# 1장. 프로그래밍의 세계로 오신 것을 환영합니다.

**학습 목표**

프로그래밍이란 무엇인지 알아보고, 프로그래밍 언어가 어떻게 발전해 왔는지를 살펴본다. 또한 C언어로 프로그래밍을 배우면 어떤 장점이 있는지를 이해한다.

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 프로그램이란 무엇이며, 프로그래밍 언어가 필요한 이유를 설명할 수 있다. |  |  |  |  |  |
| 현대 범용 컴퓨터가 어떤 장치로 구성되었는지를 설명할 수 있다. |  |  |  |  |  |
| 여러 프로그래밍 언어가 발전해 온 역사를 이해한다. |  |  |  |  |  |
| C언어의 특징과 C언어를 배웠을 때 무엇을 얻을 수 있는지를 이해한다. |  |  |  |  |  |

프로그래밍programming이란 컴퓨터에서 실행되는 프로그램program을 만드는 작업입니다. 여러분은 프로그램을 만듦으로써 데이터를 컴퓨터에 전달하여 원하는 대로 데이터를 처리하여 결과를 제공하도록 작업을 지시할 수 있습니다. 그럴려면 여러분의 의도를 컴퓨터가 이해할 수 있는 형태로 설명하는 방법이 필요한데, 이를 위해 과학기술자들이 정의한 언어가 프로그래밍 언어Programming Language입니다.

컴퓨터가 어떻게 작업을 수행하는지를 살펴보면 우리가 프로그래밍 언어를 사용하여 올바른 지시를 내리는 데 도움이 됩니다. 이번 장에서는 컴퓨터와 프로그램, 프로그래밍 언어의 관계를 살펴봅니다.

## 1-1 프로그램 개념의 탄생

### 프로그램과 요리 기계

여기, 아주 큰 요리 기계가 있습니다. 기계의 왼쪽에는 요리 재료를 넣어두는 창고가 있고, 오른쪽에는 완성된 요리가 나오는 주방 창문이 있습니다. 창고에 필요한 요리 재료를 넣고, 원하는 음식의 레시피를 전달하면 잠시 후 주방 창문으로 맛있는 요리가 나옵니다.

어떻게 이게 가능할까요? 요리 기계 안에는 무엇이든 만들 수 있는 만능 요리사가 대기하고 있습니다. 요리사는 레시피에 따라 창고에서 필요한 요리 재료를 꺼내 순서대로 음식을 만듭니다. 조리를 마친 요리사는 주방 창문으로 손님에게 요리를 전달합니다.

우리가 만들 프로그램도 이와 비슷한 과정을 거칩니다. 입력 장치를 통해 작업에 필요한 정보(데이터)를 전달하면 저장장치인 메모리에 저장됩니다. 저장된 데이터를 어떤 식으로 다뤄야 하는지를 지시하는 프로그램 코드 역시 메모리에 저장됩니다. 메모리에 저장된 프로그램의 순서에 따라 필요한 데이터를 메모리에서 꺼내 온 다음 원하는 연산 동작을 수행한 후 그 결과를 다시 메모리에 저장합니다. 연산 동작을 모두 마쳤으면 출력 장치를 통해 사용자에게 최종 결과를 알려줍니다. 이 모든 과정을 요리사인 CPU(중앙 처리장치)가 조율합니다.

만화 영화, 스케치, 사람이(가) 표시된 사진

자동 생성된 설명

|  |  |
| --- | --- |
| 요리기계 | 컴퓨터 프로그램 |
| 요리 재료 | 입력 데이터 |
| 요리 레시피 | 프로그램 |
| 주방 | 메모리 |
| 요리사의 손 | 산술/연산 장치 |
| 완성된 요리 | 출력 데이터 |

현대 범용 컴퓨터는 다양한 목적의 작업을 수행하기 위하여 다음 그림과 같이 5개의 구성 요소로 이루어져 있습니다. 이 구조를 제시한 사람의 이름을 따서 폰 노이만 구조라고 부릅니다. 폰 노이만Von Neumann은 프로그램으로 동작하는 현대 컴퓨터의 기본 개념을 설계한 과학자로 일컬어집니다.

