시작하는 사람들을 위한
고진감래 C 언어

C Language for Starter

KURUSHINDE OBOERU C-GENGO
by MMGames

Copyright © 2011 by MMGames
All right reserved.
First published in Japan in 2011 by Shuwa system Co., Ltd.

This Korean edition is published by arrangement with Shuwa System Co., Ltd,
Tokyo in care of Tuttle-Mori Agency, Inc., Tokyo through Danny Hong Agency, Seoul.
Korean translation copyright © 2016 by J-PUB

이 책의 한국어판 저작권은 대니홍 에이전시를 통한 저작권사와의 독점계약으로 제이펍에 있습니다.
신저작권법에 의해 한국 내에서 보호를 받는 저작물이므로 무단전재와 복제를 금합니다.

시작하는 사람들을 위한 고진감래 C 언어

초판 1쇄 발행 2016년 11월 30일

지은이 MMGames
옮긴이 박윤미, 박상욱
펴낸이 장성두
펴낸곳 제이펍

출판신고 2009년 11월 10일 제406-2009-000087호
주소 경기도 파주시 회동길 159 3층 3-B호
전화 070-8201-9010 / **팩스** 02-6280-0405
홈페이지 www.jpub.kr / **원고투고** jeipub@gmail.com
독자문의 readers.jpub@gmail.com / **교재문의** jeipubmarketer@gmail.com

편집부 이민숙, 황혜나, 이 슬, 이주원 / **소통·기획팀** 민지환, 현지환
본문디자인 성은경 / **표지디자인** 미디어픽스
용지 신승지류유통 / **인쇄** 한승인쇄 / **제본** 광우제책사

ISBN 979-11-85890-63-0 (93000)
값 24,000원

※ 이 책은 저작권법에 따라 보호를 받는 저작물이므로 무단 전재와 무단 복제를 금지하며,
이 책 내용의 전부 또는 일부를 이용하려면 반드시 저작권자와 제이펍의 서면동의를 받아야 합니다.
※ 잘못된 책은 구입하신 서점에서 바꾸어 드립니다.

제이펍은 독자 여러분의 아이디어와 원고 투고를 기다리고 있습니다. 책으로 펴내고자 하는 아이디어나 원고가 있으신 분께서는 책의 간단한 개요와 차례, 구성과 저(역)자 약력 등을 메일로 보내주세요. **jeipub@gmail.com**

시작하는 사람들을 위한
고진감래 C 언어

C Language for Starter

MMGames 지음 | 박윤미, 박상욱 옮김

Jpub 제이펍

※ 드리는 말씀

- 이 책에 기재된 내용을 기반으로 한 운용 결과에 대해 저/역자, 소프트웨어 개발자 및 제공자, 제이펍 출판사는 일체의 책임을 지지 않으므로 양해 바랍니다.
- 이 책에 등장하는 회사명, 제품명은 일반적으로 각 회사의 등록 상표(또는 상표)이며, 본문 중에는 TM, ©, ® 마크 등을 생략하고 있습니다.
- 이 책에서 사용하고 있는 제품 버전은 독자의 학습 시점에 따라 책의 내용과 다를 수 있습니다.
- 책의 내용과 관련된 문의사항은 옮긴이 혹은 출판사로 연락주시기 바랍니다.
  - 옮긴이: polo149278@gmail.com
  - 출판사: readers.jpub@gmail.com

차례

옮긴이 머리말 ····· xiii
베타리더 후기 ····· xv
이 책에 대하여 ····· xvii

CHAPTER 0 컴퓨터란 무엇인가? 1

0.1 컴퓨터란 무엇인가? ····· 2
0.1.1 현대인과 컴퓨터 2 / 0.1.2 컴퓨터란? 3 / 0.1.3 CPU란? 4 / 0.1.4 메모리란? 5

0.2 프로그램이란 무엇인가? ····· 6
0.2.1 프로그램이란 6 / 0.2.2 단순한 문법 7 / 0.2.3 명확한 의미 9 / 0.2.4 모호한 표현 11

CHAPTER 1 세상에서 가장 작은 프로그램 15

1.1 실행 결과가 없는 프로그램 ····· 16
1.1.1 C 언어의 구조 16 / 1.1.2 함수 생성 방법 17 / 1.1.3 특별한 main 함수 18
1.1.4 프로그램 동작 19

1.2 컴파일러는 번역 소프트웨어 ····· 20
1.2.1 모든 것은 기계어 20 / 1.2.2 프로그래밍 언어의 등장 20 / 1.2.3 C 언어 번역 소프트웨어 22

연습문제 ····· 24

CHAPTER 2 프로그램 작성법 25

2.1 프로그램 작성 방법과 규칙 ····· 26
2.1.1 토큰 26 / 2.1.2 프리 포맷 27 / 2.1.3 다른 규칙 28

2.2 작성 방법 연습 29

2.2.1 함수 작성 방법 29 / 2.2.2 들여쓰기 29 / 2.2.3 주석 30

연습문제 33

CHAPTER 3 화면에 문자 표시하기 35

3.1 문자열 표시 36

3.1.1 필수 사항 36 / 3.1.2 printf 함수 36 / 3.1.3 어디에 쓸까? 37
3.1.4 설명서 넣기 38 / 3.1.5 드디어! 40

3.2 줄바꿈 문자 42

3.2.1 줄바꿈 문제 42 / 3.2.2 이스케이프 스퀀스 43

연습문제 46

CHAPTER 4 수치 표시와 계산 49

4.1 수치 표시 50

4.1.1 문자열과 수치 50 / 4.1.2 수치 표시 51 / 4.1.3 문자열과 함께 표시 52
4.1.4 여러 수치 표시 54

4.2 기본적인 계산 56

4.2.1 계산과 그 결과 표시하기 56 / 4.2.2 사칙 연산자 58 / 4.2.3 복잡한 식 59

4.3 수치의 종류 61

4.3.1 여러 종류의 수치 61 / 4.3.2 실수의 계산 62

연습문제 64

CHAPTER 5 수치 기억과 계산 67

5.1 수치 기억 68

5.1.1 기억의 필요성 68 / 5.1.2 변수라는 메모리 68 / 5.1.3 변수 선언 69
5.1.4 변수 값 대입 71 / 5.1.5 변수를 수치 대신 사용 72 / 5.1.6 대입과 연산을 동시에 74

5.2 변수의 종류 77

5.2.1 자료형 77 / 5.2.2 실수를 기억하는 변수 77

5.3 형 변환 80

5.3.1 정수와 실수의 혼합 계산 80 / 5.3.2 강제적 변환 81

5.4 수치 자릿수 맞추기 ····· 83
5.4.1 정수 자릿수 맞추기 83 / 5.4.2 컴퓨터 형식으로 표시 85 / 5.4.3 실수 자릿수 맞추기 86
연습문제 ····· 88

CHAPTER 6 키보드를 사용한 입력 91

6.1 입력용 함수 ····· 92
6.1.1 입력의 필요성 92 / 6.1.2 scanf 함수 92 / 6.1.3 수치 입력 93 / 6.1.4 여러 개의 입력 95
6.1.5 간단한 시그마 프로그램 96
6.2 입력의 공포 ····· 99
6.2.1 공포의 입력 실수 99 / 6.2.2 구분 기호 실수 99 / 6.2.3 너무 큰 수치 100
6.2.4 문자열의 공포 100 / 6.2.5 해결 방법 101
연습문제 ····· 102

CHAPTER 7 비교와 판단 105

7.1 비교문 ····· 106
7.1.1 조건 판단 106 / 7.1.2 조건 판단문 106 / 7.1.3 비교를 위한 연산자 108
7.2 비교를 위한 연산자 ····· 111
7.2.1 등가 연산자 111 / 7.2.2 관계 연산자 112 / 7.2.3 논리 연산자 114
7.3 여러 처리의 실행 ····· 116
7.3.1 여러 처리의 필요성 116 / 7.3.2 블록문 117
연습문제 ····· 119

CHAPTER 8 조건에 따른 처리 121

8.1 두 가지 조건에 따른 처리 ····· 122
8.1.1 조건이 거짓일 때의 처리 122 / 8.1.2 사용 방법 122 / 8.1.3 보기 쉬운 작성법 123
8.2 세 가지 조건에 따른 처리 ····· 125
8.2.1 조건이 여러 개인 경우 125 / 8.2.2 보기 쉬운 작성법 127
8.3 번호를 사용한 조건에 따른 처리 ····· 128
8.3.1 번호에 대응하는 처리 128 / 8.3.2 조건 외 처리 130 / 8.3.3 같은 처리를 한 번에 132
8.3.4 취약한 판단 133
연습문제 ····· 135

CHAPTER 9 정해진 횟수의 반복 137

9.1 반복문 138
9.1.1 반복 동작 138 / 9.1.2 횟수 표시 139
9.2 루프 동작의 구조 141
9.2.1 초기화와 조건 141 / 9.2.2 무한 루프 142 / 9.2.3 강제 탈출 143
연습문제 145

CHAPTER 10 횟수를 알 수 없는 반복 147

10.1 횟수를 알 수 없는 루프 148
10.1.1 횟수를 구하는 루프 148 / 10.1.2 for문과의 호환성 151
10.2 입력 검사 152
10.2.1 사전 검사와 사후 검사 152 / 10.2.2 입력 검사 153
연습문제 157

CHAPTER 11 함수 생성 방법 159

11.1 사용자 정의 함수 생성 160
11.1.1 프로그램의 부품화 160 / 11.1.2 사용자 정의 함수 생성 161
11.1.3 프로토타입 선언 162 / 11.1.4 사용자 정의 함수 호출 164
11.2 함수에 수치 전달 166
11.2.1 인수를 가진 함수 166 / 11.2.2 함수에 수치 전달하기 167 / 11.2.3 여러 인수 168
11.3 함수에서 수치 반환하기 171
11.3.1 값을 반환하는 함수 171 / 11.3.2 반환값의 제한 173
연습문제 174

CHAPTER 12 변수의 수명 177

12.1 함수 내에서 수명이 끝나는 변수 178
12.1.1 지역 변수의 수명 178 / 12.1.2 같은 이름, 다른 변수 181 / 12.1.3 함수의 독립성 182
12.1.4 정확하게는 블록 내부 183
12.2 마지막까지 살아남는 변수 185
12.2.1 전역 변수의 수명 185 / 12.2.2 모든 함수에서 공유 186 / 12.2.3 지역 변수는 독립 188

12.3 함수 내에서 살아남는 변수 190
12.3.1 정적 지역 변수의 수명 190
연습문제 192

CHAPTER 13 여러 변수의 처리 195

13.1 여러 변수를 한 번에 처리 196
13.1.1 배열의 개념 196 / 13.1.2 배열의 선언 196 / 13.1.3 배열 처리하기 197
13.2 배열 사용법 199
13.2.1 초깃값 대입 199 / 13.2.2 전체 요소 표시 201 / 13.2.3 요소 개수 구하기 202
13.2.4 배열 복사 204
연습문제 207

