The C Programming Language: Chapter 1

Eric Bailey

March 4, 2018[1]

[1] Last updated October 24, 2018

Write an abstract

Contents

Hello, world!

Covers Exercises 1-1 and 1-2.

2a ⟨*hello.c* 2a⟩≡

```
⟨Include the standard I/O functions. 14c⟩


int main()
{
    printf("Hello, world!\n");
}
```

Uses `printf` 14c.
Root chunk (not used in this document).

Fahrenheit-Celsius table

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

2b ⟨*fahrcels.c* 2b⟩≡

```
⟨Include the standard I/O functions. 14c⟩
⟨Include the standard string functions. 14d⟩
```

This definition is continued in chunks 2 and 3.
Root chunk (not used in this document).

Declare some useful constants.

2c ⟨*fahrcels.c* 2b⟩+≡

```
#define LOWER 0
#define UPPER 300
#define STEP  20
```

Defines:
`LOWER`, used in chunk 3b.
`STEP`, used in chunk 3b.
`UPPER`, used in chunk 3b.

Exercise 1-3

2d ⟨*fahrcels.c* 2b⟩+≡

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

Defines:
`print_header`, used in chunk 3.
Uses `printf` 14c, `putchar` 14c, `puts` 14c, and `strlen` 14d.

Exercise 1-4

3a ⟨*fahrcels.c* 2b⟩+≡

```
void celsfahr()
{
    print_header("Celsius", "Fahrenheit");
    for (int celsius = 0; celsius ≤ 300; celsius += 20)
        printf("| %7d | %10.0f |\n", celsius, 32.0 + (9.0/5.0) * celsius);
}
```

Defines:
  `celsfahr`, used in chunk 3c.
Uses `printf` 14c and `print_header` 2d.

Exercise 1-5

3b ⟨*fahrcels.c* 2b⟩+≡

```
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 3c.
Uses `LOWER` 2c, `STEP` 2c, `UPPER` 2c, `printf` 14c, and `print_header` 2d.

The `main` function

3c ⟨*fahrcels.c* 2b⟩+≡

```
int main()
{
    fahrcels();
    puts("\n");
    celsfahr();

    return 0;
}
```

Uses `celsfahr` 3a, `fahrcels` 3b, and `puts` 14c.

Copy

Covers Exercises 1-6 and 1-7.

4c ⟨*copy.c* 4c⟩≡
```
⟨Include the standard I/O functions. 14c⟩


int main()
{
    int c;
    ⟨For each character c until EOF 4a⟩
        ⟨Print the character. 4b⟩

    return 0;
}
```
Root chunk (not used in this document).

4a ⟨*For each character* c *until* EOF 4a⟩≡
```
while ((c = getchar()) ≠ EOF)
```
This code is used in chunks 4–9 and 11a.

4b ⟨*Print the character.* 4b⟩≡
```
putchar(c);
```
Uses putchar 14c.
This code is used in chunks 4c, 6a, and 7a.

Character Counting

4d ⟨*wc.c* 4d⟩≡
```
⟨Include the standard I/O functions. 14c⟩
⟨Include the boolean type and values. 14b⟩
```
This definition is continued in chunks 4, 5, and 7.
Root chunk (not used in this document).

4e ⟨*wc.c* 4d⟩+≡
```
double char_count()
{
    double nc;

    for (nc = 0; getchar() ≠ EOF; ++nc)
        ;

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

Line Counting

5a ⟨*the character is a newline* 5a⟩≡
```
c == '\n'
```
This code is used in chunks 5 and 7d.

5b ⟨*wc.c* 4d⟩+≡
```
int line_count()
{
    int c, nl;

    nl = 0;
    ⟨For each character c until EOF 4a⟩
        if (⟨the character is a newline 5a⟩)
            ++nl;

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

Exercise 1-8

5c ⟨*the character is a tab* 5c⟩≡
```
c == '\t'
```
This code is used in chunks 5d and 6d.

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

5d ⟨*the character is whitespace* 5d⟩≡
```
c == ' ' || ⟨the character is a newline 5a⟩ || ⟨the character is a tab 5c⟩
```
This code is used in chunks 5e and 7–9.

5e ⟨*wc.c* 4d⟩+≡
```
bool is_whitespace(int c)
{
    return (⟨the character is whitespace 5d⟩);
}
```
Defines:
`is_whitespace`, used in chunk 5f.
Uses `bool` 14b.

5f ⟨*wc.c* 4d⟩+≡
```
double ws_count()
{
    double ns = 0;
    int c = 0;