스케치, 텍스트, 그림, 라인 아트이(가) 표시된 사진

자동 생성된 설명

그림 1.1 폰 노이만 구조

(출처<https://joone.net/2016/11/15/%ec%95%a8%eb%9f%b0-%ed%8a%9c%eb%a7%81%ea%b3%bc-%ed%8f%b0%eb%85%b8%ec%9d%b4%eb%a7%8c/>)

폰 노이만 구조에서 가장 중요한 역할을 맡고 있는 것이 바로 중앙 처리 장치, 즉 CPU입니다. CPU는 메모리에 저장해둔 데이터와 프로그램 코드를 다루며, 입력 장치를 통해 데이터를 받고, 출력 장치를 통해 데이터를 내보냅니다. 이와 같은 개념이 정립되면서 컴퓨터는 특정 목적을 위한 전용 계산기에서 여러 목적으로 사용 가능한 범용 연산 장치로 거듭나게 되었습니다.

CPU의 내부를 자세히 들여다 보면 산술/논리장치ALU, 레지스터, 제어 장치로 이루어 집니다. 폰 노이만 구조의 축소판처럼 느껴집니다.

산술/논리장치는 비트 수준에서 산술 연산이나 논리 연산을 수행합니다. 즉, 두 값을 더하는 것과 같은 수치 계산이나 데이터 비트를 조작합니다.

레지스터는 계산 성능을 빠르게 수행할 수 있도록 필요한 데이터를 고속으로 주고받을 수 있게 해주는 장치입니다. RAM 메모리에서 데이터를 꺼내오는 것도 CPU 입장에서는 느린 작업이기 때문에, 산술/논리 장치 옆에 고속의 메모리 장치를 둔 것입니다. 여러 용도의 레지스터가 존재하는데, 단기 기억을 담당하는 데이터 레지스터data register, 각 CPU의 연산마다 접근할 메모리 주소를 알려주는 주소 지정 레지스터addressing register, 다음 실행할 명령을 기억하는 프로그램 계수기 레지스터program counter register 등이 있습니다.

제어장치는 CPU에 정의된 고유의 코드 집합에 따라 ALU, 레지스터, 메모리에서 어떤 명령을 실행해야 하는지를 결정합니다. 보통 명령어를 메모리에서 가져와서fetch 해석한 후decode 실행execute하는 과정을 끊임없이 반복한다고 볼 수 있습니다.

## 1-2 프로그래밍 언어의 등장

### 기계어

컴퓨터는 어떤 작업이든 불평 없이 아주 빠른 속도로 반복해서 수행합니다. 하지만 컴퓨터를 이루고 있는 각 장치는 0과 1로만 대화를 나눌 수 있습니다. 즉, 컴퓨터에 알려줄 작업 레시피는 모두 0과 1로 이루어 집니다. 이 언어를 기계어라고 부릅니다.

Dr.Ties: 폰 노이만 구조 이전에는 어떤 식으로 프로그래밍을 작성했을까요?

힌트: 하드와이어링, 펀치 카드, 기계어?

사람이 컴퓨터에 일을 시키려면 기계어를 공부해야만 했습니다. 하지만 0과 1로만 이루어진 기계어는 사람이 이해하기엔 너무 어렵고 불편했습니다. 또한 기계마다 0과 1을 사용하는 방식이 다릅니다. 그러므로 사람의 의도를 텍스트 문서로 작성하면 이를 컴퓨터가 알아 들을 수 있는 기계어로 번역해 줄 방법이 필요했습니다. 이 고민을 해결해 준 것이 바로 프로그래밍 언어입니다.

### 저수준 언어

첫 프로그래밍 언어는 1950년대 초에 등장한 어셈블리어였습니다. 어셈블리어는 기계어처럼 0과 1로 이루어지진 않았지만, 여전히 사람보다는 컴퓨터가 이해하기에 편리한 언어였습니다. 어셈블리어로 작성된 코드는 컴퓨터의 CPU가 사용하는 언어를 영어 단어로 나타내는 정도였습니다.

코드 1.1 X86 어셈블리 코드 예제

push rbp ;

mov rbp, rsp ;

sub rsp, 32 ;

… … ;

mov [rbp-8], rdx ;

어셈블리 코드는 실행될 컴퓨터가 어떤 CPU를 사용하는가에 따라 사용하는 코드가 달라집니다. 똑같은 작업이라 하더라도 CPU가 다르면 프로그램을 처음부터 새로 작성해야 했습니다. 즉, 똑같은 요리의 레시피를 전달하더라고 한국인 요리사에게 한국어로, 미국인 요리사에게는 영어로 전달하는 셈입니다. 이와 같은 프로그래밍 언어의 특징을 CPU 아키텍처에 종속적이라고 표현하며, 이런 특징을 가진 언어를 저수준 언어(low-level language)라 부릅니다.