CHAPTER 14 문자열을 다루는 방법 209

14.1 문자를 다루는 방법 210
14.1.1 문자열을 다루는 변수 210 / 14.1.2 문자를 다루려면 211 / 14.1.3 문자 코드 212
14.1.4 문자 계산하기 212
14.2 문자열을 다루는 방법 217
14.2.1 배열로 만들자 217 / 14.2.2 문자열 초기화 218
14.3 문자열 처리 함수 221
14.3.1 수치로 변환 221 / 14.3.2 문자열 복사 222 / 14.3.3 문자열 연결 225
14.3.4 완벽한 문자열 합성 함수 226 / 14.3.5 문자열 입력 228 / 14.3.6 문자 수 세기 230
14.3.7 문자열 비교 232
연습문제 235

CHAPTER 15 포인터 변수의 구조 237

15.1 메모리 구조 238
15.1.1 메모리? 238 / 15.1.2 초대형 1열 사물함 238 / 15.1.3 CPU 비트 수 239
15.1.4 32비트 사물함 240
15.2 변수와 메모리의 관계 241
15.2.1 변수는 메모리상에 존재 241 / 15.2.2 메모리상의 번호를 표시 242
15.2.3 여러 변수의 번호 244 / 15.2.4 배열 번호 245

15.3 &를 붙여야 하는 변수의 정체 247
15.3.1 &를 붙인 변수의 정체 247 / 15.3.2 모든 것은 값에 의한 호출 247
15.3.3 scanf 함수에서 &를 붙이는 이유 248

15.4 주소를 기억하는 변수 251
15.4.1 포인터라는 단어 251 / 15.4.2 포인터형 251 / 15.4.3 포인터 값 253
15.4.4 포인터 변수 253

15.5 포인터 변수 사용하기 255
15.5.1 포인터 변수의 선언 255 / 15.5.2 주소 대입 256 / 15.5.3 변수 상태 변경 258
15.5.4 포인터는 바로 가기 260

15.6 인수로 정보 전달 261
15.6.1 포인터형 인수 261 / 15.6.2 배열형 인수 263 / 15.6.3 배열형 인수의 특이한 성질 264
15.6.4 주소 전달 266

15.7 배열과 포인터의 이상한 관계 268
15.7.1 배열과 같은 사용법 268 / 15.7.2 포인터만의 작성법 270
15.7.3 오래되고 좋지 않은 포인터 연산 273 / 15.7.4 주소 관련 내용은 잊어라 274

연습문제 275

CHAPTER 16 여러 변수형을 하나로 묶음 277

16.1 다른 형의 변수를 하나로 묶음 278
16.1.1 한 번에 데이터를 처리하고 싶을 때 278 / 16.1.2 구조체의 사용 방법 281
16.1.3 구조체 변수 자체 처리 283 / 16.1.4 구조체의 간결한 선언 285

16.2 구조체 인수 288
16.2.1 구조체로 정보 전달 288 / 16.2.2 구조체에서도 포인터 변수 290
16.2.3 구조체에서도 포인터 인수 292

16.3 구조체의 배열 295
16.3.1 구조체의 배열 295 / 16.3.2 구조체 배열 인수 296

연습문제 297

CHAPTER 17 파일 처리 299

17.1 텍스트 파일 읽고 쓰기 300
17.1.1 파일 처리 300 / 17.1.2 파일 열기와 닫기 300 / 17.1.3 파일에 쓰기 302
17.1.4 파일에서 읽기 304

17.2 바이너리 파일 읽고 쓰기 307
17.2.1 텍스트와 바이너리 307 / 17.2.2 파일 열고 닫기 307 / 17.2.3 파일에 쓰기 308
17.2.4 파일에서 읽기 311
17.3 드래그 지원 313
17.3.1 드래그된 파일명 가져오기 313 / 17.3.2 옵션 분석 315
연습문제 317

CHAPTER 18 매크로 기능 319
18.1 변하지 않는 값의 처리 320
18.1.1 처음부터 마지막까지 변하지 않는 값 320 / 18.1.2 수치에 이름 붙이기 320
18.1.3 문자열에 이름 붙이기 322
18.2 다른 상수 선언 324
18.2.1 const 상수 324 / 18.2.2 enum 상수 325 / 18.2.3 수치 지정 enum 상수 328
18.3 간단한 함수 구현 329
18.3.1 #define 전처리 지시자의 고급 기능 329 / 18.3.2 간단한 함수, 매크로 330
18.3.3 부작용의 공포 333
연습문제 335

CHAPTER 19 동적 배열 337
19.1 자유로운 배열 생성 338
19.1.1 배열의 단점 338 / 19.1.2 메모리 확보 339 / 19.1.3 동적 배열의 요소 개수 확대 342
연습문제 344

CHAPTER 20 여러 소스 파일 347
20.1 최소한의 분할 348
20.1.1 여러 소스 파일을 사용하는 이유 348 / 20.1.2 소스와 헤더 파일 348
20.1.3 최소한의 헤더 파일 349
20.2 분할의 정석 352
20.2.1 변수의 공유 352 / 20.2.2 extern 선언 353 / 20.2.3 헤더 파일의 중복 방지 355
연습문제 358

CHAPTER 21 키보드 입력 더 알아보기 361

21.1 1행 문자열 362

21.1.1 gets 함수로 키보드 입력 362 / 21.1.2 버퍼 오버런 대책 364
21.1.3 문자열에서 수치 등을 추출 366

APPENDUM 부록 369

A 단어와 기호 370

A.1 예약어 370 / A.2 출력 형식 지정자 371 / A.3 입력 형식 지정자 372
A.4 연산자와 우선 순위 374 / A.5 기억 클래스 지정자 377 / A.6 이스케이프 문자 377
A.7 정의된 상수 378

B 표준 라이브러리 함수 목록 380

B.1 입출력 <stdio.h> 380 / B.2 범용 <stdlib.h> 386 / B.3 문자 처리 <ctype.h> 392
B.4 문자열 처리 <string.h> 394 / B.5 숫자 함수 <math.h> 399 / B.6 시간 <time.h> 403
B.7 제어 406

C ASCII 코드 표 408

연습문제 정답 411
찾아보기 441

옮긴이 머리말

요즘 세계적으로 코딩 열풍이 불고 있다. 스티브 잡스는 "생각하는 방법을 알려 주기 때문에 모든 사람은 프로그래밍을 배워야 한다"라고도 강조했고, 많은 개발자가 우대받는 시대가 되었다. 이런 상황 속에서 언어의 기본이라고 할 수 있는 C 언어에 사람들은 많은 관심을 보이고 있다.

C 언어는 1972년 미국에서 개발된 프로그래밍 언어다. 처리 속도나 리소스 등의 관리 측면에서도 다양한 이점이 있어 많은 OS에 널리 사용되고 있는 언어로 알려져 있다. C 언어에 뿌리를 두고 있는 언어인 C#, C++도 Microsoft를 비롯해 다양한 곳에서 사용되고 있음을 알 수 있다.

이 책은 이런 C 언어를, 아니 프로그래밍을 전혀 알지 못하는 초보자도 쉽게 C 언어 프로그래밍을 이해할 수 있도록 구성되어 있다. 깊이 있게 기술을 다룬다기보다 원리를 이해하고 개념을 파악할 수 있도록 구성되어 있고, 개념을 이해한 후 예제 코드를 직접 실행함으로써 C 언어를 완벽하게 이해할 수 있도록 설명하고 있다. 또한, 정확히 이해하고 있는지 한 번 더 머릿속에서 정리할 수 있도록 연습문제를 제공하고 있다.

무엇보다 C 언어를 통해 프로그램의 기본을 확실히 공부할 수 있고, 컴퓨터란 무엇인지 프로그램이란 어떤 것인지 아주 쉽게 풀어 이야기하고 있으며, C 언어에서 가장 어렵다고 하는 포인터에 대해서도 비전공자도 이해할 수 있는 정도로 쉽게 설명하고 있어 C 언어 공부를 포기했거나 다시 시작하는 분, 그리고 C++이나 C#을 하려는 분에게 꼭 추천하고 싶다. 이 책을 통해 C 언어를 마스터하길 바란다.

감사의 글

저의 여섯 번째 출간을 위해 수고해 주신 많은 분들께 정말 감사의 마음을 전하고 싶다. 매번 부족한 저에게 번역의 기회를 주시고 일정을 맞추지 못했지만 너그럽게 이해해 주셨던 제이펍 출판사의 장성두 대표님 그리고 이슬 님께 감사의 말씀을 드리고 싶다. 언제나 꼼꼼한 편집을 해 주시는 성은경 님께도 감사의 말씀을 전하고 싶다. 바쁜 아빠를 위해 응원해 준 첫째 지민이, 둘째 지유에게도 이 자리를 빌려 사랑한다는 말을 전하고 싶다. 마지막으로 이번 작업은 사랑하는 저의 아내와 함께하게 되어 다른 책보다 더욱 보람이 있고 즐겁게 작업했다. 다시금 모든 분께 정말로 감사드린다.

옮긴이 박윤미, 박상욱

베타리더 후기

고승광(플랜티넷)

C 언어는 입사 초기에 사용한 후 정말 오랜만에 다시 보게 되었습니다. 제가 배웠을 당시와 책 분위기가 엄청 달라졌더군요. 덕분에 C 언어, 포인터, 어려움이라는 선입견에서 벗어나서 쉽게 읽을 수 있었습니다. 요즘엔 C 언어를 많이 안 하긴 하지만(심지어 공대에서도 C 언어를 안 배우는 경우도 있더군요), 아직도 서버, 임베디드 쪽에서는 많이 쓰이고 있으므로 쉬운 C 입문서를 읽어 보게 되어 반가웠습니다.

김용현(Microsoft MVP)

쉬운 것과 깊이 있는 것은 트레이드오프 관계이기 때문에 자칫 한쪽으로 치우치기 쉽습니다. 이 책은 나름대로 까다롭다는 C 언어를 쉽게 설명하면서도 기본적이고 필수적인 내용의 전달을 잃지 않게 구성되어 있습니다. 표와 이론만을 좇지 않고 실습과 쉽게 풀어쓴 설명, 별도의 칼럼은 호기심을 가지게 해 줍니다. 가볍게 읽고 좀 더 난이도 있는 책으로 넘어갈 수 있는, 징검다리의 첫 번째 돌이 되기를 소망합니다.

김인숙(가비아)

베타리딩을 하면서 코드를 직접 치면서 따라 하다 보니 잊어버린 C 언어가 다시 떠올랐습니다. 이 책은 처음 C 언어를 접하는 사람에게 유용한 책입니다. 쉽게 설명되어 있어서 이해하기 쉽고, 연습문제로 배운 내용을 복습하고 응용할 수 있습니다. 두세 번 반복하다 보면 항상 씁쓸하게 다가왔던 C라는 언어를 달콤하게 익힐 수 있지 않을까요?

