `^`는 XOR 연산을 수행하며, `~`는 각 비트를 반전시킨다. `<<`와 `>>`는 각각 비트를 왼쪽 또는 오른쪽으로 이동시키며, 이동된 자리에는 0이 채워진다. 예를 들어, `a << 1`은 `a`의 모든 비트를 왼쪽으로 한 칸 이동시켜 값이 두 배가 되는 결과를 낳는다.

3. 연산자 우선순위와 결합 순서

C 언어의 연산자들은 특정 우선순위와 결합 순서를 가진다. 연산자 우선순위는 여러 연산자가 함께 사용될 때 어떤 연산을 먼저 수행할지를 결정하며, 결합 순서는 같은 우선순위를 가진 연산자들 간의 연산 순서를 의미한다.

연산자 우선순위

C 언어에서 우선순위가 가장 높은 연산자는 괄호 `()`와 배열 인덱스 연산자 `[]`이다. 이들은 모든 연산자보다 먼저 실행되므로, 수식에서 괄호를 사용하여 연산 순서를 제어할 수 있다. 산술 연산에서는 곱셈과 나눗셈이 덧셈과 뺄셈보다 높은 우선순위를 가지므로, `2 + 3 \* 4`와 같은 수식은 `3 \* 4`가 먼저 계산된다. 이러한 우선순위 규칙을 통해 수식을 평가할 때의 연산 순서를 정확하게 정의할 수 있다.

결합 순서

같은 우선순위의 연산자가 있을 때, 결합 순서가 연산 순서를 결정한다. 대부분의 이항 연산자는 왼쪽에서 오른쪽으로 결합된다. 예를 들어 `a - b + c`는 `(a - b) + c`로 평가된다. 반면에, 단항 연산자(`++`, `--`)와 대입 연산자(`=`)는 오른쪽에서 왼쪽으로 결합된다. 예를 들어 `a = b = 5`는 `b`에 5를 대입한 후 `a`에 대입하여 `a`와 `b`가 모두 5가 된다.

우선순위와 결합 순서 규칙

연산 순서를 쉽게 기억하기 위해 "계산하여 비교한 뒤, 판단하여 저장한다"라는 규칙을 사용할 수 있다. 우선 계산(산술 연산)을 수행하고, 그 결과를 비교 연산이나 논리 연산을 통해 판단하며, 마지막으로 대입 연산을 통해 값을 저장한다.