# 4장 기본 데이터 타입

## 학습 목표

C언어로 값을 표현하는 방법을 이해하고, 적절한 타입을 선택할 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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| C 언어의 기본 데이터 타입을 안다. |  |  |  |  |  |
| 코드에서 값을 표현하기 위하여 적절한 데이터 타입을 가진 변수를 만들 수 있다. |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

프로그램은 주어진 데이터를 이용하여 어떤 동작을 수행하거나 얻고자 하는 결과 데이터를 만듭니다. 데이터는 값을 표현합니다. 즉, 값은 특정 데이터를 가리키는 구체적인 숫자나 문자를 의미합니다. 이번 장에서는 C언어에서 값을 표현하는 방법에 대해 살펴봅니다.

C언어는 코드에서 다루려는 값이 어떤 형태의 값인지를 미리 지정합니다. 이때 값의 형태를 데이터 타입 또는 자료형이라고 부릅니다. 이번 장에서는 여러 데이터 타입 중 가장 기본 타입에 대해 살펴 봅니다. C언어는 정수형, 실수형, 문자형, 불린형 4가지 기본 데이터 타입을 지원합니다. 각 데이터 타입마다 어떤 값을 표현할 수 있는지, 어떻게 값을 표현하는지에 대해 살펴 봅니다.

프로그램에서 값을 다루려면 메모리에 값을 담아둘 곳이 필요합니다. 이를 변수라 부릅니다. C 언어에서 변수를 정의하여 값을 저장하는 방법에 대해 살펴 봅니다. 또한 프로그램이 실행되면서 변하는 값과 변하지 않는 값으로 나뉘어 집니다. 코드에서 값의 변경 가능성을 표현하는 방법을 살펴 봅니다.

## 4.1 데이터 타입과 변수

프로그램은 코드에서 사용하는 데이터의 값을 메모리에 저장합니다. 메모리는 저장 공간마다 고유 식별자인 메모리 주소를 가집니다. 우리가 어떤 장소를 주소로 찾아갈 수 있듯이 프로그램은 메모리 주소(번지수)를 통해 특정 메모리 공간에 접근할 수 있습니다.

프로그램은 특정 메모리 공간을 확보한 다음 그 곳에 값을 저장합니다. 값의 형태에 따라 사용하는 메모리 공간의 크기가 달라집니다. C언어에서는 코드에 값의 형태를 미리 지정해 주므로, 값을 표현하는데 사용할 메모리 크기를 미리 알고 준비할 수 있습니다. 더 큰 공간을 사용할수록 더 많은 갯수의 값을 표현할 수 있습니다.

프로그램이 데이터 값을 어느 위치의 메모리 공간에 저장할 것인지는 프로그램을 실행할 때마다 달라집니다. 만약 고정된 특정 메모리 주소를 사용하여 데이터의 값을 저장한다고 생각해 봅시다. 해당 메모리 공간을 특정 프로그램만 사용할 수 있으므로 여러 프로그램을 실행하려면 메모리 용량이 무한대여야 할 것입니다. 또한 실수로 코드에서 주소를 잘못 지정하면 엉뚱한 값을 읽게 될 것입니다.

프로그램 코드를 작성할 때 개발자가 값을 다루려면 그 값을 표현하는데 사용되는 형식과 값이 저장된 공간을 표현할 방법이 필요합니다. 프로그래밍 언어에서는 값을 표현하는 형식을 데이터 타입 또는 자료형이라 부르고, 저장된 공간을 가리키기 위하여 변수라는 개념을 사용합니다.

C언어에서는 값을 다루기 위해 어떤 데이터 타입의 값인지를 코드를 통해 미리 컴파일러에게 알려줘야 합니다. C언어 컴파일러는 그에 따라 적절한 메모리 공간을 사용할 수 있도록 저장공간을 확보하는 명령 집합을 만들어 냅니다. 이렇게 확보된 메모리 공간에 저장된 값을 가리킬 때에는 변수명을 이용합니다. 이와 같이 변수의 데이터 타입을 코드에 명시해 주는 방식의 언어를 정적 언어Statically Typed Language라고 부릅니다.

정적언어? 동적 언어?