이보라(아주대학교 소프트웨어특성화대학원 석사)

그동안 다양한 C 언어 책을 거쳤습니다. 그 책 중에는 "이 내용은 다음에 배우므로 궁금해하지 말고 일단 넘어가세요"라고 말하는 책이 많았습니다. 그러나 이 책은 그런 책을 접했던 저자가 독자의 입장에서 쓴 책입니다. 허투루 넘어가는 것 없이 새로운 코드가 나오면 그 자리에서 바로 설명해 줍니다. 어려운 과정을 거쳐 C 언어를 마스터하게 된, 말 그대로 '고진감래' 끝에 얻은 저자의 C 언어 학습 노하우를 녹여 낸 책이라 생각합니다.

전찬주(원티드랩)

오랜만에 참 읽기 쉽고 이해하기 쉬운 C 언어 책을 만난 것 같습니다. C 언어로 프로그래밍을 시작하면 초반에는 쉽지만, 뒤로 갈수록 이해하기 어렵고 초심자의 입장에서는 포기하게 되는 경우가 많습니다. 하지만 이 책은 가장 어렵다는 포인터 부분까지 이해하기 쉽게 잘 설명되어 있습니다.

제이펍은 책에 대한 애정과 기술에 대한 열정이 뜨거운 베타리더들로 하여금
출간되는 모든 서적에 사전 검증을 시행하고 있습니다.

이 책에 대하여

이 책의 개요

이 책은 C 언어 프로그래밍 입문서다. 입문서이기 때문에 이 책의 내용으로만 완벽한 게임을 만들거나 편리한 도구나 앱을 만들 수는 없을 것이다. 그러나 프로그래밍을 위한 첫걸음만은 확실하게 디딜 수 있을 것이다.

누가 읽어야 하는가

이 책은 C 언어로 프로그래밍을 시작하려는 분이나 이미 다른 C 언어 책으로 배우려 했으나 성공하지 못한 분들이 C 언어의 주요 기능을 제대로 배울 수 있도록 기획되었다.

일본에서 인기가 많은 인터넷 C 언어 강좌를 고스란히 옮긴 이 책은 그 어떤 책보다 요소 하나하나에 대해 확실하게 설명한다. 이미 다른 C 언어 입문서를 읽어 봤지만 아직도 C 언어가 뭔지 잘 모르겠다고 생각하는 분들이 읽었으면 한다.

이 책을 읽기 위해 필요한 지식

프로그래밍 지식이나 경험은 결코 필요하지 않다. 그러나 컴퓨터를 사용하고 특히 키보드로 문자를 익숙하게 칠 수 있어야 한다. 또한, 개인용이 아니더라도 자유롭게 소프트웨어를 설치할 수 있는 컴퓨터 한 대가 필요하다. 프로그래밍을 공부할 때는 자신이 직접 프로그램을 입력해서 테스트해 보는 것이 가장 중요하다.

이 책을 읽는 방법

이 책은 C 언어의 기본 기능부터 시작해 처음부터 끝까지 순서대로 설명한다. 또한, 앞에서 다룬 이야기를 기반으로 후반부 내용을 설명하므로 흥미가 있는 내용이나 알고 싶은 내용만 발췌해서 보는 식은 맞지 않을 수 있다. 처음부터 끝까지 순서에 맞춰 읽는 것을 권장한다.

그러나 현재 페이지 내용을 모두 외우고 다음 페이지로 넘어가라는 이야기는 아니다. 어쨌든 처음에는 이해하지 못해도 좋으니 한 번은 끝까지 읽어 주길 바란다. 한 번을 끝까지 읽으면 어느 정도 내용이 머릿속에 남을 것이다. 그 상태에서 또 한 번 읽으면 처음 읽을 때 몰랐던 내용을 이해할 수 있을 것이다. 이런 식으로 여러 번 반복하는 것이 효과가 없을 것 같지만 가장 효과적인 방법이다. 그리고 각 장의 마지막에는 연습문제를 추가하였다.

이 책의 구성

이 책에서 프로그램이라고 표시된 부분은 프로그램 코드를 나타낸다. 그 내용을 그대로 입력하면 프로그램으로 동작한다. 그러나 프로그램이 너무 길어지지 않도록 한 부분만 나타낸 프로그램도 있으니 주의하기 바란다.

프로그램 0.1

```
int main(void) {return 0;}
```

또한, 실행 결과라고 표시된 부분은 바로 전 프로그램의 실행 결과를 나타낸 것이다. 여러분이 정확하게 프로그램을 입력했다면 컴퓨터에도 같은 실행 결과가 표시될 것이다. (실제 컴퓨터에서는 해당 프로그램과 관련이 없는 내용도 함께 표시될 수 있다.)

실행 결과

```
Hello world
```

서식이라고 표시된 부분은 '이렇게 써야만 한다'는 C 언어의 문법 규칙이다.

서식

```
형명 함수명 (인수) {처리}
```

그리고 이 책에는 칼럼도 있다. 칼럼의 내용을 알지 못해도 C 언어를 공부하는 데 문제는 없지만, 알고 있으면 C 언어나 컴퓨터에 대한 이해도를 높일 수 있다.

> **Hello, world**
>
> Hello, world는 대부분의 입문서에 등장한다. 그런 의미에서 세계에서 가장 유명한 프로그램이라고 할 수 있다.

다음과 같이 포인트(POINT) 아이콘으로 표시된 부분은 C 언어를 공부하는 데 중요한 내용을 설명한다. 대부분 중요한 내용을 강조하고 반복한다.

C 언어 프로그램의 구조

C 언어 프로그램은 함수가 모여 만들어진 것이다.

다음과 같이 키워드(KEYWORD) 아이콘으로 표시된 부분은 해당 항목을 이해하는 데 도움이 되는 중요 단어를 간단하게 설명한다.

프리포맷

프로그램을 작성에 제한이 적고 자유롭다.

계속해서, 각 장의 개요를 소개한다.

장별 안내

0장 컴퓨터란 무엇인가?

태어나서 처음 프로그래밍을 접하는 독자를 대상으로, 컴퓨터의 특징과 프로그래밍의 개념 등 컴퓨터만의 특별한 개념을 설명한다.

1장 세상에서 가장 작은 프로그램

C 언어를 처음 대면한 독자를 위해 '실행 결과가 없는 프로그램'을 설명한다. 실행 결과를 출력하지 않아도 멋진 프로그램이다.

2장 프로그램 작성법

C 언어는 어려운 제약 없이 누구나 자유롭고 편하게 코드를 작성할 수 있지만, 대부분의 개발자가 사용하는 공통된 규칙을 가지고 있다. 그 규칙에 대해 자세히 설명한다.

3장 화면에 문자 표시하기

화면에 문자를 표시하는 프로그램을 설명한다. 화면에 문자를 표시함으로써 처음으로 계산 결과를 확인할 수 있다. 뒤에서 계속 사용할 내용이므로 잘 기억해 두길 바란다.

4장 수치 표시와 계산

컴퓨터는 계산을 위한 기계다. 4장에서는 C 언어로 계산하는 방법을 설명한다. 계산은 어떤 프로그램에서든 중요한 요소다.

5장 수치 기억과 계산

계산 결과를 저장하거나 다른 계산에 이용하는 방법을 설명한다. 이를 통해 공식만으로 계산할 수 없던 것도 간단하게 계산할 수 있으므로 꼭 기억해 두길 바란다.

6장 키보드를 사용한 입력

계산할 때마다 프로그램을 작성하는 것은 불편한 일이다. 6장에서는 키보드로 수치를 입력할 수 있는 방법을 설명한다. 키보드 입력이 가능해지면 계산 프로그램은 완벽해진다.

7장 비교와 판단

조건에 따라 계산 방법 자체를 바꾸는 내용에 대해 설명한다. 모든 수치를 판단한다는 개념이 어려울 수 있겠지만, 공부해 둔다면 여러 종류의 프로그램을 작성할 수 있다.

8장 조건에 따른 처리

7장에서 공부한 프로그램 작성법을 기반으로 보다 편리하고 알기 쉬운 작성법을 설명한다. 이 방법으로 어떤 계산이든 생각한 대로 처리할 수 있다.

9장 정해진 횟수의 반복

계산을 몇 번이고 반복하는 방법에 대해 설명한다. 예를 들어, 컴퓨터는 수만 번의 반복 계산도 수행하기 때문에 반복 프로그램을 익히면 프로그램이 훨씬 편리해진다.

10장 횟수를 알 수 없는 반복

계산해 보지 않으면 몇 번 반복되는지 알 수 없는 경우를 설명한다. 공식은 알 수 없지만 몇 번이고 계산하면 반드시 답이 나오는 계산에 컴퓨터를 사용하여 답을 얻을 수 있다.

11장 함수 생성 방법

프로그램 일부를 부품화하는 방법을 설명한다. 프로그램을 서로 연결할 수 있으며, 대규모 프로그램에서는 필수 기술이므로 꼭 기억해 두어야 한다.

12장 변수의 수명

5장에서 변수에 대해 이미 다뤘지만, 11장에서 함수를 공부했으므로 그에 따른 변수 처리 방법을 설명한다. 프로그램을 부품으로 만들기 위해 아주 중요한 개념을 설명한다.

13장 여러 변수의 처리

5장에서 변수를 하나하나 별도로 처리했지만, 13장에서는 많은 변수를 한 번에 계산하는 방법을 설명한다. 9~10장에서 공부한 반복처럼 이를 이용하면 편리한 계산을 할 수 있다.

14장 문자열을 다루는 방법

지금까지는 수치만 다뤘지만 이제 프로그램으로 문장을 처리하는 방법을 설명한다. 프로그램으로 문자를 합성할 수 있게 되면 프로그램의 활용도는 더욱 넓어진다.

15장 포인터 변수의 구조

C 언어에서 가장 어렵다고 하는 포인터에 대해 설명한다. 이 책에서는 다른 입문서처럼 포인터의 겉만 설명하지 않고, 포인터의 기능과 정체를 철저하게 파헤친다.

16장 여러 변수형을 하나로 묶음

종류가 다른 변수를 한 번에 계산하는 방법을 설명한다. 지금까지는 계산 결과가 여러 종류일 경우 많은 변수를 사용했지만, 이를 한 번에 처리하면 프로그램이 정말 읽기 쉬워진다.

17장 파일 처리

계산 결과를 디스크에 파일로 저장하는 방법을 설명한다. 파일로 저장하면 의외로 간단하게 자신이 만든 프로그램의 계산 결과를 엑셀에 저장할 수 있다.

18장 매크로 기능

정해진 수치나 공식을 이해하기 쉽게 작성하는 방법을 설명한다. 지금까지는 변수나 함수로 처리한 수치와 수식도 매크로를 사용하면 보다 간단하고 이해하기 쉽다.

