ERIC BAILEY

THE C PROGRAMMING LANGUAGE

Contents

Character Counting

4a ⟨*wc.c* 4a⟩≡
This definition is continued in chunks 4 and 5.
Root chunk (not used in this document).

```
  ⟨Include the standard I/O functions. 19b⟩
  ⟨Include the boolean type and values. 19a⟩
```

4b ⟨*wc.c* 4a⟩+≡

```
double char_count()
{
    double nc;

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

    return nc;
}
```

Defines:
`char_count`, never used.

Line Counting

4c ⟨*wc.c* 4a⟩+≡

```
int line_count()
{
    int c, nl;

    nl = 0;
    ⟨For each character c until EOF 19d⟩
        if (⟨the character is a newline 20c⟩)
            ++nl;

    return nl;
}
```

Defines:
`line_count`, never used.

Exercise 1-8

4d ⟨*wc.c* 4a⟩+≡

```
bool is_whitespace(int c)
{
    return (⟨the character is whitespace 20b⟩);
}
```

Defines:
`is_whitespace`, used in chunk 5a.
Uses `bool` 19a.

5a ⟨*wc.c* 4a⟩+≡

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

    ⟨For each character c until EOF 19d⟩
        if (is_whitespace(c))
            ++ns;

    return ns;
}
```

Defines:
`ws_count`, never used.
Uses `is_whitespace` 4d.

Word Counting

5b ⟨*wc.c* 4a⟩+≡

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

Defines:
`IN`, used in chunks 5c, 10, and 14b.
`OUT`, used in chunks 5c, 10, and 14b.

5c ⟨*wc.c* 4a⟩+≡

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

    state = OUT;
    nl = nw = nc = 0;
    ⟨For each character c until EOF 19d⟩ {
        ++nc;
        if (⟨the character is a newline 20c⟩)
            ++nl;
        if (⟨the character is whitespace 20b⟩)
            state = OUT;
        else if (state == OUT) {
          state = IN;
          ++nw;
        }
    }

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

    return 0;
}
```

Uses `IN` 5b, `OUT` 5b, and `printf` 19b.

Fahrenheit-Celsius table

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

6a ⟨*fahrcels.c* 6a⟩≡
This definition is continued in chunks 6 and 7.
Root chunk (not used in this document).

```
⟨Include the standard I/O functions. 19b⟩
⟨Include the standard string functions. 19c⟩
```

Declare some useful constants.

6b ⟨*fahrcels.c* 6a⟩+≡

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

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

Exercise 1-3

6c ⟨*fahrcels.c* 6a⟩+≡

```
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 6d and 7a.
Uses `printf` 19b, `putchar` 19b, `puts` 19b, and `strlen` 19c.

Exercise 1-4

6d ⟨*fahrcels.c* 6a⟩+≡

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

Defines:
`celsfahr`, used in chunk 7b.
Uses `LOWER` 6b, `print_header` 6c, `printf` 19b, `STEP` 6b, and `UPPER` 6b.

Exercise 1-5

7a ⟨*fahrcels.c* 6a⟩+≡
```
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 7b.
Uses `LOWER` 6b, `print_header` 6c, `printf` 19b, `STEP` 6b, and `UPPER` 6b.

The `main` function

7b ⟨*fahrcels.c* 6a⟩+≡
```
int main()
{
    fahrcels();
    puts("\n");
    celsfahr();

    return 0;
}
```
Uses `celsfahr` 6d, `fahrcels` 7a, and `puts` 19b.

Chapter One

Hello, world!

Covers Exercises 1-1 and 1-2.

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

9a ⟨*hello.c* 9a⟩≡

This definition is continued in chunk 9b.
Root chunk (not used in this document).

⟨*Include the standard I/O functions.* 19b⟩

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

9b ⟨*hello.c* 9a⟩+≡

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

Uses `printf` 19b.

Arrays

Exercise 1-13

Vertical histogram

10 ⟨*wordlength.c* 10⟩≡
Root chunk (not used in this document).

```
⟨Include the standard I/O functions. 19b⟩


#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 19d⟩ {
        if (⟨the character is whitespace 20b⟩) {
            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 5b, OUT 5b, printf 19b, and putchar 19b.

Exercise 1-9

11a ⟨*catblanks.c* 11a⟩≡

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

```
⟨Include the standard I/O functions. 19b⟩
⟨Include the boolean type and values. 19a⟩
```

11b ⟨*catblanks.c* 11a⟩+≡

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

    ⟨For each character c until EOF 19d⟩ {
        if (!(prev_blank && c == ' '))
            ⟨Print the character. 20a⟩
        prev_blank = (c == ' ');
    }


    return 0;
}
```
Uses bool 19a.

Exercise 1-14

11c ⟨*charfreq.c* 11c⟩≡

This definition is continued in chunk 12.
Root chunk (not used in this document).

```
⟨Include the standard I/O functions. 19b⟩


#define MIN_ASCII 0
#define MAX_ASCII 0177
```

12a ⟨*charfreq.c* 11c⟩+≡

```
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 12b.
Uses `printf` 19b.

12b ⟨*charfreq.c* 11c⟩+≡

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

    ⟨For each character c until EOF 19d⟩
        ++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` 19b, `prchar` 12a, `putchar` 19b, and `stdout` 19b.

Exercise 1-10

Process each character `c`.

13b ⟨*unambiguous.c* 13a⟩+≡
```
  int c;