정적 언어는 변수의 데이터 타입을 코드에 명시적으로 지정해 주며, 한번 타입이 지정된 변수는 다른 타입의 변수가 될 수 없습니다. 하지만 파이썬, 자바스크립트와 같은 언어는 실행할 때 변수의 데이터 타입이 결정되며, 필요에 따라 변수의 데이터 타입이 바뀔 수도 있습니다. 이런 언어를 동적 언어Dynamically Typed Language라고 부릅니다.

정적언어는 컴파일 타임, 즉 컴파일러가 코드를 컴파일할 때 변수의 데이터 타입을 검사하고, 잘못된 값을 넣고 있는지 확인하고, 데이터 타입에 따라 코드 최적화를 수행할 수 있기 때문에 동적 언어에 비해 안정성이 높고, 실행성능이 뛰어납니다. 하지만 동적 언어는 런타임, 즉 코드가 실행될 때 변수의 타입을 결정하기 때문에 코드를 빠르게 작성할 수 있고 보다 유연하지만 정적 언어에 비해 실행 성능이 떨어지고, 실행중 타입 오류가 발생할 가능성이 있습니다.

정적 언어로는 C/C++, 자바, 고, 코틀린, 스위프트 등이 있고, 동적 언어로는 파이썬, 루비, 자바스크립트 등이 있습니다.

그러면 C 언어 코드에서 값을 사용하고 싶을 때 어떻게 사용하는지 알아 봅시다.

변수\_데이터\_타입 변수명;

변수명 = 값;

예를 들어 정수값 123을 보관한다고 가정해 봅시다. C언어에서 정수를 표현하는 기본 데이터 타입은 int입니다. 이를 val이라 이름붙인 메모리 공간에 보관하는 코드를 작성하면 다음과 같습니다.

int val;

val = 123;

이 코드의 의미는 다음과 같습니다.

* 정수(int)값을 보관하는 데 필요한 4바이트 크기의 메모리 공간을 확보하고, 이 메모리 공간의 이름을 val이라고 붙입니다.
* val이라는 메모리 공간에 정수 123의 값을 보관합니다. 이렇게 변수에 값을 넣는 과정을 할당assignment한다고 말합니다.
* 등호(‘=’)는 할당 연산자이며, 오른쪽 부분에 적힌 값을 왼쪽 부분의 변수에 할당합니다.
* 123은 정수값을 코드에서 표현하는 방법이며, 정수 리터럴이라 부릅니다. 리터럴literal은 코드에서 값을 표현하는 방법을 말합니다.

값을 보관하기 위한 메모리 공간을 확보한 다음 값을 넣는 구문을 따로 작성했습니다. 변수를 선언한 뒤에는 해당 변수에 값을 여러 번 할당할 수 있습니다. 같은 값을 저장해도 되지만, 같은 타입의 다른 값도 저장할 수 있습니다.

int val; // 정수값을 보관할 val 변수를 선언합니다.

val = 123; // val 변수값은 123이 됩니다.

val = 456; // 이제 val 변수값은 456이 됩니다.

여러 사람이 이용할 수 있는 창고를 머리속에 그려보세요. 창고지기에게 창고를 이용하겠다고 말하면 창고지기는 여러분이 이용할 수 있는 공간을 창고 어디엔가에 마련해 줍니다. 이 수납 공간에서는 여러분이 원하는 크기대로 영역을 나눠 물건을 맡길 수 있습니다. 저장하려는 물건의 크기에 따라 영역의 크기가 달라집니다. 창고에서 특정 물건이 저장되는 위치는 매번 달라지지만, 어떤 물건이 저장되는지 태그를 붙여두었으므로 태그만 알면 수납공간을 찾아서 저장된 물건을 찾거나 그 공간에 있던 물건 대신 다른 물건을 저장할 수 있습니다.

눈치를 챈 독자도 있겠지만 창고는 컴퓨터의 저장장치이고, 물건은 여러분이 저장하려는 값입니다. 여러분을 대신하여 프로그램은 창고지기, 즉 컴퓨터의 운영체계에게 창고를 이용하겠다고 요청합니다. 그러면 창고지기는 여러분만 사용할 수 있는 저장영역을 내어 줍니다. C언어로 개발된 프로그램은 어떤 물건을 저장할 것인지 그 크기를 코드에 명시된 타입으로 알 수 있기 때문에 값이 필요한 시점에 빨리 꺼낼 수 있도록 최적의 위치에 그 값을 저장합니다. 값을 찾을 때는 코드에 명시한 이름으로 찾습니다.