19장 동적 배열

13장에서 여러 변수를 한 번에 처리하는 방법을 공부했지만, 여기서는 요소의 개수를 자유롭게 조절하는 방법을 설명한다. 규칙만 잘 지키면 꽤 쉬우므로 꼭 기억해 두자.

20장 여러 소스 파일

11장에서는 함수라는 형태로 프로그램을 부품으로 만들었지만 여기서는 함수보다 더 높은 부품화 방법에 대해 설명한다. 이를 통해 길게 작성된 프로그램을 자유롭게 결합할 수 있다.

21장 키보드 입력 더 알아보기

6장을 보충하는 내용이다. 입력 실수를 방지하기 위해 키보드를 이용한 안전한 입력 방법을 설명한다.

부록

C 언어에 대한 간단한 레퍼런스 모음이다. 단위와 기호, 표준 라이브러리 함수, ASCII 코드로 구성되어 있다.

컴파일러 소개

이 책은 프로그래밍 입문자를 위해 C 언어를 설명한다. 그러나 영어도 그렇지만 단순히 문법이나 명령어를 외운다고 해서 C 언어로 프로그래밍을 할 수 있는 것은 아니다. 그래서 실제로 프로그래밍을 만들어 보거나 계속 연습하는 것이 가장 중요하다.

여기서 연습이란, 프로그램을 실제로 만들어 보고 그 프로그램을 동작시켜 보는 것이다. 그러기 위해서는 프로그램을 작성하는 소프트웨어와 프로그램을 동작시키는 소프트웨어가 필요하다. C 언어 프로그램을 동작시키는 소프트웨어를 컴파일러(compiler)라고 한다. 컴파일러를 사용하면 C 언어로 만들어진 프로그램을 동작시킬 수 있다.

세상에는 수많은 컴파일러가 있다. 이 책에서는 범용 C 언어를 설명하고 있으므로 어떤 C 언어 컴파일러에서도 동작할 것이다. 여러분 중에는 학교나 회사에서 C 언어를 공부할 수도 있으며, 아마도 학교나 회사의 컴파일러를 사용하게 될 것이다.

그리고 집에서 개인적으로 공부하고 싶은 독자도 있을 것이다. 그러나 아직 개인 컴파일러가 없는 분이 많을 것이다. C 언어를 공부하기 위해서는 자신의 손으로 직접 입력한 프로그램을 컴파일러를 사용해서 동작시켜 봐야 한다. 이를 위해 여기서는 C 언어 공부를 위한 추천 컴파일러를 아주 간단하게 소개한다.

앞서 언급한 대로, 이 책에서는 범용 C 언어를 중심으로 설명하므로 여러 종류의 컴파일러 설치 방법이나 사용 방법 등을 따로 다루지 않고 간단한 소개로 대신하도록 한다.

또한, 요즘은 편집기(프로그램을 작성하는 소프트웨어)와 컴파일러(프로그램을 동작시키는 소프트웨어)의 기능을 모두 가진 소프트웨어가 일반적이다. 그 소프트웨어를 통합 개발 환경 혹은 IDE(Integrated Development Environment)라고 부른다.

Eclipse IDE for C/C++ Developers

설치 환경: Windows, Linux, Mac OS X

다운로드: http://www.eclipse.org/downloads/packages/eclipse-ide-cc-developers/mars1

공개 소프트웨어 기반의 개발 플랫폼으로 C 언어, Java 등 다양한 언어를 지원하는 범용 응용 소프트웨어 플랫폼이다. C/C++, Java, PHP 패키지 등을 제공한다.

Microsoft Visual Studio Community 2015

설치 환경: Windows

다운로드: https://www.visualstudio.com/ko-kr/products/visual-studio-express-vs.aspx

Windows 업체인 Microsoft 통합 개발 환경이다. 커뮤니티 버전이지만 매우 강력한 개발 환경 도구다. 초보자는 다루기 어려울 수 있지만 회사나 학교에서 사용한다면 이 도구를 사용하는 것도 좋을 것이다.

Apple Xcode 7

설치 환경: Mac OS X

다운로드: https://developer.apple.com/xcode/download

Mac 업체인 Apple의 통합 개발 환경이다. Apple답게 개발자를 위한 소프트웨어이면서 사용자 인터페이스를 매우 강조한 것이 특징이다. iPhone 애플리케이션 개발에도 사용되므로 향후 iPhone 애플리케이션 개발을 염두에 두고 있다면 이 도구를 공부해 두는 것이 좋다.

FSF GNU Compiler Collection

설치 환경: Linux 등

다운로드: https://gcc.gnu.org

Unix용 컴파일러로 오랫동안 커뮤니티에서 개발된 컴파일러다. 많은 플랫폼을 지원하는 것이 가장 큰 특징이다. Linux/Unix 계열의 OS를 사용한다면 이 컴파일러는 기본으로 설치되어 있을 것이다. 그러나 초보자용이라고 하기 어려우며, IDE(마우스로 조작 가능한 편집기)는 포함되어 있지 않아 별도로 설정해야 하는 어려움이 있다.

Ideone 온라인 컴파일러

설치 환경: 별도의 설치 없이 웹 브라우저에서 사용할 수 있다.

접속 URL: http://ideone.com

별도의 설치 없이 웹 브라우저에서 온라인 컴파일러에 접속하여 C 언어 샘플 코드를 실행할 수 있다. C 언어 이외에 Python, PHP, Perl, Java, Ruby 등을 사용할 수 있다.

CHAPTER

0

컴퓨터란 무엇인가?

C programming

0.1 컴퓨터란 무엇인가?

0.1.1 현대인과 컴퓨터

내가 프로그램을 시작했을 때는 아직까지 컴퓨터가 일반화되지 않았었다. 물론, 계산기를 모르는 사람은 없었으며, 사무실에서는 컴퓨터 전문가가 아니더라도 사무용 컴퓨터나 워드프로세서를 사용하는 것이 별로 드문 일이 아니었고, TV 게임기로 오락을 하는 아이들도 많았다. 그러나 지금과 비교해 보면 그것들은 컴퓨터이기보다는 단순한 계산기, 사무용 기기, 문서 작성기, 장난감 같은 형태의 편리한 기계로 인식되었던 것 같다.

그러나 지금은 Windows로 대표되는 개인용 컴퓨터를 누구나 가지고 있고, 많은 사람이 그것을 컴퓨터라고 인식하며 사용하고 있다. 또한 초등학생 때부터 핸드폰을 가지고 전화를 거는 것이 당연하게 여겨지는 시대이기도 하다. 컴퓨터도 옛날처럼 단순한 화면이 아니라 전문가가 아니더라도 사용할 수 있도록 만들어져 있다.

가깝기 때문에 발생하는 어려움

한편으로, 지금의 컴퓨터는 도구로서 잘 사용되고 있지만 그 본질은 완전히 숨어 있어 이해하기가 점점 더 어려워지는 것 같다. 프로그래밍 공부도 마찬가지로, 평소에는 컴퓨터를 잘 다루더라도 컴퓨터 프로그래밍과 그것을 이해하는 것은 더 어려운 일이다.

컴퓨터를 이해하다

이 장에서는 지금의 컴퓨터가 감추고 있는 컴퓨터의 본질에 초점을 맞추어 설명하겠다. 프로그래밍을 습득할 수 없는 사람 중에는 프로그래밍이 너무 어려워서 힘들다기보다는 처음부터 컴퓨터라는 기계에 대한 오해가 있기 때문이라고 생각한다. 이런 사람들은 프로그래밍을 배워도 제대로 된 프로그래밍을 할 수 없을 것이다. 이 장에서는 컴퓨

터라는 기계의 정의, 프로그램 정의, 그리고 실제로 여러분이 사용하는 컴퓨터의 기능 정의 등을 통해 컴퓨터의 본질을 설명한다.

0.1.2 컴퓨터란?

컴퓨터란 도대체 무엇일까? 컴퓨터를 우리말로 번역하면 전자계산기다. 결국 전기를 사용하여 계산하는 기계라는 것이다.

컴퓨터가 할 수 있는 것은 계산뿐이다. 컴퓨터는 계산만 가능하다. 계산 이외의 것은 아무것도 할 수 없다. 여기서 계산이란 '1+1=2' 같은 것을 말한다.

여러분이 컴퓨터로 네트워크에 연결하거나 화면에 그림을 그리고 문서를 작성하는 다양한 기능은 컴퓨터 계산에 의해 이루어진다.

수학과 다른점

계산이라고 말했지만, 이 기능이 모두 수학 공식만으로 처리되는 것은 아니다. 물론, 수학 공식도 사용되지만 그것만으로 컴퓨터의 복잡한 기능을 구현하는 것은 불가능하다.

컴퓨터는 여러 개의 계산식을 조합할 수 있다. 하나의 계산식으로 불가능한 계산이 있더라도 수백 개, 수천 개, 수만 개의 계산식을 조합해서 계산을 할 수 있다. 물론 답도 하나가 아니다. 수백 개, 수천 개, 수만 개의 답을 기억하는 것이 가능하며, 계산식의 답을 다른 계산에 재이용하는 것도 가능하다.

또한, 계산식 자체를 계산 결과에 맞추어 변경하는 것도 가능하다. 여러분도 공식을 모르는 수학 문제를 풀 때, 먼저 적당한 수를 대입하여 계산해 보고 틀리면 또 다른 수를 대입해서 문제를 푼 경험이 있을 것이다.

수학에서야 이 방법이 잘못된 방법이지만, 현실 세계의 문제는 매우 복잡하여 공식을 간단히 적용시킬 수 있는 문제가 결코 많지 않다. 따라서 적당한 수를 대입하여 답이 나올 때까지 계산하거나 계산 결과에 맞춰 계산식 자체를 계속 바꿔 보는 방법이 꼭 필

요하다. 그리고 수학 세계에서는 이 방법이 문제가 될지 모르지만 컴퓨터 세계에서는 이것이 옳은 방법이다.

계산식도 계산한다

컴퓨터는 수만 개의 계산식과 계산 결과를 기억하며, 수만 개의 계산식 순서를 자유롭게 변경하고, 그 수만 개의 계산식으로 수만 개의 계산 결과를 낼 수 있다. 이것이 컴퓨터의 특징이며, 공식만으로 계산할 수 없는 것을 컴퓨터가 계산할 수 있는 이유다.

0.1.3 CPU란?

0.1.2절의 내용만으로도 이미 컴퓨터에 대한 이해는 충분해 보이지만, 더 나아가 실제로 여러분이 사용하는 컴퓨터와의 관계를 설명하여 더 깊은 이해를 끌어내고자 한다.

여러분이 사용하는 컴퓨터에는 반드시 CPU가 탑재되어 있다(MPU라고도 부른다). 컴퓨터 매장에 가면 카탈로그 사양에 CPU가 반드시 기재되어 있으며, 인텔의 텔레비전 광고에서도 많이 보았을 것이다.