        ⟨For each character c until EOF 19d⟩ {
```

Replace each tab by \t.

13c ⟨*unambiguous.c* 13a⟩+≡
```
            if (⟨the character is a tab 20d⟩)
                fputs("\\t", stdout);
```
Uses `fputs` 19b and `stdout` 19b.

Replace each backspace by \b.

13d ⟨*unambiguous.c* 13a⟩+≡
```
            else if (⟨the character is a backspace 20e⟩)
               fputs("\\b", stdout);
```
Uses `fputs` 19b and `stdout` 19b.

Replace each backslash by \\.

13e ⟨*unambiguous.c* 13a⟩+≡
```
            else if (⟨the character is a backslash 20f⟩)
                fputs("\\\\", stdout);
```
Uses `fputs` 19b and `stdout` 19b.

Otherwise print the character unchanged.

13f ⟨*unambiguous.c* 13a⟩+≡
```
            else
                ⟨Print the character. 20a⟩
```

13a ⟨*unambiguous.c* 13a⟩≡

This definition is continued in chunks 13 and 14a.
Root chunk (not used in this document).
```
  ⟨Include the standard I/O functions. 19b⟩


  int main()
  {
```

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

14a ⟨*unambiguous.c* 13a⟩+≡
```
        }

        return 0;
    }
```

Exercise 1-12

14b ⟨*words.c* 14b⟩≡
Root chunk (not used in this document).

```
⟨Include the standard I/O functions. 19b⟩


#define IN    1
#define OUT   0


int main()
{
    int c, state;

    state = OUT;
    ⟨For each character c until EOF 19d⟩ {
        if (⟨the character is whitespace 20b⟩) {
            if (state == IN)
                putchar('\n');
            state = OUT;
        } else {
            state = IN;
        }

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


    return 0;
}
```
Uses `IN` 5b, `OUT` 5b, and `putchar` 19b.

Copy

Covers Exercises 1-6 and 1-7.

14c ⟨*copy.c* 14c⟩≡
This definition is continued in chunk 15a.
Root chunk (not used in this document).

```
⟨Include the standard I/O functions. 19b⟩
```

15a ⟨*copy.c* 14c⟩+≡
```
int main()
{
    int c;
    ⟨For each character c until EOF 19d⟩
        ⟨Print the character. 20a⟩

    return 0;
}
```

Functions

Exercise 1-16

15b ⟨*longestline.c* 15b⟩≡
This definition is continued in chunks 15–17.
Root chunk (not used in this document).

```
⟨Include the standard I/O functions. 19b⟩


#define MAXLINE 3
```

Defines:
`MAXLINE`, used in chunks 15d and 16b.

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

15c ⟨*longestline.c* 15b⟩+≡
```
int getline(char line[], int maxline);
```
Uses `getline` 16d.

15d ⟨*longestline.c* 15b⟩+≡
```
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 `copy` 17d, `getline` 16d, and `MAXLINE` 15b.

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

16a ⟨*longestline.c* 15b⟩+≡
```
        printf("The longest line had %d characters:\n%s", max, longest);
```
Uses `printf` 19b.

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

16b ⟨*longestline.c* 15b⟩+≡
```
        if (max ≥ MAXLINE && longest[MAXLINE-1] ≠ '\n')
            fputs("...\n", stdout);
```
Uses `fputs` 19b, `MAXLINE` 15b, and `stdout` 19b.

16c ⟨*longestline.c* 15b⟩+≡
```
    }


    return 0;
}
```

16d ⟨*longestline.c* 15b⟩+≡
```
/* 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 16d and 15.

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

16e ⟨*longestline.c* 15b⟩+≡
```
    if (c == '\n')
        return i;
```

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

17a ⟨*longestline.c* 15b⟩+≡

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

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

17b ⟨*longestline.c* 15b⟩+≡

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

Return the length of the longest line.

17c ⟨*longestline.c* 15b⟩+≡

```
    return i;
}
```

17d ⟨*longestline.c* 15b⟩+≡

```
/* 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 15d.

Common

Headers

19a ⟨*Include the boolean type and values.* 19a⟩≡
This code is used in chunks 4a and 11a.

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

Defines:
`bool`, used in chunks 4d and 11b.

19b ⟨*Include the standard I/O functions.* 19b⟩≡
This code is used in chunks 4a, 6a, 9–11, and 13–15.

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

Defines:
`fputs`, used in chunks 12, 13, and 16b.
`printf`, used in chunks 5–7, 19b, 9b, 10, 12a, and 16a.
`putchar`, used in chunks 6c, 10, 12b, 14b, and 20a.
`puts`, used in chunks 6c and 7b.
`stdout`, used in chunks 12, 13, and 16b.

19c ⟨*Include the standard string functions.* 19c⟩≡
This code is used in chunk 6a.

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

Defines:
`strlen`, used in chunk 6c.

Patterns

Control

19d ⟨*For each character* `c` *until* `EOF` 19d⟩≡
This code is used in chunks 4, 5, and 10–15.

```
while ((c = getchar()) ≠ EOF)
```

I/O

20a ⟨*Print the character.* 20a⟩≡
This code is used in chunks 11b, 13f, and 15a.

```
putchar(c);
```
Uses `putchar` 19b.

Predicates

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

20b ⟨*the character is whitespace* 20b⟩≡
This code is used in chunks 4d, 5c, 10, and 14b.

```
c == ' ' || ⟨the character is a newline 20c⟩ || ⟨the character is a tab 20d⟩
```

20c ⟨*the character is a newline* 20c⟩≡
This code is used in chunks 4c, 5c, and 20b.

```
c == '\n'
```

20d ⟨*the character is a tab* 20d⟩≡
This code is used in chunks 13c and 20b.

```
c == '\t'
```

20e ⟨*the character is a backspace* 20e⟩≡
This code is used in chunk 13d.

```
c == '\b'
```

20f ⟨*the character is a backslash* 20f⟩≡
This code is used in chunk 13e.

```
c == '\\'
```

Chunks

Index

To-Do