{변수명 규칙}

변수이름은 영어 대소문자, 숫자, 밑줄(\_)을 조합하여 만들 수 있다. 모두 숫자이거나 숫자로 시작하는 문자열은 변수명이 안됨.

공백도 안됨.

대소문자를 구별함.

예약어 안됨.

NOTE

변수명 명명법

4.2 비트와 바이트

## 4.2 기본 데이터 타입

컴퓨터에서는 모든 값이 바이너리, 즉 2진수의 형태로 표현됩니다. 그러면 C 언어에서는 어떤 데이터 타입을 사용할 수 있을까요? 먼저 가장 간단한 불린형부터 알아봅시다.

### 4.2.1 불린 값

불린 값은 참 또는 거짓, 둘 중 한 값을 가집니다. 예를 들어 전등이 켜졌는지 꺼졌는지, 회의에 참석했는지 불참했는지, 시험에 합격했는지 불합격했는지 등과 같이 이 값 아니면 저 값을 가지는 데이터를 표현하는데 적합합니다. 영국 수학자인 조지 불Gorge Boole이 창안한 불 대수에 기반하므로 불린Boolean 이라는 이름을 가지게 되었습니다. 조지 불은 참과 거짓, 이 두가지 값을 가지고 이루어지는 논리 연산으로 대수 체계를 설계한 수학자입니다.

C언어에서 불린 값의 타입은 bool 이며, true 또는 false를 값으로 할당할 수 있습니다. 주로 조건을 판단하거나 논리적 연산을 수행하거나 프로그램의 실행 흐름을 변경하는 상황에서 사용됩니다. 이 값을 표현하기 위해서는 1비트가 필요합니다. 하지만 컴퓨터는 비트 단위로 처리하지 않고 1바이트의 메모리를 사용합니다. 예제 코드를 통해 살펴 봅시다.

1. #include <stdbool.h> // 불린 타입을 사용하기 위한 헤더 포함
2. int main() {
3. bool isPassed = true; // 불린 값을 저장할 isPassed 변수를 선언하고 참(true)을 할당
4. if (isPassed) { // isPassed 변수값이 참이면 중괄호 코드 블록을 실행
5. print(“시험 합격”);
6. }
7. return 0;
8. }

흥미롭게도 C언어에 처음부터 불린 타입이 존재하지는 않았습니다. 1999년 제정되었기 때문에 C99라 불리는 C 언어 표준안에서 처음 불린 타입이 등장했습니다. 그래서 다른 데이터타입과 달리 불린타입은 stdbool.h 헤더파일에 정의되어 있습니다.

그럼 표준이 되기 전에는 불린 타입의 값을 어떻게 표현했을까요? 바로 뒤에 살펴볼 정수 타입의 값으로 표현했습니다. 대부분 정수값 0을 거짓(false), 그외의 값을 참(true)로 간주하여 사용했습니다. 다음 조각코드와 같이 매크로로 TRUE과 FALSE를 정의하여 사용했습니다.

#define TRUE 1

#define FALSE 0

int isPassed = TRUE;

매크로(MACRO)

자주 반복되는 코드나 원주율처럼 자주 사용되는 상수값을 다른 코드로 대체하도록 정의하는 기능입니다. #define으로 시작됩니다. 위와 같은 매크로가 정의되었다면 코드 본문에 쓰인 TRUE와 FALSE는 각각 1과 0으로 바뀐 다음, 컴파일이 진행됩니다.

매크로를 사용하면 가독성이 향상될 수 있지만, 타입의 안정성과 디버깅이 어렵기 때문에 상수를 정의하거나 inline 함수를 사용하는 것이 좋습니다.

이제는 표준이 제정되었으므로, 불린 값을 표현하기 위해 매크로를 사용하여하지 말고 표준 불린 데이터 타입을 사용합시다. 예전 프로젝트 코드를 이해하기 위해 소개했습니다.

### 4.2.2 정수 값

컴퓨터에서 가장 많이 다루는 값이 아마도 숫자입니다. C 언어에서는 숫자를 정수와 소수로 구분하여 데이터 타입을 지정합니다. 정수는 소수점을 가지지 않으며, 양의 정수와 0, 음의 정수로 이루어 집니다.

