ERIC BAILEY

THE C PROGRAMMING LANGUAGE

Contents

Chapter One 5

Hello, world! 5

Fahrenheit-Celsius table 5

Exercise 1-3 6

Exercise 1-4 6

Exercise 1-5 7

The `main` *function* 7

Copy 7

Exercise 1-9 8

Exercise 1-10 8

Character Counting 9

Line Counting 9

Exercise 1-8 10

Word Counting 10

Exercise 1-12 12

Arrays 13

Exercise 1-13 13

Exercise 1-14 14

Functions 16

Exercise 1-16 16

Exercise 1-17 18

Exercise 1-18 20

Headers 21
Patterns 21
Control 21
I/O 21
Predicates 21
Library 22

Chapter One

Hello, world!

Covers Exercises 1-1 and 1-2.

Include the standard I/O functions, notably `printf`.

5a ⟨*hello.c* 5a⟩≡
  ⟨*Include the standard I/O functions.* 21b⟩
This definition is continued in chunk 5b.
Root chunk (not used in this document).

Define a `main` function that prints `Hello, world!`.

5b ⟨*hello.c* 5a⟩+≡
```
int main()
{
    printf("Hello, world!\n");
}
```
Defines:
  `main`, never used.

Fahrenheit-Celsius table

Covers Exercises 1-3, 1-4, and 1-5.

Include the standard I/O and string functions.

5c ⟨*fahrcels.c* 5c⟩≡
  ⟨*Include the standard I/O functions.* 21b⟩
  ⟨*Include the standard string functions.* 21c⟩
This definition is continued in chunks 5–7.
Root chunk (not used in this document).

Declare some useful constants.

5d ⟨*fahrcels.c* 5c⟩+≡
```
#define LOWER 0
#define UPPER 300
#define STEP  20
```

Defines:
  `LOWER`, used in chunks 6d and 7b.
  `STEP`, used in chunks 6d and 7b.
  `UPPER`, used in chunks 6d and 7b.

Exercise 1-3

6a ⟨*fahrcels.c* 5c⟩+≡

```
void hrule(size_t width)
{

}
```

Defines:
`hrule`, never used.

To print a two-column table header in https://orgmode.org format,

6b ⟨*fahrcels.c* 5c⟩+≡

```
void print_header(char lhs[], char rhs[])
{
    printf("| %s | %s |\n", lhs, rhs);
    putchar('|');
    for (size_t i = 0; i < strlen(lhs)+2; ++i)
        putchar('-');
    putchar('+');
    for (size_t i = 0; i < strlen(rhs)+2; ++i)
        putchar('-');
    puts("|");
}
```

Defines:
`print_header`, used in chunks 6d and 7b.

Exercise 1-4

To convert from Celsius to Fahrenheit, multiply by nine, then divide by five, then add 32.

6c ⟨*fahrcels.c* 5c⟩+≡

```
#define cels2fahr(cels) ((9.0/5.0)*cels+32.0)
```

Defines:
`cels2fahr`, used in chunk 6d.

6d ⟨*fahrcels.c* 5c⟩+≡

```
void celsfahr()
{
    print_header("Celsius", "Fahrenheit");
    for (int celsius = LOWER; celsius ≤ UPPER; celsius += STEP)
        printf("| %7d | %10.0f |\n", celsius, cels2fahr(celsius));
}
```

Defines:
`celsfahr`, used in chunk 7c.
Uses `cels2fahr` 6c, `LOWER` 5d, `print_header` 6b, `STEP` 5d, and `UPPER` 5d.

Exercise 1-5

To convert from Fahrenheit to Celsius, subtract 32, multiply by five, then divide by nine.

7a ⟨*fahrcels.c* 5c⟩+≡
```
#define fahr2cels(fahr) ((fahr-32.0)*(5.0/9.0))
```
Defines:
fahr2cels, never used.