    ⟨For each character c until EOF 4a⟩
        if (is_whitespace(c))
            ++ns;

    return ns;
}
```
Defines:
`ws_count`, never used.
Uses `is_whitespace` 5e.

Exercise 1-9

```
6a  ⟨catblanks.c 6a⟩≡
      ⟨Include the standard I/O functions. 14c⟩
      ⟨Include the boolean type and values. 14b⟩


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

          ⟨For each character c until EOF 4a⟩ {
              if (!(prev_blank && c == ' '))
                  ⟨Print the character. 4b⟩
              prev_blank = (c == ' ');
          }


          return 0;
      }
```

Uses `bool` 14b.
Root chunk (not used in this document).

Exercise 1-10

```
6b  ⟨unambiguous.c 6b⟩≡
      ⟨Include the standard I/O functions. 14c⟩


      int main()
      {
```

This definition is continued in chunks 6 and 7.
Root chunk (not used in this document).

Process each character `c`.

```
6c  ⟨unambiguous.c 6b⟩+≡
      int c;

          ⟨For each character c until EOF 4a⟩ {
```

Replace each tab by `\t`.

```
6d  ⟨unambiguous.c 6b⟩+≡
              if (⟨the character is a tab 5c⟩)
                  fputs("\\t", stdout);
```

Uses `fputs` 14c and `stdout` 14c.

```
6e  ⟨the character is a backspace 6e⟩≡
      c == '\b'
```

This code is used in chunk 6f.

Replace each backspace by `\b`.

```
6f  ⟨unambiguous.c 6b⟩+≡
              else if (⟨the character is a backspace 6e⟩)
                 fputs("\\b", stdout);
```

Uses `fputs` 14c and `stdout` 14c.

```
6g  ⟨the character is a backslash 6g⟩≡
      c == '\\'
```

This code is used in chunk 6h.

Replace each backslash by `\\`.

```
6h  ⟨unambiguous.c 6b⟩+≡
              else if (⟨the character is a backslash 6g⟩)
                  fputs("\\\\", stdout);
```

Uses `fputs` 14c and `stdout` 14c.

Otherwise print the character unchanged.

7a ⟨*unambiguous.c* 6b⟩+≡

```
        else
            ⟨Print the character. 4b⟩
```

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

7b ⟨*unambiguous.c* 6b⟩+≡

```
    }

    return 0;
}
```

Word Counting

7c ⟨*wc.c* 4d⟩+≡

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

Defines:
IN, used in chunks 7–9.
OUT, used in chunks 7–9.

7d ⟨*wc.c* 4d⟩+≡

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

    state = OUT;
    nl = nw = nc = 0;
    ⟨For each character c until EOF 4a⟩ {
        ++nc;
        if (⟨the character is a newline 5a⟩)
            ++nl;
        if (⟨the character is whitespace 5d⟩)
            state = OUT;
        else if (state == OUT) {
          state = IN;
          ++nw;
        }
    }

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

    return 0;
}
```

Uses IN 7c, OUT 7c, and printf 14c.

Exercise 1-12

8 ⟨*words.c* 8⟩≡

```
⟨Include the standard I/O functions. 14c⟩


#define IN    1
#define OUT   0


int main()
{
    int c, state;

    state = OUT;
    ⟨For each character c until EOF 4a⟩ {
        if (⟨the character is whitespace 5d⟩) {
            if (state == IN)
                putchar('\n');
            state = OUT;
        } else {
            state = IN;
        }

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


    return 0;
}
```

Uses `IN` 7c, `OUT` 7c, and `putchar` 14c.
Root chunk (not used in this document).

Arrays

Exercise 1-13

Vertical histogram

9 ⟨*wordlength.c* 9⟩≡

```
  ⟨Include the standard I/O functions. 14c⟩


  #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 4a⟩ {
          if (⟨the character is whitespace 5d⟩) {
              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;
}
```

Uses IN 7c, OUT 7c, printf 14c, and putchar 14c.
Root chunk (not used in this document).