지금도 CPU는 인텔이 개발한 32비트 기반의 X86 아키텍처, 이를 64비트로 확장하여 AMD가 개발한 X64 아키텍트, ARM에서 만든 ARM 아키텍처 등으로 나뉩니다. 어셈블리로 프로그램을 작성하려면 실행될 CPU 아키텍처에 따라 다른 명령어 코드를 사용해야 하며, 결국 프로그래밍 언어로 작성된 코드의 재사용성이 떨어지게 됩니다.

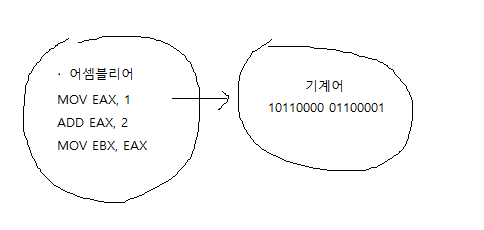


그림 1.2 어셈블리어에서 기계어로의 번역

DrTies: 기계어에서 말하는 기계는 컴퓨터를 말합니다. 세계적인 컴퓨터 기업 IBM도 International Business Machine, 즉 국제 업무 기계라는 뜻입니다.

### 고수준 언어

하드웨어의 발전과 더불어 프로그래밍 언어도 발전을 거듭하면서 기계가 이해할 수 있는 동작을 설명하기 보다는 사람의 의도를 보다 쉽게 표현할 수 있는 모습으로 바뀌게 됩니다. 이전에는 요리사에 초점을 맞춰 레시피를 작성했다면, 이제는 요리에 관심을 둔 레시피를 작성하게 되었습니다.

1960년을 전후로 특정 목적에 적합한 프로그래밍 언어들이 등장합니다. 과학 계산용으로 설계된 포트란(FORTRAN), 일반 금융 및 행정 처리용으로 설계된 코볼(COBOL) 등이 등장합니다. 이런 언어를 고수준 언어(high-level language)라고 부릅니다.

고수준 언어는 각자의 기호 체계를 사용하여 의도, 즉 컴퓨터가 수행해야 할 작업을 표현합니다. 따라서다른 CPU를 사용하는 컴퓨터에서 동작할 프로그램이더라도 같은 작업을 수행한다면 새로 코드를 작성할 필요가 없게 되었고, 사람의 의도를 보다 쉽게 표현할 수 있게 되었습니다.

이후에도 프로그래밍 언어는 사람의 의도를 보다 쉽게 표현할 수 있도록 발전합니다. MZ세대, 알파 세대처럼 프로그래밍 언어도 특징에 따라 세대Language Generation를 구분하기도 합니다. 다음 표는 각 세대별 언어의 특징입니다. 하지만 이러한 세대를 딱 떨어지게 나눠지지는 않습니다만 발전되는 방향을 이해하는데 도움이 됩니다.

표 1.1 컴퓨터 프로그래밍 언어 세대

|  |  |  |
| --- | --- | --- |
| 세대 | 특징 | 대표 언어 |
| 1세대 | 컴퓨터마다 다른 언어를 사용. 0과 1로 작성함. | 기계어 |
| 2세대 | 어셈블리어로 작성하면 기계어로 1대 1 변환  기계마다 다른 명령어 세트를 가짐. | 어셈블리어 |
| 3세대 | 영어와 유사한 일련의 단어로 지시사항을 작성함.  기계어로 번역하는 프로그램(컴파일러)이 필요. | C. 포트란, 코볼, 파스칼 |
| 4세대(4GL) | 특정 분야의 프로그래밍에 특화된 언어 | SQL, R |
| 5세대(5GL) | 주어진 제약조건을 사용하여 문제를 해결  제약조건 기반 및 논리 프로그래밍 언어 | Prolog, Mercury |

## 1-3 C 언어의 등장

오늘날 IT 기술을 이끌고 있는 지역을 떠올리면 미국 서부 샌프란시스코 인근의 실리콘밸리를 떠올리지만, 1960년대 기술은 미국 동부 지역을 중심으로 발전했습니다. 최초의 컴퓨터인 에니악(ENIAC)도 미국 동부에 위치한 펜실베니아 주립대에서 만들어졌습니다. 당시 뉴저지주에 위치한 벨 연구소에서는 컴퓨터를 동작시킬 운영체제를 어셈블리어로 개발합니다.