이 CPU가 계산을 하고 있다는 것은 여러분도 알고 있을 것이다. 0.1.2절에서 설명한 것처럼 CPU는 수만 개 이상의 계산식을 인간과 비교도 안 되는 아주 빠른 속도로 계산한다.

얼마나 빠른가?

현재 CPU는 2~3GHz(기가헤르츠) 정도의 속도를 가진다. 참고로 헤르츠는 '1초에 몇 번' 동작할 수 있는지를 나타내는 단위다. 그리고 G(기가)는 '10억'을 나타내는 접두사다. 즉, 여러분이 사용하는 컴퓨터는 1초에 20~30억 번 정도의 계산을 할 수 있다는 의미다.

그리고 CPU는 계산뿐만 아니라 0.1.2절에서 설명한 계산식의 순서를 자유롭게 바꿀 수 있는 능력도 함께 가지고 있다. 0.1.2절에서 설명한 것처럼 계산과 계산식의 순서를 자유롭게 바꾸는 기능이야말로 컴퓨터의 중요한 부분이며, 그 두 가지의 기능을 모두 가진 CPU는 컴퓨터의 중요한 부품이라 할 수 있다.

0.1.4 메모리란?

지금까지 CPU가 컴퓨터에 얼마나 중요한 것인지 설명했지만, 실제로 CPU가 컴퓨터의 역할을 모두 해내기는 역부족이다. CPU에는 계산을 수행하는 기능과 계산 수행 순서를 자유롭게 변경하는 기능이 있다. 그러나 0.1.2절에서는 계산을 수행하고, 계산 결과를 기억하고, 계산식 자체를 변경하는 기능이 컴퓨터의 가장 중요한 부분이라고 설명했다.

그렇다. CPU는 계산을 하는 것은 가능하지만, 계산 결과를 기억할 수는 없다. 수학 시험을 볼 때 시험지 가장자리에 쓰면서 계산하는 것처럼, CPU에도 약간의 계산 결과를 기억하는 기능이 있지만 안타깝게도 조금밖에 저장할 수 없다. 이 저장 공간으로는 CPU가 계산한 수만 개의 계산 결과를 모두 기억할 수는 없다.

얼마나 기억할 수 있을까?

벌써 알고 있듯이, 이 계산 결과를 기억하는 것은 메모리다. 메모리도 컴퓨터의 카탈로그 사양이므로 익숙할 것이다. 요즘 컴퓨터는 2~4GB 정도의 메모리가 일반적이다. 바이트는 0~255 범위의 수 중의 하나를 나타내며, 앞서 말했듯이 기가는 10억이다. (엄밀히 말하면 2^{30}이다) 즉, 여러분이 사용하는 컴퓨터는 대략 세 자리 숫자 20~40억 개를 기억할 수 있는 것이다.

이 메모리가 많으면 많을수록 컴퓨터는 많은 계산 결과를 기억할 수 있다. 계산 결과를 기억하는 것은 같은 계산식을 반복하지 않아도 된다는 의미이기도 하고, 컴퓨터의 속도 향상에도 관련이 있다.

0.2 프로그램이란 무엇인가?

0.2.1 프로그램이란

지금까지 컴퓨터라는 기계의 특징(계산을 실행한다, 계산 결과를 기록한다, 계산식을 변경한다)을 설명했다. 이 설명은 소프트웨어, 즉 프로그램의 개념 중에서도 가장 본질적인 부분에 초점을 둔 것이다. (회로의 이야기나 하드웨어에 대한 이해는 프로그래밍에서 불필요한 지식이므로 따로 설명하지 않는다.)

그래서 지금까지 설명한 컴퓨터의 특징이 곧 프로그램의 특징이기도 하다. 편리한 컴퓨터에 익숙해지면 프로그램에 대해서도 편리한 것을 만들어 내는 좋은 도구로 생각할 수 있을 것이다. 실제로, 프로그래밍에 익숙해진다면 여러 가지 편리한 프로그램을 만들 수 있다. 그러나 프로그램의 본질은 계산이다.

그리고 프로그래밍을 하려면 프로그래밍을 위해 만들어진 프로그래밍 언어를 배울 필요가 있다. 컴퓨터의 세계에는 셀 수 없을 정도로 많은 프로그래밍 언어가 있다(비주류 언어도 포함). 앞으로 여러분이 배울 C 언어는 프로그래밍 언어 중에서도 가장 유명하고 폭넓게 활용된 프로그래밍 언어라고 해도 될 것이다.

인간이 사용하는 한국어, 일본어, 영어 등의 언어를 자연어라고 부르고, 프로그래밍 언어는 인공 언어라고 불린다. 그리고 프로그래밍 언어는 인간과 인간의 소통을 위해 있는 것이 아니라 컴퓨터의 계산을 위해 만들어졌다. 프로그래밍 언어는 감정이나 뉘앙스, 의미나 의도 같은 개념은 하나도 표현할 수 없다. 할 수 있는 것이라고는 그저 계산뿐이다.

그래서 프로그래밍 언어는 계산을 표현하는 것에 특화되어 있다. 계산 이외의 표현은

모두 생략되어 있고, 계산을 위한 표현에 집중되어 있다. 이를 통해 자연어와 프로그래밍 언어가 같은 언어이긴 하지만, 그 표현 방법이 전혀 다르다는 것을 알 수 있다.

결국 프로그래밍 언어와 자연어의 차이가 프로그래밍 언어를 배우는 데 첫 번째 걸림돌이며, 동시에 그 차이를 이해하는 것이 프로그램을 이해하는 데 매우 중요하다고 말할 수 있다. 여기서는 구체적인 예를 통해 자연어와 프로그래밍 언어의 차이를 설명한다.

0.2.2 단순한 문법

첫 번째 차이는 프로그래밍 언어가 자연어보다 문법이 단순하다는 것이다. 프로그래밍 문법은 단어의 배열 순서가 정해져 있다. 여기서부터는 예를 들어 설명하고자 한다. 다음을 보자.

프로그램 0.1

```
REFLECT HELLO
```

이 프로그램은 화면에 "HELLO라는 단어를 보여줘"라는 의미다. 이 프로그램을 우리말로 적으면 이렇게 된다.

프로그램 0.2

```
보여줘 안녕하세요
```

이 역시 화면에 "'안녕하세요'라는 단어를 보여줘"라는 의미다. 두 단어뿐이지만 이것이 프로그래밍 언어의 문법인 것이다. 프로그래밍 언어에서 사용되는 품사는 실제로 동사와 목적어가 대부분이다. 조금 전 프로그램에서는 '보여줘'가 동사, '안녕하세요'가 목적어에 해당 된다고 말할 수 있다. 또 동사를 먼저 가져오는 것은 컴퓨터가 대부분 영어권 나라에서 발전해 왔기 때문이다.

바로 앞의 프로그램을 우리말로 번역하면 다음과 같다.

프로그램 0.3

```
화면에 '안녕하세요'라는 단어를 보여줘.
```

이 두 가지를 비교해 보면 프로그래밍 문법의 단순함을 알 수 있다.

자연어	화면에 '안녕하세요'라는 단어를 보여줘.
프로그래밍 언어	REFLECT HELLO
우리말 표현	보여줘 안녕하세요

프로그래밍 언어에 익숙하지 않은 사람은 이 단순함에 의문을 가질 것이다. 그러나 앞에서 언급한 것처럼, 프로그래밍 언어는 컴퓨터로 계산을 하는 언어다. 즉, 컴퓨터에 계산 명령만 표현할 수 있다면 충분하다.

그런데 이 프로그래밍 언어에는 '보여줘'는 있어도 '화면에'라는 말은 어디에도 없다. 이것은 '보여줘'에 이미 '화면에'라는 의미가 포함되어 있기 때문이다. 어디에 보여줘야 하는지를 명확하게 표현할 때는 다음과 같이 표기한다.

프로그램 0.4

```
보여줘 화면 안녕하세요
```

이 방법이라면 '안녕하세요'를 보여줄 수 있는 대상을 변경할 수 있다.

프로그램 0.5

```
보여줘 프린터 안녕하세요
```

이처럼 변경하면 이번에는 프린터로 인쇄된다.

0.2.3 명확한 의미

앞서 프로그래밍 언어의 문법이 정말 단순하다고 설명했지만, 프로그래밍 언어와 자연어의 차이점은 그뿐만 아니다. 두 번째 차이는 프로그래밍 언어가 자연어와 비교하여 의미가 명확하다는 것이다. 모호한 표현이 전혀 없으며 엄격하고 확실한 의미가 정해져 있다.

앞에서 프로그래밍 언어란 계산 언어라고 설명했다. 그런데 이전 예제에서는 화면에 글자를 표시하라는, 언뜻 계산과 관계없는 것처럼 생각되는 프로그램을 만들었다. 하지만 실제로 '보여줘'라는 명령에는 이미 능숙한 개발자가 만들어 둔 화면에 글자를 표시하기 위한 계산 명령이 가득 담겨 있는 것이다.

프로그래밍 언어는 매우 정확하고 엄격한 계산만 할 수 있다. 그래서 화면에 글자를 표시하라는 단순한 명령에도 (아마도 여러분의 상상을 뛰어넘는) 엄청난 계산이 필요하다. 그러나 전 세계의 프로그래머가 항상 그런 방대한 계산 명령을 쓰는 것은 매우 성가신 일이다. 그래서 '화면에 글자를 보여줘'처럼 누구라도 사용할 수 있는 유형에 대해서는 이미 컴퓨터 제조사의 개발자가 만들어 둔 계산 내용이 내장되어 있다.

그래서 우리는 거기까지는 정확하고 엄격하게 생각하지 않아도 프로그래밍을 실행할 수 있다. 그러나 정확하고 엄격한 계산이 프로그래밍의 본질이며, 이를 이해하지 못한다면 프로그래밍도 이해할 수 없을 것이다. 여기서는 화면에 삼각형을 그리는 예제를 통해 컴퓨터 세계의 치밀함과 엄격함이 일상 생활과 얼마나 다른지 설명한다.

예를 들어, 삼각형을 그린다고 생각해 보자. 사람이라면 삼각형을 그린다는 것만으로 의미가 통하지만, 컴퓨터의 경우 삼각형이라는 언어는 너무 추상적이고 모호하다. 명확하게 어떤 순서로 어떻게 삼각형을 그려야 하는지를 모호함 없이 명확한 순서로 명령해야 한다.

어디서부터 어떻게 삼각형을 그릴지 명령해 보자. 우선, '어디'라는 위치부터 모호함 없이 명령해야 한다. 지구상에서 위치를 완벽하게 지정하기 위해 위도와 경도를 사용하듯이, 여기서는 비슷한 방법으로 화면의 '왼쪽 위에서부터 오른쪽으로 몇 밀리미터, 이어서 그

아래로 몇 밀리미터'와 같이 명령한다. 이 위치를 삼각형 모양이 되도록 정하면 삼각형을 그릴 수 있다. 다음은 가상의 프로그래밍 언어로 나타낸 삼각형 그리기 예제다.

