1. 자료형의 개념과 메모리 크기

자료형은 변수에 저장된 데이터가 어떤 성질을 가졌고, 이를 메모리에서 어떻게 관리할지를 결정하는 중요한 요소이다. 컴퓨터는 모든 데이터를 이진수로 저장하므로, 이를 특정 형식에 맞춰 해석해야 데이터의 의미가 분명해진다. 예를 들어, 숫자로 해석할지, 문자로 해석할지, 또는 다른 구조로 해석할지는 자료형에 달려있다. C 언어에서 `int`, `float`, `double`, `char` 등의 자료형이 제공되는 이유도 바로 여기에 있다. 자료형이 없으면 데이터가 메모리에서 해석되지 않아 혼란을 초래하게 되므로, 자료형 지정은 필수적이다.

자료형마다 메모리에서 차지하는 크기가 고정되어 있는데, 예를 들어 `int`는 보통 4바이트를 차지하며, `double`은 8바이트를 차지한다. 이러한 크기는 시스템 및 컴파일러에 따라 차이가 있을 수 있지만, 대체로 `char`는 1바이트, `short`는 2바이트, `int`와 `float`는 4바이트, `double`은 8바이트로 할당된다. 메모리 크기를 알기 위해 `sizeof()` 함수를 사용하며, 이를 통해 변수 또는 자료형이 차지하는 메모리 공간을 쉽게 확인할 수 있다.

2. 정수형과 실수형 표현

C 언어의 정수형 자료형은 `char`, `short`, `int`, `long`, `long long` 등이 있으며, 이들 각각은 서로 다른 크기와 범위의 정수를 표현할 수 있다. `int`는 32비트로 약 -2.1억에서 2.1억 사이의 범위를 다루고, `unsigned`를 앞에 붙이면 양수로만 표현 가능하여 범위가 두 배로 확장된다. 예를 들어, `unsigned int`는 0부터 약 4.2억까지 표현 가능하다. 이처럼 `unsigned`를 사용하여 부호 없는 정수를 표현하면 메모리 효율이 증가하며, 더 큰 양의 값을 처리할 수 있게 된다.

컴퓨터는 2의 보수 방식(Two's Complement)으로 음수를 표현한다. 2의 보수 방식은 첫 번째 비트를 부호 비트로 사용하는데, 이 비트가 1이면 음수, 0이면 양수로 해석된다. 예를 들어, 양수 5는 이진수로 00000101이며, 이를 2의 보수로 변환하면 11111011이 되어 -5를 표현하게 된다. 이러한 방식은 산술 연산에서 복잡한 추가 연산 없이 음수를 효율적으로 처리할 수 있도록 돕는다.

실수를 표현하기 위해 `float`, `double`, `long double` 등의 자료형이 사용된다. 실수는 부동 소수점(floating-point) 표현 방식을 따르며, IEEE 754 표준에 의해 32비트(단정밀도)와 64비트(배정밀도)로 나뉜다. `float`는 약 7자리의 유효숫자를, `double`은 15자리의 유효숫자를 표현할 수 있으며, 매우 큰 수나 작은 수를 효율적으로 나타내기 위한 목적으로 활용된다. 실수 표현은 정수형보다 더 복잡하며, 특히 소수점 이하의 값을 정확히 표현하기 위해 과학적 표기법이 사용된다.

3. 진수와 숫자 시스템

컴퓨터는 모든 데이터를 2진수로 표현하므로, 8진수와 16진수로 변환하여 사용하기도 한다. C 언어에서는 `0b` 접두사를 사용해 이진수를, `0x` 접두사로 16진수를 나타낼 수 있다. 이처럼 다양한 기수 체계는 데이터의 구조적 표현과 연산을 더욱 간결하게 만들어주며, 특히 시스템 프로그래밍과 하드웨어 제어에 유용하다.

컴퓨터에서 음수를 표현하는 데는 2의 보수 방식을 사용하는데, 이 방법은 음수의 표현을 표준화하고, 덧셈과 뺄셈을 단순화해준다. 예를 들어, 양의 정수 5를 이진수로 나타내면 00000101인데, 이를 2의 보수로 변환하면 11111011이 되며 이는 -5를 나타낸다. 2의 보수 방식은 덧셈과 뺄셈 연산에서 효율적일 뿐만 아니라 0을 유일하게 표현할 수 있다는 장점이 있다.

컴퓨터 프로그래밍에서 8진수와 16진수는 메모리 주소나 하드웨어 레지스터의 값을 간결하게 표현하는 데 사용된다. 예를 들어, 메모리 주소를 나타낼 때 16진수로 표현하면 단위가 4비트로 고정되어 가독성이 높아진다. 이처럼 다양한 진수 체계는 프로그래밍의 효율성을 높이고, 하드웨어와의 통신을 원활하게 한다.

4. 문자 자료형과 ASCII 코드

C 언어에서 `char` 자료형은 문자 데이터를 저장하며, ASCII(American Standard Code for Information Interchange) 코드 체계를 사용한다. ASCII 코드는 영어 알파벳, 숫자, 기호 등의 문자에 고유의 숫자 값을 할당하는 방식으로, 'A'는 65, 'a'는 97과 같은 식으로 각 문자마다 고유의 코드 값을 가진다. 이로써 컴퓨터는 문자 데이터를 숫자로 변환하여 처리할 수 있다.

`scanf`와 `printf`에서 `%c` 형식을 사용하여 문자를 입력받고 출력할 수 있으며, 이를 통해 ASCII 값과 직접 연산할 수 있다. 예를 들어, 대문자 'A'를 소문자로 변환하려면 `+32`를 더하거나, 소문자 'a'를 대문자로 변환할 때는 `-32`를 하면 된다. 이는 알파벳 소문자와 대문자가 ASCII 코드에서 일정한 거리만큼 떨어져 있기 때문이다. 이러한 특성을 활용해 숫자나 문자 여부를 판별하는 것도 가능하다.

추가로, `char` 자료형을 정수형으로도 사용할 수 있는데, 이는 `char` 자료형이 사실상 1바이트의 정수형으로 처리되기 때문이다. ASCII 코드는 8비트(1바이트)로 구성되어 있어 문자를 표현하는 데 충분하며, 메모리의 효율성을 높이는 장점이 있다.