The C Programming Language: Chapter 1

Eric Bailey

March 4, 2018 [1]

[1] Last updated October 16, 2018

Write an abstract

Contents

Hello, world!

Covers Exercises 1-1 and 1-2.

2 ⟨*hello.c* 2⟩≡
⟨*Include the standard I/O functions.* 12c⟩

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

Uses `printf` 12c.
Root chunk (not used in this document).

Fahrenheit-Celsius table

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

3a ⟨*fahrcels.c* 3a⟩≡

```
⟨Include the standard I/O functions. 12c⟩
⟨Include the standard string functions. 13⟩
```

This definition is continued in chunks 3 and 4.
Root chunk (not used in this document).

Declare some useful constants.

3b ⟨*fahrcels.c* 3a⟩+≡

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

Defines:
LOWER, used in chunk 4a.
STEP, used in chunk 4a.
UPPER, used in chunk 4a.

Exercise 1-3

3c ⟨*fahrcels.c* 3a⟩+≡

```
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 chunks 3d and 4a.
Uses printf 12c, putchar 12c, puts 12c, and strlen 13.

Exercise 1-4

3d ⟨*fahrcels.c* 3a⟩+≡

```
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 4b.
Uses printf 12c and print_header 3c.

Exercise 1-5

4a ⟨*fahrcels.c* 3a⟩+≡

```
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 4b.
Uses `LOWER` 3b, `STEP` 3b, `UPPER` 3b, `printf` 12c, and `print_header` 3c.

The `main` *function*

4b ⟨*fahrcels.c* 3a⟩+≡

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

    return 0;
}
```

Uses `celsfahr` 3d, `fahrcels` 4a, and `puts` 12c.

Copy

Covers Exercises 1-6 and 1-7.

4e ⟨*copy.c* 4e⟩≡

⟨*Include the standard I/O functions.* 12c⟩

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

    return 0;
}
```

Root chunk (not used in this document).

4c ⟨*For each character* `c` *until* `EOF` 4c⟩≡

```
while ((c = getchar()) ≠ EOF)
```

This code is used in chunks 4–6, 8–10, and 12a.

4d ⟨*Print the character.* 4d⟩≡

```
putchar(c);
```

Uses `putchar` 12c.
This code is used in chunks 4e, 6b, and 7f.

Character Counting

4f ⟨*wc.c* 4f⟩≡

⟨*Include the standard I/O functions.* 12c⟩
⟨*Include the boolean type and values.* 12b⟩

This definition is continued in chunks 5–8.
Root chunk (not used in this document).

5a ⟨*wc.c* 4f⟩+≡
```
double char_count()
{
    double nc;

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

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

Line Counting

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

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

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

5b ⟨*the character is a newline* 5b⟩≡
```
c == '\n'
```
This code is used in chunks 5 and 8.

Exercise 1-8

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

5e ⟨*the character is whitespace* 5e⟩≡
```
c == ' ' || ⟨the character is a newline 5b⟩ || ⟨the character is a tab 5d⟩
```
This code is used in chunks 5f and 8–10.

5f ⟨*wc.c* 4f⟩+≡
```
bool is_whitespace(int c)
{
    return (⟨the character is whitespace 5e⟩);
}
```
Defines:
`is_whitespace`, used in chunk 6a.
Uses `bool` 12b.

5d ⟨*the character is a tab* 5d⟩≡
```
c == '\t'
```
This code is used in chunks 5e and 7a.

6a ⟨*wc.c* 4f⟩+≡

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

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

    return ns;
}
```

Defines:
ws_count, never used.
Uses is_whitespace 5f.

Exercise 1-9

6b ⟨*catblanks.c* 6b⟩≡

```
⟨Include the standard I/O functions. 12c⟩
⟨Include the boolean type and values. 12b⟩


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

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


    return 0;
}
```

Uses bool 12b.
Root chunk (not used in this document).

Exercise 1-10

Process each character c.

6d ⟨*unambiguous.c* 6c⟩+≡

```
int c;

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

6c ⟨*unambiguous.c* 6c⟩≡

```
⟨Include the standard I/O functions. 12c⟩


int main()
{
```

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

Replace each tab by \t.