7b ⟨*fahrcels.c* 5c⟩+≡
```
void fahrcels()
{
    print_header("Fahrenheit", "Celsius");
    for (int fahr = UPPER; fahr ≥ LOWER; fahr -= STEP)
        printf("| %10d | %7.1f |\n", fahr, (5.0/9.0) * (fahr-32.0));
}
```
Defines:
fahrcels, used in chunk 7c.
Uses LOWER 5d, print_header 6b, STEP 5d, and UPPER 5d.

The `main` *function*

7c ⟨*fahrcels.c* 5c⟩+≡
```
int main()
{
    fahrcels();
    puts("\n");
    celsfahr();

    return 0;
}
```
Defines:
main, never used.
Uses celsfahr 6d and fahrcels 7b.

Copy

Covers Exercises 1-6 and 1-7.

7d ⟨*copy.c* 7d⟩≡
⟨*Include the standard I/O functions.* 21b⟩

```
int main()
{
    int c;
    ⟨For each character c until EOF 21d⟩
        ⟨Print the character. 21e⟩

    return 0;
}
```
Root chunk (not used in this document).
Defines:
main, never used.

Exercise 1-9

Copy the program's input to its output, replacing each string of one or more blanks, by a single blank.

8a ⟨*catblanks.c* 8a⟩≡

```
⟨Include the standard I/O functions. 21b⟩
⟨Include the boolean type and values. 21a⟩


int main()
{
    int c;
    bool prev_blank = false;

    ⟨For each character c until EOF 21d⟩ {
        if (!(prev_blank && c == ' '))
            ⟨Print the character. 21e⟩
        prev_blank = (c == ' ');
    }


    return 0;
}
```

Root chunk (not used in this document).
Defines:
`main`, never used.

Exercise 1-10

Process each character `c`.

8c ⟨*unambiguous.c* 8b⟩+≡

```
    int c;

    ⟨For each character c until EOF 21d⟩ {
```

Replace each tab by \t.

8d ⟨*unambiguous.c* 8b⟩+≡

```
        if (⟨the character is a tab 22a⟩)
            fputs("\\t", stdout);
```

Replace each backspace by \b.

8e ⟨*unambiguous.c* 8b⟩+≡

```
        else if (⟨the character is a backspace 22b⟩)
            fputs("\\b", stdout);
```

Replace each backslash by \\.

8f ⟨*unambiguous.c* 8b⟩+≡

```
        else if (⟨the character is a backslash 22c⟩)
            fputs("\\\\", stdout);
```

8b ⟨*unambiguous.c* 8b⟩≡

```
⟨Include the standard I/O functions. 21b⟩


int main()
{
```

This definition is continued in chunks 8 and 9.
Root chunk (not used in this document).
Defines:
`main`, never used.

Otherwise print the character unchanged.

9a ⟨*unambiguous.c* 8b⟩+≡
```
        else
            ⟨Print the character. 21e⟩
```

Finally, close the **while** loop and exit.

9b ⟨*unambiguous.c* 8b⟩+≡
```
    }

    return 0;
}
```

Character Counting

9c ⟨*wc.c* 9c⟩≡
```
⟨Include the standard I/O functions. 21b⟩
⟨Include the boolean type and values. 21a⟩
```
This definition is continued in chunks 9–11.
Root chunk (not used in this document).

Until the end of the file, count characters by incrementing a **double** nc.

9d ⟨*wc.c* 9c⟩+≡
```
double char_count()
{
    double nc = 0;

    while (getchar() ≠ EOF)
        ++nc;

    return nc;
}
```
Defines:
char_count, never used.

Line Counting

Until the end of the file, count the newline characters.

9e ⟨*wc.c* 9c⟩+≡
```
int line_count()
{
    int c, nl = 0;
    ⟨For each character c until EOF 21d⟩
        if (⟨the character is a newline 21g⟩)
            ++nl;

    return nl;
}
```
Defines:
line_count, never used.