자바스크립트, 루아 등 일부 언어는 이 둘을 구분하지 않고 Number(숫자)타입으로 다루기도 합니다.

먼저 부호를 가지지 않은 양의 정수와 0에 대해 살펴보겠습니다. 부호(sign)를 가지지 않기 때문에 모든 비트를 값을 표현하는데 사용합니다. 당연한 이야기이지만, 비트를 더 많이 사용할수록 더 많은 값을 표현할 수 있습니다. 정수는 크기에 따라 타입을 가리키는 용어가 달라지지만, 정수를 의미하는 integer에서 앞 부분을 따와서 int 로 표현합니다. 부호를 가지지 않으므로 타입 앞에 unsigned를 붙여 줍니다. unsigned int는 부호를 가지지 않는, 즉 0 또는 양의 정수를 표현하는 타입을 의미합니다.

|  |  |  |
| --- | --- | --- |
| 데이터타입 | 타입 크기(바이트) | 표현할 수 있는 값의 범위 |
| unsigned char | 1 | 0 ~ 255 |
| unsigned short | 2 | 0 ~ 65,535 |
| unsigned int | 4 | 0 ~4,294,967,295 |
| unsigned long | 8 | 0 ~ (264-1). 대략 1경 8464조 7440억 |
| unsigned long long | 8 | 0 ~ (264-1). 대략 1경 8464조 7440억 |

unsigned long long은 0부터 1경 8464억까지의 숫자를 표현할 수 있습니다. 만약 시간을 초 단위로 표현한다면 대략 584억년의 시간을 초단위로 표현할 수 있습니다. 우주의 나이는 대략 138억년으로 알려져 있습니다.

n개의 비트로 표현할 수 있는 값의 개수는 2n 개입니다. 하지만 비트가 모두 0일 경우 0값을 표현하기 때문에, 0부터 2n-1까지의 정수값을 표현할 수 있습니다.

이번에는 음의 정수까지 표현해 봅시다. 양수인지 음수인지를 표현하는데 1비트를 사용하므로 같은 타입의 크기를 사용하더라도 표현할 수 있는 값의 범위가 줄어듭니다. 따라서 1바이트로 정수를 표현하는 char는

한가지 문제가 더 있습니다. 비트 한 개를 부호로 사용하다보니 음수인 0과 양수인 0이 생기게 됩니다. 전산과학자들의 선구자들도 이 문제를 잘 이해하고 있었습니다. 해결책으로 찾아낸 것이 음수를 표현할 때 2의 보수로 표현하는 방법입니다. 그렇게 되면 음수를 하나 더 표현할 수 있습니다. 따라서 부호를 가지는 정수 타입과 표현할 수 있는 값의 범위는 다음과 같습니다.

|  |  |  |
| --- | --- | --- |
| 데이터타입 | 타입 크기(바이트) | 표현할 수 있는 값의 범위 |
| char | 1 | -128 ~ 127 |
| short | 2 | -32768 ~ 32,767 |
| int | 4 | -2,147,483,648 ~ 2.147.483,657 |
| long | 8 | -263 ~ (263-1) |
| long long | 8 | -263 ~ (263-1) |

음수를 비트로 표현하는 방법은 이 책의 범위를 벗어나므로 자세히 다루지는 않습니다. 왜 2의 보수로 음수를 표현하고, 이를 기반으로 정수 연산을 수행하는 것은 다른 전산 이론 기본 서적에 많이 소개되어 있습니다. 찰스 패촐드의 <코드:하드웨어와 소프트웨어에 숨어있는 언어> 9장부터 12장까지 자세히 다루고 있습니다. 2의 보수로 음수를 표현하면 동일한 방식으로 음수와 양수의 덧셈을 처리할 수 있습니다. 계산 과정이 통일되기 때문에 같은 하드웨어 회로로 처리가 가능해져서 연산이 간소해지고 효율은 더 좋아집니다.

부호가 있을 경우 0을 양수로 생각하므로, 음수보다 1만큼 작은 수까지 표현할 수 있습니다.

각 정수 타입의 최댓값과 최솟값은 limits.h 헤더 파일에 정의되어 있습니다.