1960년대에 벨 연구소는 시분할 운영 체제인 멀틱스를 개발 중이었습니다. 멀틱스 개발 프로젝트는 우여곡절 끝에 완성은 되지만, 계획된 일정보다 늦게 개발되었습니다. 이 프로젝트에 참여했던 데니스 리치와 켄 톰슨은 PDP-11 컴퓨터에서 동작하는 프로그램을 만드는 데 사용했던 B언어를 개선하여 1973년 프로그래밍 언어의 기본 기능을 구현합니다. B 언어를 기반으로 만들어진 언어라 그냥 C 언어라 불렀습니다.

DrTies: C 언어를 만들었을 때 데니스 리치는 31살, 켄 톰슨은 30살이었어요. 켄 톰슨은 2006년부터 구글에 근무하면서 Go 언어의 개발에 참여했습니다.

데니스 리치와 켄 톰슨이 개발한 C 언어는 운영체제의 핵심인 커널Kernel까지 작성할 수 있을 정도로 강력하면서도 다양한 목적의 프로그램을 개발할 수 있었습니다. C언어를 사용하여 멀틱스의 후속 운영체제인 유닉스 커널과 많은 유틸리티들이 다시 작성되기 시작했습니다. C언어로 운영체제를 만들게 되면서, 다양한 CPU를 사용하는 컴퓨터에 C언어가 이식되었습니다. 즉, C언어만 동작할 수 있으면 사용되는 CPU는 달라도 C언어로 만든 프로그램을 모든 컴퓨터에서 사용할 수 있게 되었습니다. 이 과정에서 C 언어는 사실상의 업계 표준 프로그래밍 언어의 자리를 차지하게 됩니다.

DrTies: 보다 상세한 C언어의 탄생 이야기가 궁금하시면 <컴파일러 개발자가 들려주는 C 이야기> (페테르 판데르린던 저, 인사이트) 1장을 읽어보세요.

C언어는 다른 언어에도 많은 영향을 미쳤습니다. 데이터와 동작 코드를 구조적으로 결합하여 하나의 객체로 만들어 보다 안전하게 코드를 사용하자는 객체지향 프로그래밍 언어의 조상이 되었습니다. C언어를 기반으로 객체 지향 프로그래밍 언어를 만든 것이 C++입니다. 마이크로소프트는 이를 더 확장하여 닷넷 프레임워크에 맞게 C#언어를 개발하게 됩니다.

## 1-4 C언어 학습의 장점

서점을 방문해 보신 적이 있나요? IT 분야를 방문해 보면 정말 다양한 프로그래밍 언어가 존재합니다. 그럼에도 불구하고 나온 지 만들어진지 40년이 넘은 C언어를 배워야 하는 이유는 무엇일까요? 왜 선배 개발자들은 프로그래밍을 공부한다고 하면 C 언어부터 공부해 보라고 권하는 것일까요?

**1. 컴퓨터의 기본 구조에 충실한 언어**

C언어의 탄생부터 운영체제와 밀접한 관련이 있습니다. 운영체제를 만드는데 사용되는 C언어는 컴퓨터의 동작 방식부터 제어까지 다룰 수 있는 시스템 프로그래밍 언어로 부족함이 없습니다. 파이썬 처럼 일부 프로그래밍 언어는 C언어로 개발됩니다. 하드웨어와 직접 동작하는 부분이 많으므로 성능이 뛰어납니다. 안드로이드 운영체제 역시 대부분이 C언어로 개발되었습니다.

**2. 간결한 언어**

데니스 리치와 브라이언 캐니언이 적은 최초의 C 프로그래밍 언어 책인 “The C Programming Language”는 원서를 기준으로 272쪽에 불과합니다. 프로그래밍 언어의 기본 요소인 변수, 함수를 비롯하여 메모리 관리 등을 직접 사용하면서 익힐 수 있는 간결한 언어입니다.

**3. 높은 이식성을 갖춘 언어**

C언어로 작성된 코드는 높은 이식성을 자랑합니다. 표준에 따라 작성된 C언어는 윈도우, 리눅스, 맥OS 등 주요 운영체제에서 그대로 동작할 수 있습니다. 최근 데이터 분야에서 주목받는 파이썬도 C언어로 개발되었습니다.

**4.참고자료가 풍부한 언어**

오랜 역사를 지닌 언어인 까닭에 프로그래밍 공부에 도움이 되는 다양한 오픈소스 프로젝트, 알고리즘 구현 코드를 손쉽게 구하여 읽어볼 수 있습니다. 또한 C언어의 생산성을 높여주기 위한 표준 라이브러리 및 외부 라이브러리도 풍부합니다.