Exercise 1-8

10a ⟨*wc.c* 9c⟩+≡

```
bool is_whitespace(int c)
{
    return (⟨the character is whitespace 21f⟩);
}
```

10b ⟨*wc.c* 9c⟩+≡

```
double ws_count()
{
    double ns = 0;
    int c = 0;

    ⟨For each character c until EOF 21d⟩
        if (is_whitespace(c))
            ++ns;

    return ns;
}
```

Defines:
ws_count, never used.

Word Counting

10c ⟨*wc.c* 9c⟩+≡

```
#define IN  1
#define OUT 0
```

Defines:
IN, used in chunk 11.
OUT, used in chunk 11.

11 ⟨*wc.c* 9c⟩+≡

```
int main()
{
    int c, nl, nw, nc, state;

    state = OUT;
    nl = nw = nc = 0;
    ⟨For each character c until EOF 21d⟩ {
        ++nc;
        if (⟨the character is a newline 21g⟩)
            ++nl;
        if (⟨the character is whitespace 21f⟩)
            state = OUT;
        else if (state == OUT) {
          state = IN;
          ++nw;
        }
    }

    printf("%7d%8d%8d\n", nl, nw, nc);

    return 0;
}
```

Defines:
  main, never used.
Uses IN 10c 12 13 and OUT 10c 12 13.

Exercise 1-12

12 ⟨*words.c* 12⟩≡
⟨*Include the standard I/O functions.* 21b⟩

```
#define IN    1
#define OUT   0

int main()
{
    int c, state;

    state = OUT;
    ⟨For each character c until EOF 21d⟩ {
        if (⟨the character is whitespace 21f⟩) {
            if (state == IN)
                putchar('\n');
            state = OUT;
        } else {
            state = IN;
        }

        if (state == IN)
            putchar(c);
    }

    return 0;
}
```

Root chunk (not used in this document).
Defines:
IN, used in chunk 11.
main, never used.
OUT, used in chunk 11.

Arrays

Exercise 1-13

Vertical histogram

13 ⟨*wordlength.c* 13⟩≡
⟨*Include the standard I/O functions.* 21b⟩

```
#define IN    1
#define OUT   0

#define MAX_WORD_LENGTH 10
#define TERM_WIDTH 80


int main()
{
    int c, state, wl;
    int length[MAX_WORD_LENGTH+1];

    for (int i = 0; i ≤ MAX_WORD_LENGTH; ++i)
        length[i] = 0;

    state = OUT;
    wl = 0;
    ⟨For each character c until EOF 21d⟩ {
        if (⟨the character is whitespace 21f⟩) {
            if (state == IN) {
                state = OUT;
                ++length[wl ≤ MAX_WORD_LENGTH ? wl-1 : MAX_WORD_LENGTH];
            }
        } else {
            if (state == OUT) {
                state = IN;
                wl = 0;
            }
            ++wl;
        }
    }

    for (int j = 0; j ≤ MAX_WORD_LENGTH; ++j) {
        if (j == MAX_WORD_LENGTH)
            printf(">%d: ", MAX_WORD_LENGTH);
        else
            printf(" %2d: ", j+1);

        for (int k = 0; k < length[j]; ++k)
            putchar('#');
        putchar('\n');
    }
```

```
    return 0;
}
```

Root chunk (not used in this document).
Defines:
IN, used in chunk 11.
main, never used.
MAX_WORD_LENGTH, never used.
OUT, used in chunk 11.
TERM_WIDTH, never used.

Exercise 1-14

14 ⟨*charfreq.c* 14⟩≡
⟨*Include the standard I/O functions.* 21b⟩

```
#define MIN_ASCII 0
#define MAX_ASCII 0177
```

This definition is continued in chunks 15 and 16a.
Root chunk (not used in this document).
Defines:
MAX_ASCII, used in chunk 16a.
MIN_ASCII, never used.

15 ⟨*charfreq.c* 14⟩+≡

```
void prchar(int c)
{
    switch (c) {
        case ' ':
            printf("%11s", "<space>");
            break;
        case '\b':
            printf("%11s", "<backspace>");
            break;
        case '\n':
            printf("%11s", "<newline>");
            break;
        case '\t':
            printf("%11s", "<tab>");
            break;
        default:
            /* FIXME: why can't I return this? */
            /* return ((char[2]) { (char) c, '\0' }); */
            printf("%11c", c);
            break;
    }
}
```

Defines:
prchar, used in chunk 16a.