앞에서 int 또는 unsigned int는 4바이트로 정수값을 표현한다고 말했습니다. 엄격히 말하자면 항상 그렇다고 말할 수는 없습니다. C언어가 만들어질 초창기 하드웨어의 상황을 고려하여 표준에서는 각 데이터 타입의 최소 크기만 규정하였습니다. 즉, 표준안에 따르면 int타입은 최소 16비트로 표현된다고만 정의되어 있습니다. 그로 인해 int 타입의 크기는 플랫폼이나 컴파일러에 따라 다를 수도 있습니다만, 요즘 사용되는 컴파일러와 플랫폼은 거의 대부분 앞에서 설명한 크기를 따릅니다.

만약 다른 플랫폼과 바이너리 데이터를 주고 받아야 한다면 값을 표현하는 방식도 같아야 합니다. 이런 상황을 해결하기 위해 C99 표준안에서는 크기를 지정한 정수 타입도 추가하였습니다. 크기를 명시적으로 지정한 정수타입은 stdint.h 헤더 파일에 정의되어 있으며, 다음과 같습니다.

|  |  |  |
| --- | --- | --- |
| 타입 크기(바이트) | 부호 있는 정수 타입 | 부호 없는 정수 타입 |
| 1 (8비트) | int8\_t | uint8\_t |
| 2(16비트) | int16\_t | uint16\_t |
| 3 (32비트) | int32\_t | uint32\_t |
| 8 (64비트) | int64\_t | uint64\_t |

불린 타입에 비해 정수 타입은 여러 종류의 타입이 존재합니다. 표현하려는 숫자의 범위에 적합한 타입을 선택하는 것이 좋습니다만, 일반적으로 int를 선택하는 것도 나쁘지 않은 선택입니다.

그러면 이렇게 선언한 타입에 값을 할당하려면 어떻게 해야 할까요? C언어에서는 우리가 사용하는 10진수 이외에도 8진수 또는 16진수로 값을 할당할 수 있습니다. 8진수로 값을 설정할 때에는 0로, 16진수로 값을 설정할 때에는 0x로 시작합니다.

int decimalValue = 123; // 10진수 값은 그대로 표현

int hexaValue =0x7B; // 16진수 값은 0x로 시작

int octetValue = 0173; // 8진수 값은 0으로 시작

위 코드에서 세 변수는 모두 십진수 123을 저장합니다.

그러면 어떤 경우에 10진수가 아닌 8진수나 16진수의 형태로 값을 표현할까요? 첫째, 하드웨어 수준의 작업에서 메모리 주소를 다루거나 특정 값을 표현할 때 16진수를 사용합니다. 둘째, 비트 단위의 연산이나 처리가 필요할 때 16진수로 표현하면 가독성이 좋아집니다. 셋째, 10진수로 표현하기 어려운 값을 8진수 또는 16진수로 표현하면 가독성이 높아집니다.

int filePermission = 0755; // 유닉스의 파일 권한값을 8진수로 표현

unsigned int color = 0xFF5533; // RGB 색상값을 16진수로 표현

unsigned int mask = 0xFF00; // 상위 바이트값을 얻기 위한 비트 연산 마스트를 16진수로 표현

리터럴(literal)

리터럴은 코드에 직접 표현된 고정된 값을 의미합니다. 즉, 프로그램이 실행되더라도 변하지 않는 값입니다. 리터럴로 표현된 값은 직접 변수에 저장하거나 연산식에 사용될 수 있습니다. 정수를 표현하는 다양한 방식을 정수형 리터럴이라고 부릅니다. 곧 만나게 되겠지만 소수값은 소수점을 포함한 값으로, 문자는 작은 따옴표를 감싼 형태의, 문자열은 큰 따옴표로 감싼 형태의 리터럴 표현을 사용합니다.

리터럴에는 접미사를 사용하여 타입을 보다 명시적으로 나타낼 수 있습니다. 부호없는 정수 타입은 U 또는 u를, long 정수 타입은 L 또는 l을, 실수 타입은 F 또는 f를 뒤에 덧붙입니다.

unsigned int maxLimit = 1234U // unsigned 값

long bigNumber = 123456L // long 값

float pi = 3.14F // 소수 값

### 4.2.3 실수 타입

이제 소수점이 있는 숫자를 표현하는 방법에 대해 살펴봅니다. 정수는 주어진 범위에서 표현될 수 있는 정수의 개수가 정해져 있습니다만, 소수점은 그렇지 않습니다. 소수점 아래에 숫자는 무한이 존재할 수 있습니다. 3.141592로 대표되는 원주율이 대표적인 예입니다.