프로그램 0.6

```
LINE 50, 50 - 250, 100
LINE 250, 100 - 120, 160
LINE 120, 160 - 50, 50
```

이를 우리말로 표현하면 다음과 같다.

프로그램 0.7

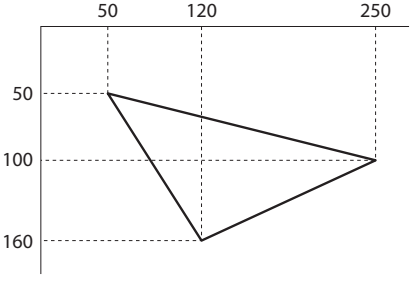
```
선 50, 50 부터 250, 100
선 250, 100 부터 120, 160
선 120, 160 부터 50, 50
```

다시, 이를 우리말로 번역하면 다음과 같다.

프로그램 0.8

```
화면 왼쪽 위에서부터 오른쪽으로 50밀리미터, 아래로 50밀리미터 위치부터,
오른쪽으로 250밀리미터, 아래로 100밀리미터 위치까지 선을 긋는다.
화면 왼쪽 위에서부터 오른쪽으로 250밀리미터, 아래로 100밀리미터 위치부터,
오른쪽으로 120밀리미터, 아래로 160밀리미터 위치까지 선을 긋는다.
화면 왼쪽 위에서부터 오른쪽으로 120밀리미터, 아래로 160밀리미터 위치부터,
오른쪽으로 50밀리미터, 아래로 50밀리미터 위치까지 선을 긋는다.
```

이 순서대로 정확하게 그리면 다음과 같은 그림이 그려진다. 여기서는 1밀리미터를 하나의 점으로 생각하고 있다.



이처럼 매우 명확한 순서로 명령하는 것이 프로그래밍 언어의 특징이다. 그 순서에는 모든 동작을 아주 정확하게 자세히 명령해야 한다.

또 다른 예로, 로봇에게 컵라면을 만들게 하려면 '컵라면을 만들어'라는 말로는 불가능하다. '1미터 전진, 오른쪽으로 90도 회전, 10센티미터 앞으로 손 내밀어' 같은 동작부터 '라면을 꺼내, 주전자에 물을 넣고, 불을 켜고, 그릇에 물을 부어'까지의 모든 동작을 마치 무선 조종기를 조작하는 것처럼 모든 동작을 상세하고 정확하게 지시해야만 한다.

컴퓨터에게는 이러한 동작이 단순한 동작일 뿐이다. 컴퓨터에게 각 개별 동작은 모두 독립된 동작이며, 그 전체 동작이 컵라면을 만드는 일이라고 인식하지 않는다. 단순히 배운 대로 계속 움직이는 정도일 것이다. 만약 그 사이 주전자의 물을 쏟아 버리는 일이 발생하더라도 이는 단순한 동작의 하나이며 그 동작을 실수라고 인식하지 않는다.

0.2.4 모호한 표현

지금까지 설명한 내용만으로도 컴퓨터가 인간에 비하면 매우 엄격하고 정확하다고 생각할 것이다. 하지만 더 놀라운 것은 앞에서 소개한 순서가 컴퓨터 입장에서는 매우 모호한 순서라는 것이다. 사실 '선을 긋는다'는 것이 컴퓨터에게는 이미 모호한 명령이다. 컴퓨터는 직선을 그리는 계산 방법을 모른다. 직선이 어떤 내용의 계산인지를 우리가 컴퓨터에게 알려줘야 한다.

여러분도 알다시피 컴퓨터 화면은 (도트(dot)라 불리는) 점의 모임이다. 컴퓨터에는 화면 하나하나의 점에 대해 어느 점을 그릴지 명령해야만 한다. 만약 그렇게 완벽하고 정확하게 명령을 실행하려면 다음과 같이 점을 그리라는 명령을 계속 반복해야 한다.

프로그램 0.9

```
DOT 50, 50
DOT 51, 50
DOT 52, 50
DOT 53, 50
DOT 54, 51
```

```
DOT 55, 51
DOT 56, 51
     :
```

이 정도가 너무하다고 생각할 수도 있겠지만, 컴퓨터가 요구하는 치밀함은 이 정도로 끝나지 않는다. 실제로 컴퓨터에겐 '점을 그려라'도 모호한 명령이다. 컴퓨터는 점을 계산하는 방법을 알지 못한다. 인간이 미리 컴퓨터에게 점을 그리는 것이 어떤 계산인지 다음과 같이 완벽하고 명확하게 알려줘야만 한다.

프로그램 0.10

```
컴퓨터에 접속되어 있는 00번째 장치가 기억하고 있는 수치 중에서, 32050번째 수치를 1로 설정해라.
```

이것이 터무니없는 명령으로 보이지만 이 계산 방법은 완벽하고 명확하다. 오해의 여지가 절대 없는 명령이다. 컴퓨터에 접속되어 있는 장치 번호는 그 컴퓨터의 회로에 의해서 결정되므로 컴퓨터가 어느 장치인지 알 수 없는 경우는 발생하지 않는다.

너비 640도트의 디스플레이를 예로 생각해 보면, 32050번째는 화면 왼쪽 위에서부터 오른쪽으로 50, 아래로 50의 위치를 의미한다. 00번째 장치에 디스플레이(정확하게는 비디오 카드)가 연결되어 있다고 생각해 보자. 그리고 '1로 설정'하라는 것은 빛을 밝히라는 것이다. 이 컴퓨터에 접속되어 있는 디스플레이는 컴퓨터에 접속되어 있는 00번째 장치의 메모리에 저장되어 있는 수치가 0이라면 그 점을 빛나게 하지 않고, 1이라면 그 점을 빛나게 하도록 만들어져 있다.

프로그래밍 언어 수준에 알맞는, 정확하고 치밀하게 표현된 삼각형을 화면에 그리는 계산은 다음과 같다.

프로그램 0.11

```
컴퓨터에 접속되어 있는 00번째 장치가 기억하고 있는 수치 중에서, 32050번째 수치를 1로 설정해라.
컴퓨터에 접속되어 있는 00번째 장치가 기억하고 있는 수치 중에서, 32051번째 수치를 1로 설정해라.
컴퓨터에 접속되어 있는 00번째 장치가 기억하고 있는 수치 중에서, 32052번째 수치를 1로 설정해라.
```

```
컴퓨터에 접속되어 있는 00번째 장치가 기억하고 있는 수치 중에서, 32053번째 수치를 1로 설정해라.
컴퓨터에 접속되어 있는 00번째 장치가 기억하고 있는 수치 중에서, 32690번째 수치를 1로 설정해라.
컴퓨터에 접속되어 있는 00번째 장치가 기억하고 있는 수치 중에서, 32691번째 수치를 1로 설정해라.
컴퓨터에 접속되어 있는  00번째 장치가 기억하고 있는 수치 중에서, 32692번째 수치를 1로 설정해라.
        :
```

이 명령을 점을 찍는 모든 곳에 반복하는 것이다. 그리고 컴퓨터는 실제로 이런 계산 명령이 몇 억 개라도 모두 순서대로 계산해서 화면에 삼각형을 그려준다.

이와 같이 프로그래밍 언어는 매우 엄격하고 정확한 계산 명령을 수행한다. 그리고 그 계산에서 인간은 의미를 찾으려 하지만, 컴퓨터는 단지 계산만을 실행한다. 결과적으로 화면에 삼각형이 그려지지만, 컴퓨터는 단지 지정된 순서대로 1을 기억하고 있을 뿐이다.

이미 언급한 바와 같이, 실제로 컴퓨터 제조업체의 개발자에 의해 이처럼 엄격하고 정확한 프로그램이 이미 내장되어 있고, 우리는 이처럼 프로그램을 작성하는 것만으로 간단히 직선을 화면에 찍을 수 있다.

프로그램 0.12

```
LINE 50, 50 – 250, 100
```

그러나 컴퓨터에 내장되지 않은 새롭고 편리한 프로그램을 만들려면 우리도 엄격하고 정확한 프로그램을 작성해야만 한다. 생각대로 프로그램을 만드는 것이 어려운 것은 우리가 일반적으로 생각하는 것이 매우 모호하고, 그것을 이렇게 정확하고 엄격한 계산으로 표현하는 것이 어렵기 때문이다.

그래서 프로그램을 반복해서 작성해 봄으로써 실력을 쌓을 수밖에 없다. 또한, 여기서는 컴퓨터를 생각하는 대로 움직이는 것이 이처럼 엄격하고 정확한 계산이 필요하다는 것을 인식하는 것만으로 프로그램 언어의 다양한 특성을 이해하는 데 도움이 될 것이다.

추상화

여러분 중에는 컴퓨터의 엄격함에 놀람과 동시에 벌써부터 머리가 아픈 독자가 있을 것이다. 그러나 이것이 바로 컴퓨터다. 컴퓨터는 처음부터 완벽하게 엄격한 기계였다. 이제 컴퓨터의 엄격함은 충분하다. 오히려 추상화가 더 되었으면 한다.

사실 C 언어는 컴퓨터를 추상화하는 방법 중 하나다. 컴퓨터의 작성 방법대로라면 프로그래밍도 모두 숫자로만 실행해야 하지만, C 언어에서는 데이터에 이름을 붙일 수 있다. 이는 C 언어가 숫자와 이름을 처리하기 위한 표를 자동적으로 준비하는 구조를 가지고 있기 때문이다.

이처럼 숫자로 표현할 수 있는 것을 일부러 표를 만들어 숫자 이외의 방법으로 표현하는 것을 추상화라고 부른다. 프로그래밍 역사는 곧 추상화의 역사이기도 하다. C 언어는 여러 기능을 가지고 있지만, 모든 기능은 컴퓨터의 엄격한 구조를 추상화하기 위해 존재한다.

옛날부터 컴퓨터를 아주 잘 다루는 사람들도 컴퓨터의 엄밀함에 힘들어 했었다. 그래서 컴퓨터를 보다 추상화하기 위한 구조를 만들어 온 것이다. 여러분이 실제로 프로그래밍을 할 때도 숫자를 그대로 사용하는 것이 아니라 항상 인간에게 의미 있는 형태로 표현할 수 있는가를 생각해야 한다. 결국 여러분이 만든 프로그래밍을 사용하는 것은 바로 인간이기 때문이다.

CHAPTER

1

세상에서 가장 작은 프로그램

C programming

1.1 실행 결과가 없는 프로그램

1.1.1 C 언어의 구조

이제부터 C 언어로 첫 프로그램을 만들어 보기로 하자. 그러나 처음부터 여러 기능을 가진 프로그램을 만들 수는 없기 때문에 우선 아무 기능도 하지 않는 실행 결과가 없는 프로그램을 만드는 것부터 시작한다.