16a ⟨*charfreq.c* 14⟩+≡

```
int main()
{
    int c;
    int freq[MAX_ASCII+1] = {0};

    ⟨For each character c until EOF 21d⟩
        ++freq[c];

    for (int i = 0; i ≤ MAX_ASCII; ++i) {
        if (!freq[i]) continue;

        prchar(i);
        fputs(": ", stdout);
        for (int j = 0; j < freq[i]; ++j)
            putchar('#');
        putchar('\n');
    }


    return 0;
}
```

Defines:
  main, never used.
Uses MAX_ASCII 14 and prchar 15.

Functions

Exercise 1-16

16b ⟨*longestline.c* 16b⟩≡

```
/*!
  @file
  @brief Longest Line
  @author Eric Bailey
  @date 2019-04-13
*/
```

This definition is continued in chunks 16–18.
Root chunk (not used in this document).

16c ⟨*longestline.c* 16b⟩+≡

```
⟨Include the standard I/O functions. 21b⟩
#include "get_line.h"
```

Uses get_line 23a.

Define the maximum line length to read into memory.

17a ⟨*longestline.c* 16b⟩+≡
```
/// The maximum line length to read into memory.
#define MAXLINE 80
```

Defines:
MAXLINE, used in chunks 17 and 19.

17b ⟨*longestline.c* 16b⟩+≡
```
void copy(char to[], char from[]);
```

Uses copy 18a.

17c ⟨*longestline.c* 16b⟩+≡
```
int main()
{
    int len, max;
    char line[MAXLINE], longest[MAXLINE];

    max = 0;
    while ((len = get_line(line, MAXLINE)) > 0)
        if (len > max) {
            max = len;
            copy(longest, line);
        }

    if (max > 0) {
```

Defines:
main, never used.
Uses copy 18a, get_line 23a, and MAXLINE 17a 18d 20.

Print the length of the longest line, and as much of it as possible:

17d ⟨*longestline.c* 16b⟩+≡
```
        printf("The longest line had %d characters:\n%s", max, longest);
```

If the line was too long to print fully, print an ellipsis and a newline.

17e ⟨*longestline.c* 16b⟩+≡
```
        if (max ≥ MAXLINE && longest[MAXLINE-1] ≠ '\n')
            fputs("...\n", stdout);
```

Uses MAXLINE 17a 18d 20.

17f ⟨*longestline.c* 16b⟩+≡
```
    }

    return 0;
}
```

18a ⟨*longestline.c* 16b⟩+≡

```
/* copy: copy 'from' into 'to'; assume 'to' is big enough */
void copy(char to[], char from[])
{
    int i;
    i = 0;
    while ((to[i] = from[i]) != '\0')
        ++i;
}
```

Defines:
copy, used in chunk 17.

Exercise 1-17

18b ⟨*longlines.c* 18b⟩≡

```
/*!
  @file
  @brief Print long lines.
  @author Eric Bailey
  @date 2019-04-13
*/
```

This definition is continued in chunks 18 and 19.
Root chunk (not used in this document).

18c ⟨*longlines.c* 18b⟩+≡

```
⟨Include the standard I/O functions. 21b⟩
#include "get_line.h"
```

Uses get_line 23a.

Define the maximum line length to read into memory.

18d ⟨*longlines.c* 18b⟩+≡

```
/// The maximum line length to read into memory.
#define MAXLINE 57
```

Defines:
MAXLINE, used in chunks 17 and 19.

Define the minimum line length to be considered long.

18e ⟨*longlines.c* 18b⟩+≡

```
/// The minimum line length to be considered long.
#define MINLENGTH 54
```

Defines:
MINLENGTH, used in chunk 19.