컴퓨터에서 사용할 수 있는 메모리나 처리 회로의 제약으로 인해 정수와 달리 부호, 유효값과 지수값으로 나누어 값을 표현합니다. 이 표현방식을 부동소수점floating point 형식이라 하며, 미국 전자전기공학회 IEEE 754 표준으로 정해져 있습니다. 부동浮動이란 소수점의 위치가 변경된다는 뜻입니다. 유효값 부분에 사용하는 비트가 많으면 많을 수록 값을 보다 정확히 표현할 수 있습니다. 하지만 결국 실수 타입으로 표현할 수 있는 근사값으로 저장된다는 점을 유의해야 합니다.

실수값의 기본 타입은 float 입니다. 부동 소수점을 의미하는 floating point의 앞글자를 따온 것입니다. float보다 많은 바이트를 사용하여 유효값을 대략 2배로 늘린 타입이 double 이며, 배정밀도 부동소수점이라고 부릅니다. 일반적으로 64비트 컴퓨터 환경에서는 실수값을 표현할 때 double을 사용합니다. 이보다 더 정밀도가 높은 실수값이 필요하다면 long double을 사용합니다. 다만 long double타입의 크기는 컴파일러와 플랫폼에 따라 달라질 수도 있다는 점에 유의해야 합니다.

|  |  |  |  |
| --- | --- | --- | --- |
| 데이터 타입 | 타입 크기(바이트) | 유효 자리수 | 표현할 수 있는 값의 범위 |
| float | 4 (32비트) | 6~7자리 | +- 10-38 ~1038 |
| double | 8(64비트) | 15~16자리 | +- 10-308 ~ 10308 |
| long double | 16바이트(128비트)  (환경에 따라 다를 수 있음) | 대략 34자리 | +- 10-4932 ~ 104932 |

부동소수값을 표현하는 리터럴에서 float타입은 F 또는 f를, long double 타입은 L 또는 l을 끝에 덧붙입니다. 아무것도 없다면 컴파일러는 double 타입의 부동소수값으로 인식합니다.

float value1 =123.456f; // float 타입 변수에 float 값을 할당

float value2 = 123.456; // float 타입 변수에 double 값을 할당.

double value3 = 123.456; // double 타입 변수에 double 값을 할당

double value4 = 123.456f; // double 타입 변수에 float 값을 할당

long double value5 = 123.456l; // long double 타입 변수에 long double 값을 할당

### 4.2.4 문자 타입

이 절을 시작하면서 컴퓨터에서 모든 값은 이진으로 표현된 값으로 표현된다고 말했습니다. 문자도 마찬가지입니다. 컴퓨터가 만들어 졌을 때 표현될 수 있는 문자마다 숫자값을 부여했습니다. 이중 가장 널리 사용된 것이 지금까지도 사용되는 아스키 코드입니다.

아스키 코드에서 표현할 수 있는 문자는 128개였기 때문에 1바이트로 표현이 가능합니다. 따라서 C언어에서 문자를 표현할 때에는 char 타입을 사용합니다. 문자와 숫자를 같은 방식으로 다루게 되면 컴퓨터는 추가적인 변환없이 데이터를 다룰 수 있습니다.

char c = ‘A’;

c = ‘A’ + 1;

c = ‘5’;

int val = c –‘0’;

문자 값을 코드에서 표현할 때 작은 따옴표로 감싸줍니다. 문자 리터럴은 작은 따옴표입니다. 문자 리터럴은 아스키 코드값으로 바뀌어 처리됩니다. 즉, ‘A’는 아스키코드에서 65입니다. 따라서 변수 c에 저장된 값은 65입니다. 따라서 ‘A’+1은 65+1이므로 66이라는 값을 가지는데, 이 값은 아스키코드에서 ‘B’에 해당합니다. 세번째 줄은 처음 보면 헷갈릴 수 있습니다. 변수 c에는 ‘5’라는 문자의 아스키코드값이 저장되어 있습니다. 따라서 c의 값은 글자에 표시된 5가 아니라 53이라는 값을 가집니다. 아스키코드에서 숫자는 0부터 9까지 차례대로 값이 저장되어 있으므로 숫자를 의미하는 문자에서 글자 ‘0’, 즉 아스키코드 58을 뺌으로써 숫자 값을 구할 수 있습니다.