7a ⟨*unambiguous.c* 6c⟩+≡
```
if (⟨the character is a tab 5d⟩)
    fputs("\\t", stdout);
```
Uses fputs 12c and stdout 12c.

Replace each backspace by \b.

7c ⟨*unambiguous.c* 6c⟩+≡
```
else if (⟨the character is a backspace 7b⟩)
   fputs("\\b", stdout);
```
Uses fputs 12c and stdout 12c.

Replace each backslash by \\.

7e ⟨*unambiguous.c* 6c⟩+≡
```
else if (⟨the character is a backslash 7d⟩)
    fputs("\\\\", stdout);
```
Uses fputs 12c and stdout 12c.

Otherwise print the character unchanged.

7f ⟨*unambiguous.c* 6c⟩+≡
```
else
    ⟨Print the character. 4d⟩
```

Word Counting

7h ⟨*wc.c* 4f⟩+≡
```
#define IN  1
#define OUT 0
```
Defines:
IN, used in chunks 8–10.
OUT, used in chunks 8–10.

7b ⟨*the character is a backspace* 7b⟩≡
```
c == '\b'
```
This code is used in chunk 7c.

7d ⟨*the character is a backslash* 7d⟩≡
```
c == '\\'
```
This code is used in chunk 7e.

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

7g ⟨*unambiguous.c* 6c⟩+≡
```
    }

    return 0;
}
```

8 ⟨*wc.c* 4f⟩+≡

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

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

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

    return 0;
}
```

Uses IN 7h, OUT 7h, and printf 12c.

Exercise 1-12

9 ⟨*words.c* 9⟩≡
⟨*Include the standard I/O functions.* 12c⟩

```
#define IN    1
#define OUT   0


int main()
{
    int c, state;

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

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


    return 0;
}
```

Uses IN 7h, OUT 7h, and putchar 12c.
Root chunk (not used in this document).

Exercise 1-13

Vertical histogram

10 ⟨*wordlength.c* 10⟩≡
⟨*Include the standard I/O functions.* 12c⟩

```
#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 4c⟩ {
        if (⟨the character is whitespace 5e⟩) {
            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 7h, OUT 7h, printf 12c, and putchar 12c.
Root chunk (not used in this document).

Exercise 1-14

11a ⟨*charfreq.c* 11a⟩≡

```
⟨Include the standard I/O functions. 12c⟩


#define MIN_ASCII 0
#define MAX_ASCII 0177
```

This definition is continued in chunks 11b and 12a.
Root chunk (not used in this document).

11b ⟨*charfreq.c* 11a⟩+≡

```
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 12a.
Uses printf 12c.

12a ⟨*charfreq.c* 11a⟩+≡

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

    ⟨For each character c until EOF 4c⟩
        ++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 12c, prchar 11b, putchar 12c, and stdout 12c.

Common Headers

12b ⟨*Include the boolean type and values.* 12b⟩≡

```
#include <stdbool.h>
```

Defines:
bool, used in chunks 5f and 6b.
This code is used in chunks 4f and 6b.

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

```
#include <stdio.h>
```

Defines:
fputs, used in chunks 7 and 12a.
printf, used in chunks 2–4, 8, 10, and 11b.
putchar, used in chunks 3c, 4d, 9, 10, and 12a.
puts, used in chunks 3c and 4b.
stdout, used in chunks 7 and 12a.
This code is used in chunks 2–4, 6, and 9–11.

13 ⟨*Include the standard string functions.* 13⟩≡

```
#include <string.h>
```

Defines:
`strlen`, used in chunk 3c.
This code is used in chunk 3a.

Chunks

Index

ws_count: 6a