19 ⟨*longlines.c* 18b⟩+≡

```
int main()
{
    int len;
    char line[MAXLINE];

    while ((len = get_line(line, MAXLINE)) > 0) {
       if (len > MINLENGTH)
           printf("%s", line);
       if (len ≥ MAXLINE && line[MAXLINE - 1] ≠ '\n')
            fputs("...\n", stdout);
    }

    return 0;
}
```

Defines:
  `main`, never used.
Uses `get_line` 23a, `MAXLINE` 17a 18d 20, and `MINLENGTH` 18e.

Exercise 1-18

20 ⟨*trimlines.c* 20⟩≡

```
⟨Include the boolean type and values. 21a⟩
⟨Include the standard I/O functions. 21b⟩
#include <string.h>
#include "get_line.h"


#define MAXLINE 120


size_t trim_line(char line[], size_t len)
{
    while (line[-len] == ' ' || line[len] == '\t' || line[len] == '\n') {
        if (len == 0) {
            return 0;
        } else {
            line[len] = '\0';
        }
    }

    line[++len] = '\n';

    return len;
}


int main()
{
    size_t len;
    char line[MAXLINE];

    while ((len = get_line(line, MAXLINE)) > 0)
        if (trim_line(line, len) > 0)
            fputs(line, stdout);

    return 0;
}
```

Root chunk (not used in this document).
Defines:
  `main`, never used.
  `MAXLINE`, used in chunks 17 and 19.
Uses `get_line` 23a.

Common

Headers

21a ⟨*Include the boolean type and values.* 21a⟩≡
```
#include <stdbool.h>
```
This code is used in chunks 8a, 9c, and 20.

21b ⟨*Include the standard I/O functions.* 21b⟩≡
```
#include <stdio.h>
```
This code is used in chunks 5, 7–9, 12–14, 16c, 18c, 20, and 23a.

21c ⟨*Include the standard string functions.* 21c⟩≡
```
#include <string.h>
```
This code is used in chunk 5c.

Patterns

Control

21d ⟨*For each character* `c` *until* `EOF` 21d⟩≡
```
while ((c = getchar()) ≠ EOF)
```
This code is used in chunks 7–13 and 16a.

I/O

21e ⟨*Print the character.* 21e⟩≡
```
putchar(c);
```
This code is used in chunks 7–9.

Predicates

For our purposes, whitespace is a space, tab, or newline.

21f ⟨*the character is whitespace* 21f⟩≡
```
c == ' ' || ⟨the character is a newline 21g⟩ || ⟨the character is a tab 22a⟩
```
This code is used in chunks 10–13.

21g ⟨*the character is a newline* 21g⟩≡
```
c == '\n'
```
This code is used in chunks 9e, 11, and 21f.

22a ⟨*the character is a tab* 22a⟩≡
```
c == '\t'
```
This code is used in chunks 8d and 21f.

22b ⟨*the character is a backspace* 22b⟩≡
```
c == '\b'
```
This code is used in chunk 8e.

22c ⟨*the character is a backslash* 22c⟩≡
```
c == '\\'
```
This code is used in chunk 8f.

Library

22d ⟨*get_line.h* 22d⟩≡
```
/**
 * @file get_line.h
 * @brief Read a line and return its length.
 * @author Eric Bailey
 * @date 2019-04-13
 */
```

This definition is continued in chunk 22e.
Root chunk (not used in this document).
Uses get_line 23a.

Declare a function get_line that, given a character array and maximum line length to copy to it, returns the length of the longest line.

22e ⟨*get_line.h* 22d⟩+≡
```
int get_line(char line[], int maxline);
```

Uses get_line 23a.

23a ⟨*get_line.c* 23a⟩≡

```
/**
 * @file get_line.c
 * @brief Read a line and return its length.
 * @author Eric Bailey
 * @date 2019-04-13
 */


⟨Include the standard I/O functions. 21b⟩
#include "get_line.h"


/**
 * Read a line into @p s, up to @p lim characters.
 *
 * @param s A character array.
 * @param lim The length of @p s.
 *
 * @return The full length of the line.
 */
int get_line(char s[], int lim)
{
    int c = EOF, i;

    for (i = 0; i < lim-1 && (c = getchar()) != EOF; ++i) {
        s[i] = c;
        if (c == '\n') {
            ++i;
            break;
        }
    }

    s[i] = '\0';
```

This definition is continued in chunks 23 and 24.
Root chunk (not used in this document).
Defines:
get_line, used in chunks 16–20 and 22.