## 1.5 프로그래밍 언어의 공부 범위

결국 개발자가 원하는 작업을 컴퓨터에게 지시하기 위해서 프로그래밍 언어를 배웁니다. 프로그래밍 언어는 기본적인 데이터를 선언하고 값을 다루는 방법, 반복해서 어떤 동작을 수행하는 방법, 조건에 따라 다른 동작을 수행하는 방법 등을 공통적으로 배우게 됩니다.

프로그래밍 언어로 코드를 작성하면 컴퓨터가 할 수 있는 모든 일을 지시할 수 있습니다. 하지만 모든 것을 하나하나 지시하려면 번거로울 것입니다. 그래서 프로그래밍 언어는 자주 사용되는 기능을 미리 만들어 놓은 표준 라이브러리를 제공합니다.

표준 라이브러리는 주방 도구와 비슷합니다. 전문 요리사라면 칼, 도마와 같은 기본적인 주방도구만 가지고도 원하는 요리를 만들 수 있지만, 믹서기, 오븐 등 자주 사용하는 주방도구가 함께 제공되면 훨씬 편리하게 요리할 수 있습니다. 우리처럼 초보 요리사들도 주방 도구를 사용하면 더 다양한 요리를 만들 수 있습니다.

DrTies: 마치 다른 요리사가 만들어 둔 주방 도구를 자신의 요리에 이용할 수 있는 것처럼 다른 개발자가 만들어 둔 도구를 가져와 이용할 수 있습니다. 언어에서 제공하는 표준 라이브러리와 구별하기 위하여 이런 도구를 서드파티 라이브러리third party library라고 부릅니다.

개발자의 길은 요리사와 비슷한 면이 많습니다. 이 책에서는 기본적인 프로그래밍의 개념과 도구의 사용법을 익힌 후, 표준 라이브러리를 통해 사용할 수 있는 기능의 범위를 넓히는 방식으로 공부합니다.

지금까지 프로그래밍 언어와 C 언어의 탄생에 대해 알아보았습니다. 2장에서는 C언어가 실행파일로 만들어지는 과정을 배우며, 이 과정에서 필요한 개발 환경을 준비합니다.

프로그래밍 언어 학습 방법

프로그래밍 언어를 배운다는 말에는 여러 의미가 포함되어 있습니다. 프로그래밍 언어 자체의 문법은 물론이고, 컴퓨터 하드웨어의 동작 원리부터 운영체제, 자료구조, 알고리즘까지 익혀야 합니다. 이러한 주제들은 서로 밀접한 관계를 맺고 있으므로, 한번에 모든 것을 배우고 이해하는 것은 불가능합니다.

저는 ‘점진적으로 해상도를 높여가는 학습법’을 제안합니다. 인터넷에서 큰 그림을 내려받을 때 처음에는 어렴풋한 윤곽만 표시되었다가 점차 선명해 지면서 온전한 그림이 화면에 그려진 것을 보신 적이 있으신가요?

이 책은 처음에는 개념을 설명하면서 C언어를 통해 프로그래밍의 전체적인 모습을 그리는데 중점을 둡니다. 이후 반복 과정을 통해 각 부분마다 상세한 내용을 살펴봅니다. 처음에는 어렴풋하게 떠오르는 생각이 어느 순간, 아하 하는 순간을 거치면서 흩어져 있던 생각들이 결합되면서 선명해 지는 경험을 체험하실 수 있을 것입니다.

프로그래밍의 즐거운 여정에 함께 하게 되어 기쁩니다.

## 1장 정리

### 과제

만세! 1장에는 복습 내용이 없어요. 이후 장에서는 각 장에서 배운 내용을 이용하여 몇 가지 추가 문제를 풀어봅니다.

### 심화 학습

1장에서는 다음과 같은 내용을 소개했습니다.

* 현대적 컴퓨터 구조라 일컬어지는 폰 노이만 구조와 프로그램의 관계를 살펴보았습니다.
* 프로그래밍 언어의 역할과 발전과정을 살펴보았습니다.
* C언어는 높은 이식성을 갖춘 시스템 언어로, 지금도 프로그래밍의 기본기를 갖추는데 매우 유용한 언어로 인정받고 있습니다.

### 도움이 되는 책

* <컴파일러 개발자가 들려주는 C 이야기> (페테르 판데르린던 저, 인사이트)
* 그래서 컴퓨터는 어떻게 동작하나요, 인사이트
* 유닉스의 역사, 한빛 미디어
* 오픈소스 만화 - 데니스 리치의 일생: <https://joone.net/2017/08/29/10-%eb%8d%b0%eb%8b%88%ec%8a%a4-%eb%a6%ac%ec%b9%98%ec%9d%98-%ec%9d%bc%ec%83%9d/>