C 언어 프로그램을 만들기 위해서는 어떤 지식이 필요할까? 어떤 프로그램이든 그 구조를 모르면 만들 수 없을 것이다. 책장을 만들기 위해 책장의 설계 구조를 알아야 하는 것처럼 C 언어도 그 구조를 파악해야 한다.

그럼 C 언어의 구조란 도대체 무엇일까? 기계를 만들 때는 여러 부품을 조합하여 만든다. 이와 마찬가지로 프로그램에서도 부품을 조합하여 만드는데, 그 부품을 함수라고 부른다. 즉, 여러 함수가 모여 만들어진 것을 C 언어 프로그램이라고 한다.

C 언어 프로그램의 구조

C 언어 프로그램은 함수가 모여 만들어진 것이다.

여기서 주의해야 할 것은 함수가 나열되어 있는 것이 아니라 함수의 집합이라는 점이다. 나열은 순서를 가지고 있지만 집합은 순서가 없다.

C 언어의 함수는 일부를 제외하면 순서 개념이 전혀 없다. C 언어 프로그램 구조는 여러 종류의 함수가 한 덩어리로 만들어진 구조이고, 각각의 함수는 순서에 상관없이 사용할 수 있다.

1.1.2 함수 생성 방법

C 언어 프로그램 구조는 함수의 집합임을 알았다. 그리고 그다음 알아야 할 것은 함수의 구조일 것이다. C 언어 함수는 아주 명확한 구조를 가지고 있다. 다음은 C 언어 함수의 구조다.

서식 함수의 구조

```
형명 함수명 (인수)  {처리}
```

형명이란, 함수가 계산 결과를 반환할 때 사용하는 수치의 종류다. 여기에서는 일단 int라는 형을 기억해 두자. int[1]란 정수를 의미한다.

함수명이란, 말 그대로 함수의 이름이다. 이름을 부여하는 방법에는 다음과 같은 규칙이 있다.

함수명 규칙

1. 알파벳, 숫자, _(언더바)를 사용할 수 있다.
2. 첫 번째 문자에는 숫자를 사용할 수 없다.
3. 미리 정해진 예약어는 사용할 수 없다.

예약어란 C 언어에서 사용되는 키워드다. 그러나 이 예약어는 외울 수 있을 만큼 수가 많지 않다. 이 규칙을 만족한다면 어떤 이름이든 사용 가능하지만, 여기서는 main[2]이라는 이름을 사용하기로 한다.

인수란 함수에게 전달하는 수치의 종류(형)를 말한다. 함수는 전달받은 수치로 계산을 수행하고 그 결과를 반환할 수 있다. 그러나 처음 언급했듯이, 이 절의 목표는 실행 결

1 int: integer(정수)의 약자

2 main: 주된, 가장 큰

과가 없는 프로그램을 만드는 것이다. 실행 결과가 없는 프로그램에 수치를 전달할 필요는 없다. 그래서 이 장에서는 수치가 없다는 것을 나타내는 void[3]를 사용한다.

처리란 그 이름 그대로 처리를 말한다. 앞서 말했듯이 이 장의 목표는 실행 결과가 없는 프로그램을 만드는 것이지만, 함수를 종료시키는 처리를 반드시 해야 한다. 함수를 끝내려면 return[4]문을 사용한다. 또 return문은 계산 결과의 수치를 반환하는 기능이 있지만, 여기서는 실행 결과가 없는 프로그램을 만들기 때문에 일단 0으로 설정한다.

예약된 식별자

C 언어에서는 예약어 외에 예약된 식별자도 이름으로 사용할 수 없다. 예약된 식별자는 내부에서 사용되는 이름으로, 언더바에 대문자가 이어지는 이름이나 C 언어 표준으로 사용되고 있는 이름은 사용할 수 없다.

이를 토대로 프로그램을 만들면 다음과 같은 함수가 완성된다.

프로그램 1.1

```
int main(void) {return 0;}
```

이 프로그램에서 사용되었으나 아직 설명하지 않은 {}(중괄호)와 ;(세미콜론)에 대해서는 뒤에서 설명하도록 한다.

1.1.3 특별한 main 함수

이제 앞서 만든 함수를 사용하여 프로그램을 만들어 보자. 그런데 그전에 꼭 알아 둬야 할 것이 하나 있다. 1.1.1절에서 함수에는 순서가 없다고 설명했는데, 프로그램을 만

3 void: 빈 공간, 하나도 없는, 무효의

4 return: 복귀

들 때 한 가지 문제가 발생한다. 제일 먼저 어떤 함수로 시작할 것인가? C 언어에서는 이 문제를 매우 간단하게 해결하고 있다. C 언어에는 main이라는 이름의 함수가 가장 먼저 동작하도록 정해져 있다.

main 함수

C 언어에서는 main 함수가 가장 먼저 동작한다.

만약 프로그램 안에 main 함수가 어디에도 없다면, 가장 먼저 동작할 함수가 없으므로 그 프로그램은 동작하지 않는다. 거꾸로 말해서 main 함수만 있다면 C 언어 프로그램은 동작 가능하다. 앞에서 만든 함수는 이름이 main이기 때문에 프로그램을 동작시킬 수 있다.

1.1.4 프로그램 동작

그럼 이제 프로그램을 실행해 보자. 앞서 만든 실행 결과가 없는 프로그램이다.

프로그램 1.2

```
int main(void) {return 0;}
```

이 프로그램을 보이는 그대로 입력한다. 가능한 한 제공된 코드를 복사해서 사용하지 말고 프로그램이 익숙해질 수 있도록 키보드를 사용하여 직접 입력한다. 입력 후 바로 동작시켜 보자. 이 프로그램의 실행 결과는 다음과 같다.

실행 결과

```

```

실행 결과에는 아무것도 표시되지 않는다. 목표한 대로 실행 결과가 없는 프로그램이 되었다. 어떤 개발 환경에서는 여러 텍스트가 출력될 수도 있는데, 그것은 프로그램 동작과는 관계없는 텍스트이므로 신경 쓰지 않아도 된다.

1.2 컴파일러는 번역 소프트웨어

1.2.1 모든 것은 기계어

1.1절에서 실행 결과가 없는 프로그램을 작성했다. 이 장에서는 프로그램 동작에 대해 보다 자세하게 설명하도록 하겠다. 여러분이 방금 작성한 프로그램은 일반 문자열에 지나지 않는다. 컴퓨터는 2진수로 동작한다는 말을 들은 적이 있을 것이다. 왜 2진수로 움직이는 컴퓨터가 문자열을 인식하고 동작하는 것일까? 원래 컴퓨터가 이해할 수 있는 것은 2진수로 쓰인 명령뿐이다. 2진수로 쓰인 명령, 즉 기계어 말이다.

프로그램 1.3 기계어 표시

```
00010100110101010100101010
00010101010101010010110100
10101000101010110101010110
00010100101010101001001101
00010100101010101000000101
```

이것은 순서 없이 0과 1을 입력한 내용이지만 기계어도 분명 이런 식으로 표현될 것이다. 세상의 어떤 컴퓨터든 이런 형태의 명령만 인식할 수 있다.

1.2.2 프로그래밍 언어의 등장

컴퓨터가 이해할 수 있는 것은 기계어뿐이다. 실제로 초창기 컴퓨터에는 토글 스위치(toggle switch)를 똑딱똑딱하면서 방대한 수의 0과 1을 입력했다.

그러나 이런 작성 방법은 시간과 손이 많이 간다. 그래서 보다 편하게 작성하기 위해 가장 먼저 생각해 낸 것이 숫자의 자릿수를 줄이는 것이었다.

프로그램 1.4 16진수의 기계어

```
01  85  AD  7F  7C  A4  FA  6B  AD
06  F5  AB  74  7E  DC  18  FA  A4
01  7F  A7  C5  D8  6B  4E  A4  FA
```

이 방식이 이른바 16진수라고 하는 표현 방법이다. 2진수 4행을 16진수에서는 1행으로 표현할 수 있기 때문에 프로그램이 짧아진다. 그리고 이 숫자 하나하나를 기호화한 어셈블러(assembler)도 만들 수 있게 되었다.

프로그램 1.5 어셈블러

```
MOV    AH, BH
ADD    AX, 70
JPN    AF, 01
```

이 기호화로 프로그래밍은 상당히 효율적으로 바뀌었지만, 여전히 보통 사람이 취급하기에 어려운 기호임은 틀림없다. 이를 크게 바꾼 세계 최초의 프로그래밍 언어가 바로 포트란(FORTRAN)이다.

프로그램 1.6 포트란

```
    DO 10 I=1, 10000
      READ *, X
      IF(X, GT, MAX)MAX=X
10 CONTINUE
```

이러한 프로그래밍 언어는 영어 단어와 수식을 사용하며, 일반인도 공부를 한다면 충분히 이해할 수 있다. C 언어도 영어 단어와 수식을 사용한 고급 언어 중 하나다.

고급 언어

인간이 알기 쉽게 쓸 수 있는 프로그래밍 언어를 말한다. 이에 반해, 기계어와 어셈블리를 저급 언어라고 부른다.

1.2.3 C 언어 번역 소프트웨어

고급 언어는 인간이 만들어 낸 언어이므로 컴퓨터는 이해할 수 없다. 그러므로 이것을 컴퓨터가 이해할 수 있도록 번역을 해야 한다. 영어를 한국어로 번역하는 것과 기본적으로 같다. 이 작업은 인간이 수작업으로 하는 것도 가능하지만, 그렇게 되면 기계어로 프로그래밍하는 것과 다를 바 없다. 그래서 기계어로 번역하기 위한 소프트웨어가 존재한다.

번역 방법은 주로 두 종류가 사용되는데, C 언어의 경우에는 전부를 한 번에 번역하는 방법을 사용한다. 이 번역을 컴파일(compile), 번역을 실행하는 소프트웨어를 컴파일러(compiler)라고 부른다.

컴파일러

C 언어로 작성된 문자열 파일을 기계어로 번역하는 소프트웨어

동시 번역

C 언어 이외의 언어 중에는 동시 번역을 하는 유형도 있다. 이러한 방식을 인터프리터(interpreter)라고 부르며, 자유로운 구성이 가능하지만 속도가 느리다.

C 언어 인터프리터

C 언어에도 인터프리터 방식의 환경이 있다. 프로그램을 하나하나 확인하면서 실행할 수 있어 오류 수정이 매우 편리하지만, C 언어의 가장 큰 장점인 속도가 저하되므로 실무 프로그래밍에서는 사용되지 않는다.