그런데 글자 하나로는 우리의 일상을 표현할 수 없습니다. 우리가 읽고 있는 이 문장도 글자들이 모여 이루어집니다. 프로그래밍 언어에서는 문자열string로 부릅니다. C언어에서는 문자열을 문자의 배열로 표현합니다.배열은 동일한 타입의 값이 여러 개 이어져 있는 타입을 말합니다. 뒤에서 자세히 배우게 되니 걱정하지 않아도 됩니다.

char greeting[100] = “Hello World”;

문자열과 문자는 서로 다른 데이터 타입이므로 리터럴도 다릅니다. 문자는 작은 따옴표로 감싸줬지만 문자열 은 큰 따옴표로 감싸줍니다. 우리가 처음 작성했던 Hello World 예제 프로그램에서도 문자열 리터럴을 사용했습니다. 각괄호는 배열을 의미하는데, 각괄호사이에 있는 숫자는 문자를 100개까지 저장할 수 있다는 뜻입니다. 뒤에서 자세히 다룹니다.

문자열 리터럴에서 중요한 점은 코드에는 보이지 않지만 문자열의 끝을 의미하는 보이지 않는 널NULL 문자가 맨 뒤에 붙어 있다는 점입니다. 이를 널로 끝나는 문자열(Null Terminated String)이라고 합니다. 따라서 위 코드는 12개의 문자로 이루어진 문자열입니다.

문자열은 프로그래밍에서 자주 다루어야 하는 데이터이므로 8장에서 좀 더 자세히 살펴봅니다.

ASCII 코드

아스키코드는 1963년 제정된 정보 교환을 위한 미국 표준 코드(American Standard Code for Information Interchange)의 약자입니다. 7개의 비트를 사용하여 영문 대소문자, 0부터 9까지의 숫자, 특수 문자 32개, 공백 문자 1개, 화면에 표시할 수 없는 제어 문자 33개 등 총 128개의 문자에 대한 코드 값이 정의되어 있습니다.

컴퓨터는 1바이트 즉 8비트로 동작하는데, 7비트만 사용한 까닭은 통신과정에서 발생할지도 모르는 오류를 검출하는 패리티 비트로 한 비트를 사용했습니다.

ASCII 코드는 당시 컴퓨팅 환경의 기술적 제약과 실용성을 고려하여 제정되었습니다. 1바이트로는 한국어와 같은 글자를 표현할 수 없었기 때문에 이후 유니코드 문자 집합이 제정되었습니다. 유니코드는 2바이트 또는 4바이트를 사용하여 여러 언어에서 사용되는 글자를 표현할 수 있는 문자 타입입니다.

하나의 값을 저장하는 단일 변수 타입을 스칼라 변수Scala variable라고 부릅니다.

4.3 타입 변환

좁아지는 변환

커지는 변환

4.4 변수와 상수

변수의 수명

변수명 규칙

상수

변수의 수명

그럼 부호가 있는 정수, 즉 음수까지 표현하려면 어떻게 할까요? 그렇습니다. 1비트를 음수인지, 양수인지를 표현하는데 사용합니다. 보통 최상위비트(MSB)를 사용하여 0이면 양수, 1이면 음수임을 표시합니다. 부호는 표시할 수 있게 되었지만, 한가지 문제가 발생하였습니다. 0은 양수일까요? 음수일까요? 또한 음수와 양수를 더하거나 빼려면 어떤 과정을 거쳐야 할까요?

초기 엔지니어는 음수를 표현할 때 양수에 2의 보수를 취하는 방식으로 표현하면 음수 표현도 단순해 지고, 일관된 규칙에 따라 덧셈과 뺄셈이 이루어져 하드웨어 설계가 단순해 진다는 것을 증명했습니다. 따라서 양수에 2의 보수를 취하여 음수임을 표현합니다. 2의 보수로 음수를 표현하면 양의 0만 존재하게 되고, 음수는 양수보다 한 개 더 많은 값을 표현할 수 있습니다.