Exercise 1-14

10a ⟨*charfreq.c* 10a⟩≡

```
  ⟨Include the standard I/O functions. 14c⟩


  #define MIN_ASCII 0
  #define MAX_ASCII 0177
```

This definition is continued in chunks 10b and 11a.
Root chunk (not used in this document).

10b ⟨*charfreq.c* 10a⟩+≡

```
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 11a.
Uses printf 14c.

11a ⟨*charfreq.c* 10a⟩+≡

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

    ⟨For each character c until EOF 4a⟩
        ++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;
}
```

Uses `fputs` 14c, `prchar` 10b, `putchar` 14c, and `stdout` 14c.

Functions

Exercise 1-16

11b ⟨*longestline.c* 11b⟩≡

```
⟨Include the standard I/O functions. 14c⟩


#define MAXLINE 3
```

Defines:
`MAXLINE`, used in chunk 12.
This definition is continued in chunks 11–14.
Root chunk (not used in this document).

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

11c ⟨*longestline.c* 11b⟩+≡

```
int getline(char line[], int maxline);
```

Uses `getline` 13a.

12a ⟨*longestline.c* 11b⟩+≡

```
void copy(char to[], char from[]);


int main()
{
    int len, max;
    char line[MAXLINE], longest[MAXLINE];

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

    if (max > 0) {
```

Uses MAXLINE 11b, copy 14a, and getline 13a.

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

12b ⟨*longestline.c* 11b⟩+≡

```
        printf("The longest line had %d characters:\n%s", max, longest);
```

Uses printf 14c.

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

12c ⟨*longestline.c* 11b⟩+≡

```
        if (max ≥ MAXLINE && longest[MAXLINE-1] ≠ '\n')
            fputs("...\n", stdout);
```

Uses MAXLINE 11b, fputs 14c, and stdout 14c.

12d ⟨*longestline.c* 11b⟩+≡

```
    }


    return 0;
}
```

13a ⟨*longestline.c* 11b⟩+≡
```
/* getline: read a line into s, return length */
int getline(char s[], int lim)
{
    int c, i;

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

    if (c == '\n') {
        s[i] = c;
        ++i;
    }

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

Defines:
`getline`, used in chunks 13a, 11c, and 12a.

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

13b ⟨*longestline.c* 11b⟩+≡
```
    if (c == '\n')
        return i;
```

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

13c ⟨*longestline.c* 11b⟩+≡
```

    while ((c = getchar()) != '\n' && c != EOF)
        ++i;
```

If we ended on a newline character, increment the count.

13d ⟨*longestline.c* 11b⟩+≡
```
    if (c == '\n')
        ++i;
```

Return the length of the longest line.

13e ⟨*longestline.c* 11b⟩+≡
```
    return i;
}
```

14a ⟨*longestline.c* 11b⟩+≡
```
/* 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 12a.

Character Arrays

Exercise 1-17

Exercise 1-18

Exercise 1-19

External Variables and Scope

Exercise 1-20

Exercise 1-21

Exercise 1-22

Exercise 1-23

Exercise 1-24

Common Headers

14b ⟨*Include the boolean type and values.* 14b⟩≡
```
#include <stdbool.h>
```
Defines:
  `bool`, used in chunks 5e and 6a.
This code is used in chunks 4d and 6a.

14c ⟨*Include the standard I/O functions.* 14c⟩≡
```
#include <stdio.h>
```
Defines:
  `fputs`, used in chunks 6, 11a, and 12c.
  `printf`, used in chunks 2, 3, 7d, 9, 10b, and 12b.
  `putchar`, used in chunks 2d, 4b, 8, 9, and 11a.
  `puts`, used in chunks 2d and 3c.
  `stdout`, used in chunks 6, 11a, and 12c.
This code is used in chunks 2, 4, 6, and 8–11.

14d ⟨*Include the standard string functions.* 14d⟩≡
```
#include <string.h>
```
Defines:
  `strlen`, used in chunk 2d.
This code is used in chunk 2b.

Chunks

Index