컴파일러는 3단계 구조로 동작한다. 먼저, 전처리기라는 소프트웨어로 문자열 조정이 진행된다. 공백, 줄바꿈 결합, 기호 치환 등을 통해 프로그램을 해석하기 쉽게 만든다. 또한, 뒤에서 설명할 #define 전처리 지시자 등도 이 단계에서 처리된다.

다음은 컴파일러에 의해 컴파일이 진행되는데, 이때 최적화가 이루어진다. 같은 C 언어 프로그램에서도 번역 방법에 따라 동작 속도가 달라지기 때문에 컴파일러는 더 빨리 동작할 수 있는 방향으로 번역을 실행한다.

최적화
보다 빠르게 동작하도록 기계어로 번역하는 기능

마지막으로, 번역된 기계어 프로그램이 링커(Linker)라는 소프트웨어에 의해 결합된다. 링커로 결합하는 것을 링크(link)라고 부른다. 링크됨으로써 기계어 데이터가 실행 가능한 파일이 된다. 실행 가능 파일(EXE 파일)이 된 파일은 애플리케이션으로 실행할 수 있다.

이러한 일련의 작업은 완전히 자동화되어 있고, 우리는 특별히 신경 쓰지 않아도 버튼을 누르는 것만으로 동작 가능한 애플리케이션이 만들어진다.

컴파일 순서

다음은 컴파일 순서를 보여 준다. 전처리기, 컴파일, 링크를 합쳐 컴파일이라고 하며, 실행 가능 파일을 생성하는 단계까지 합쳐 빌드(build)라고 한다.

연습문제

개념

1. C 언어 프로그램은 OO의 집합으로 만들어진다. OO은 무엇인가?
2. 문제 1에서 OO의 구조는 네 가지 기능의 조합으로 되어 있다. 네 가지 기능은 무엇인가? 순서와 구분 기호까지 답해라.
3. C 언어는 인간을 위한 언어이기 때문에 컴퓨터는 이해할 수 없다. 이를 위해 C 언어에는 컴퓨터가 이해할 수 있는 언어로 번역하는 소프트웨어가 있다. 그런 소프트웨어를 무엇이라고 부르는가?

프로그램 만들기

4. 실행 결과가 없는 main 함수만으로 이루어진 프로그램을 작성해라.

주관식

5. C 언어에서는 함수에 자유롭게 이름을 지정할 수 있지만, 반드시 하나는 main이라는 이름의 함수여야 한다. 그 이유를 간단히 써라.

CHAPTER

2

프로그램 작성법

C programming

2.1 프로그램 작성 방법과 규칙

2.1.1 토큰

한국어의 문장은 문자, 단어, 문장, 단락과 같이 문장을 구성하고 있는 다양한 요소로 분할하여 볼 수 있다. C 언어 프로그램도 이와 같이 분리할 수 있다.

문법적으로 C 언어 프로그램을 분할했을 때 최소 단위는 토큰(token)이다. 토큰은 단어라고 볼 수 있다. 예를 들어, 다음 프로그램을 보자. (이 프로그램은 1장에서도 작성했었다.)

프로그램 2.1

```
int main(void) {return 0;}
```

이 프로그램을 토큰 단위로 분할하면 다음과 같다.

프로그램 2.2 토큰 분리

```
int
main
(
void
)
{
return
0
;
}
```

토큰을 최소 단위로 정한 이유는 더 분할하게 되면 다른 의미로 바뀌어 버리기 때문이

다. 예를 들어, return을 ret와 urn으로 분할하면 오류가 출력되고 프로그램은 동작하지 않는다.

2.1.2 프리 포맷

C 언어는 프리 포맷(Free Format), 즉 자유 서식이다. 프로그램 작성에 거의 제한이 없기 때문에 자유롭게 작성할 수 있다.

프리 포맷
프로그램 작성에 제한이 거의 없어 자유롭게 쓸 수 있다.

C 언어 프로그램을 작성하는 데 내가 가장 중요하다고 느끼는 규칙은 딱 하나다. 토큰끼리 연결하여 작성하면 안 된다. 예를 들어, 앞서 작성한 프로그램을 다음과 같이 작성해서는 안 된다.

프로그램 2.3

```
intmain(void) {return 0;}
```

이 예제의 문제는 int와 main이 연결되어 있다는 것이다.

그러나 앞서 작성한 프로그램을 보면 알 수 있듯이 예외는 존재한다. 바로, 기호는 연결하여 작성해도 된다는 것이다. 프로그램 2.1에서 (), {}, ; 등은 다른 토큰에 연결되어 있다.

그 이유는 기호가 미리 정해진 한 글자이므로 연결하여 써도 구별할 수 있기 때문이다. intmain은 어떻게 봐도 intmain이라는 토큰으로밖에 보이지 않는다. 그러나 main(void)라면 main, (), void인 것을 알 수 있다.

C 언어 프로그램의 작성 규칙은 아주 간단하다. 일단 기호 이외의 토큰을 연결하지 않고 작성한다는 것만 기억하면 된다. 다음 프로그램은 정상적이지는 않지만, 작성 규칙을 따르고 있어 오류 없이 컴파일이 가능하다.

프로그램 2.4

```
int
main
(
void
)
{
return
0
;
}
```

2.1.3 다른 규칙

C 언어에는 앞에서 설명한 규칙 외에 별도의 작성 규칙이 있다. 먼저 C 언어는 대문자와 소문자를 구별한다. 예를 들어, main, MAIN, maIN은 C 언어에서 전혀 다른 토큰으로 해석된다. 따라서 프로그램을 입력할 때 대문자와 소문자를 주의해야 한다.

또한 C 언어의 문장 끝에는 ; 기호를 사용하게 되어 있다. 방금 전 프로그램에서는 return문 마지막에 ;이 있었다. 이 규칙을 지키지 않는다면 오류가 발생하므로 기억해 두길 바란다.

2.2 작성 방법 연습

2.2.1 함수 작성 방법

2.1절에서 설명한 내용처럼 아무리 C 언어가 프리 포맷이라고 해도 모두가 각자의 방식대로 작성한다면 프로그램은 읽기가 어려워진다. 그래서 C 언어에는 프로그램 작성에 대한 관습이 있다.

앞에서 작성해 온 프로그램은 다음과 같이 모든 내용을 한 줄에 작성하였다.

프로그램 2.5

```
int main(void) {return 0;}
```

그러나 이 작성 방법은 읽기 어려워 좋은 프로그램이라고 볼 수 없다. 함수는 일반적으로 다음과 같이 작성하는 것이 좋다.

프로그램 2.6

```
int main(void)
{
    return 0;
}
```

이렇게 작성하면 함수의 시작과 끝을 쉽게 이해할 수 있다.

2.2.2 들여쓰기

다음 프로그램을 다시 한 번 보도록 하자.

프로그램 2.7

```
int main(void)
{
    return 0;
}
```

return문이 오른쪽으로 밀려 있다. 이것은 들여쓰기라고 불리는 C 언어의 특징적인 작성 방법이다.

들여쓰기
계층을 표현하기 위해 오른쪽으로 밀어서 쓰는 방법

들여쓰기는 계층적 구조를 표현하기 위해 사용된다. C 언어는 {}로 둘러싸인 문장을 오른쪽으로 들여쓰기하는 것이 관습이다. 또한, 오른쪽으로 들여쓰기를 할 때는 탭 키(Tab)를 사용하는 것이 관습이다. 스페이스로 들여쓰기하는 것이 아니라 탭 키를 사용해서 들여쓰기하도록 한다.

최근 개발 환경이나 텍스트 편집기에는 {}를 입력할 때 자동으로 들여쓰기가 되는 기능이 있다. 가능하면 이러한 편집기를 사용하여 자동으로 들여쓰기하는 것이 편리할 것이다.

간편하게 프로그래밍하기

개발자에게 가장 중요한 것은 간편한 프로그래밍이다. 컴퓨터가 할 수 있는 것은 모두 컴퓨터가 하도록 해야 한다. 손이 많이 가는 반복적인 작업들은 컴퓨터에 맡기고 인간은 보다 창조적인 작업을 하도록 한다.

2.2.3 주석

간단한 프로그램이라면 그 내용을 바로 알 수 있지만, 복잡한 프로그램은 그 내용을 한눈에 알기가 어렵다. 이런 경우 프로그램 안에 설명을 달아 두면 프로그램을 쉽게 읽을 수 있다.

C 언어에서는 프로그램에 설명을 달아 두는 기능이 있으며, 이를 주석이라고 한다.

주석

프로그램 안에 적는 설명문. 프로그램 동작에는 아무런 영향을 주지 않는다.

C 언어에서는 /*와 */ 사이에 주석을 쓴다. 이 주석은 프로그램 실행에는 아무런 영향을 주지 않는다.

프로그램 2.8

```
int main(void)
{
    /* 여기가 주석 */
    return 0;
}
```

이와 같이 프로그램 안에 설명을 포함할 수 있다. 또 한 번 말하지만, 주석은 프로그램 동작에 아무런 영향을 주지 않는다.

주석은 여러 행으로도 쓸 수 있으며, 프로그램의 어디든 쓸 수 있다.

프로그램 2.9

```
int main(void)
{
    /*
    이 전부가
    주석입니다.
    */
    return 0;   /* 여기도 주석입니다. */
}
```

주석을 작성하는 데는 특별히 통일된 관습이 없다. 주석을 많이 쓰는 사람도 있고 조금씩 쓰는 사람도 있다. 그러나 길이가 긴 프로그램을 작성할 때 주석을 사용하지 않으면, 자신이 작성한 프로그램 내용을 잊어버리기 쉽다.

앞으로 이 책에서 설명하는 프로그램에는 필요한 부분에 주석을 달도록 한다. 여러분도 직접 작성한 프로그램에는 주석을 달아 두는 것이 좋을 것이다.

C++ 주석 기능

C 언어 확장판인 C++에서는 여기서 소개한 작성 방법 외에 다음과 같은 주석을 사용한다.

프로그램 2.10

```
// 이 행은 모두 주석이다.
```

C++에서는 이처럼 한 행에 주석을 쓸 수 있다. 이 작성 방법은 매우 편리하여 C 언어 컴파일러에서도 사용할 수 있는 경우가 많다.

연습문제

개념

1. C 언어 문법의 최소 단위는 무엇인가?
2. C 언어 프로그램은 자유롭게 작성할 수 있다. 이러한 특징을 무엇이라 부르는가?
3. C 언어는 대문자와 소문자를 구분하는가?
4. 프로그램 안에 탭을 넣는 것으로 계층을 표현하는 작성 방법을 무엇이라 부르는가?
   (사람에 따라 탭 키를 사용하지 않는 경우도 있지만 이는 오래된 방식이다.)
5. 프로그램 안에 쓸 수 있는 설명문을 무엇이라고 부르는가?

프로그램 만들기

6. 실행 결과가 없는 main 함수만의 프로그램을 작성해라. 단, 관습에 따라 작성하고 들여쓰기도 사용해야 한다.