If the last character read is a newline, return the number of characters in the line.

23b ⟨*get_line.c* 23a⟩+≡

```
    if (c == '\n')
        return i;
```

Otherwise, continue to count characters, until the end of the line or file.

24a ⟨*get_line.c* 23a⟩+≡

```
    while ((c = getchar()) ≠ EOF) {
        ++i;
        if (c == '\n')
            break;
    }
```

Return the length of the longest line.

24b ⟨*get_line.c* 23a⟩+≡

```
    return i;
}
```

Chunks

⟨*catblanks.c* 8a⟩ 8a
⟨*charfreq.c* 14⟩ 14, 15, 16a
⟨*copy.c* 7d⟩ 7d
⟨*fahrcels.c* 5c⟩ 5c, 5d, 6a, 6b, 6c, 6d, 7a, 7b, 7c
⟨*For each character* `c` *until* `EOF` 21d⟩ 7d, 8a, 8c, 9e, 10b, 11, 12, 13, 16a, 21d
⟨*get_line.c* 23a⟩ 23a, 23b, 24a, 24b
⟨*get_line.h* 22d⟩ 22d, 22e
⟨*hello.c* 5a⟩ 5a, 5b
⟨*Include the boolean type and values.* 21a⟩ 8a, 9c, 20, 21a
⟨*Include the standard I/O functions.* 21b⟩ 5a, 5c, 7d, 8a, 8b, 9c, 12, 13, 14, 16c, 18c, 20, 21b, 23a
⟨*Include the standard string functions.* 21c⟩ 5c, 21c
⟨*longestline.c* 16b⟩ 16b, 16c, 17a, 17b, 17c, 17d, 17e, 17f, 18a
⟨*longlines.c* 18b⟩ 18b, 18c, 18d, 18e, 19
⟨*Print the character.* 21e⟩ 7d, 8a, 9a, 21e
⟨*the character is a backslash* 22c⟩ 8f, 22c
⟨*the character is a backspace* 22b⟩ 8e, 22b
⟨*the character is a newline* 21g⟩ 9e, 11, 21f, 21g
⟨*the character is a tab* 22a⟩ 8d, 21f, 22a
⟨*the character is whitespace* 21f⟩ 10a, 11, 12, 13, 21f
⟨*trimlines.c* 20⟩ 20
⟨*unambiguous.c* 8b⟩ 8b, 8c, 8d, 8e, 8f, 9a, 9b
⟨*wc.c* 9c⟩ 9c, 9d, 9e, 10a, 10b, 10c, 11
⟨*wordlength.c* 13⟩ 13
⟨*words.c* 12⟩ 12

Index

cels2fahr: 6c, 6d
celsfahr: 6d, 7c
char_count: 9d
copy: 17b, 17c, 18a
fahr2cels: 7a
fahrcels: 7b, 7c
get_line: 16c, 17c, 18c, 19, 20, 22d, 22e, 23a
hrule: 6a
IN: 10c, 11, 12, 13
line_count: 9e
LOWER: 5d, 6d, 7b
main: 5b, 7c, 7d, 8a, 8b, 11, 12, 13, 16a, 17c, 19, 20
MAX_ASCII: 14, 16a
MAX_WORD_LENGTH: 13
MAXLINE: 17a, 17c, 17e, 18d, 19, 20
MIN_ASCII: 14
MINLENGTH: 18e, 19
OUT: 10c, 11, 12, 13
prchar: 15, 16a
print_header: 6b, 6d, 7b
STEP: 5d, 6d, 7b
TERM_WIDTH: 13
UPPER: 5d, 6d, 7b
ws_count: 10b