4.2.2 실수 - 부동소수

따라서 프로그램에서는 메모리 공간이 고정적이기 때문에 부동 소수점(floating point) 형태로 실수를 표현합니다. 부동 소수점이란 아주 큰 숫자나 작은 숫자를 저장할 때 유효 숫자와 지수로 나눠 표현합니다. 값을 저장하는 크기에 따라 Float, double , long double형으로 나뉘어 집니다.

123.456789라는 값을 float 타입으로 표현하는 방법을 알아보겠습니다. Float는 실수값을 4바이트(32비트)의 메모리 공간에 표현합니다. 전체 32비트중 1비트는 이 값이 양수인지 음수인지를 표시하는 부호 비트(sign bit)로 사용합니다. 그런 다음 맨 처음 1인 비트만 두도록

이와 같은 방식으로 제한된 크기의 비트를 사용하여 넓은 범위의 실수를 표현할 수 있습니다. 다만 정확한 그 값을 표현하기 보다는 표현할 수 있는 가장 가까운 실수값을 가지게 됩니다. 이를 정밀도 손실이라고 합니다.

정밀도 손실로 실수 값을 표현할 때 주의해야 할 부분이 생깁니다. 0.1과 0.2를 더하면 0.3이 되는 것을 알고 있습니다. 하지만 이와 같이 실수를 표현하면

double타입을 배정밀도 부동소수형이라고 부릅니다. 즉 float와 동일한 방식으로 실수를 표현하지만, 더 많은 비트를 사용하여 정밀도와 표현범위가 넓어집니다.

4.4 문자 데이터

문자 타입은 char형으로 1바이트의 크기를 가집니다. 결국 문자도 하나의 숫자 값에 불과합니다. 문자마다 숫자값을 부여하는 표를 만드는데 이것을 문자집합(character-set)이라 부릅니다. 가장 널리 사용되는 것은 영문자와 숫자, 특수기호로 이루어진 아스키 코드입니다. 문자 집합에 정의가 안된 문자는 컴퓨터로 표현할 수 없습니다.

4.2 변수

값은 메모리에 저장됩니다만, 그 값을 이용하려면 무언가 표식, 즉 이름이 필요합니다.

변수/상수 개념 설명

변수 선언: 컴파일러에게 데이터 타입과 변수의 이름을 알려주는 것

변수 정의: 변수를 메모리에서 사용할 수 있도록 공간을 할당하는 것

변수 할당: 값을 저장하거나 변경하는 것

메모리 주소는 운영체제가 알아서 수행하는 것이므로 메모리 번지수는 매번 달라질 수 있다.

C언어에서는 변수를 선언할 때 다음과 같이

4.2 형 변환1

## 4장 요약

### 심화학습

### 과제

적절한 데이터타입 선택하기 문제

### 도움 자료

### 전처리기

전처리기(preprocessor)는 컴파일러가 소스 코드 파일 하나를 컴파일하기 전에 소스코드에 변형을 가하는 동작을 수행합니다.

조건 컴파일: #if, #ifdef, #ifndef #else #elif

교체/치환 #define, #undef, # 및 ##

포함: #include

정의된 구현 동작 #pragma

파일 이름 및 줄 번호: #line

미리 정의된 매크로

\_\_STDC\_VERSION\_\_

\_\_FILE\_\_

\_\_LINE\_\_

\_\_DATE\_\_

\_\_TIME\_\_

자주 사용하는 #define 예

#define FUNCTION(name, a) int fun\_##name(int x) { return (a) \* x; }

FUNCTION(quadruple, 4)

FUNCTION(double, 2)

#undef FUNCTION

#define FUNCTION 34

#define OUTPUT(a) puts( #a )

int main(void)

{

printf("quadruple(13): %d\n", fun\_quadruple(13) );

printf("double(21): %d\n", fun\_double(21) );

printf("%d\n", FUNCTION);

OUTPUT(billion); // note the lack of quotes

}

선언문(Declaration)

한 개 이상의 식별자(identifier)를 프로그램에 도입하고, 그 의미와 속성을 지정한다.

스코프…

각 선언문은 구문과 마찬가지로 세미콜론으로 끝난다.

한정자: 타입 한정,

한 개 이상의 식별자를 사용하고 싶을 경우에는 쉼표(콤마)를 이용

Int a, b=2;

재선언

Extern에 대한 설명 필요.