

The C Programming Language: Chapter 1

Eric Bailey

March 4, 2018¹

¹ Last updated October 15, 2018

Write an abstract

Contents

<i>Hello, world!</i>	1
<i>Fahrenheit-Celsius table</i>	2
<i>Exercise 1-3</i>	2
<i>Exercise 1-4</i>	2
<i>Exercise 1-5</i>	3
<i>The main function</i>	3
<i>Copy</i>	3
<i>Character Counting</i>	3
<i>Line Counting</i>	4
<i>Exercise 1-8</i>	4
<i>Exercise 1-9</i>	5
<i>Exercise 1-10</i>	5
<i>Word Counting</i>	6
<i>Exercise 1-12</i>	8
<i>Common Headers</i>	8
<i>Chunks</i>	10
<i>Index</i>	10

Hello, world!

Covers Exercises 1-1 and 1-2.

1 `<hello.c 1>≡`
`<Include the standard I/O functions. 8c>`

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

Uses `printf 8c`.

Root chunk (not used in this document).

Fahrenheit-Celsius table

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

2a `<fahrrels.c 2a>≡`
`<Include the standard I/O functions. 8c>`
`<Include the standard string functions. 9>`

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

Declare some useful constants.

2b `<fahrrels.c 2a>+≡`
`#define LOWER 0`
`#define UPPER 300`
`#define STEP 20`

Defines:
 LOWER, used in chunk 3a.
 STEP, used in chunk 3a.
 UPPER, used in chunk 3a.

Exercise 1-3

2c `<fahrrels.c 2a>+≡`
`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 2d and 3a.
 Uses printf 8c, putchar 8c, puts 8c, and strlen 9.

Exercise 1-4

2d `<fahrrels.c 2a>+≡`
`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 3b.
 Uses printf 8c and print_header 2c.

Exercise 1-5

3a `<fahrcls.c 2a>+≡`

```
void fahrcls()
{
    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:

`fahrcls`, used in chunk 3b.

Uses `LOWER 2b`, `STEP 2b`, `UPPER 2b`, `printf 8c`, and `print_header 2c`.

The main function

3b `<fahrcls.c 2a>+≡`

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

    return 0;
}
```

Uses `celsfahr 2d`, `fahrcls 3a`, and `puts 8c`.

Copy

Covers Exercises 1-6 and 1-7.

3e `<copy.c 3e>≡`
<Include the standard I/O functions. 8c>

```
int main()
{
    int c;
    <For each character c until EOF 3c>
    <Print the character. 3d>

    return 0;
}
```

Root chunk (not used in this document).

Character Counting

3f `<wc.c 3f>≡`
<Include the standard I/O functions. 8c>
<Include the boolean type and values. 8b>

This definition is continued in chunks 4-7.

Root chunk (not used in this document).

3c *<For each character c until EOF 3c>≡*
`while ((c = getchar()) ≠ EOF)`

This code is used in chunks 3-5, 7, and 8a.

3d *<Print the character. 3d>≡*
`putchar(c);`

Uses `putchar 8c`.

This code is used in chunks 3e, 5b, and 6f.

4a $\langle wc.c\ 3f \rangle + \equiv$

```
double char_count()
{
    double nc;

    for (nc = 0; getchar()  $\neq$  EOF; ++nc)
        ;

    return nc;
}
```

Defines:
char_count, never used.

Line Counting

4c $\langle wc.c\ 3f \rangle + \equiv$

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

    nl = 0;
     $\langle$ For each character c until EOF 3c $\rangle$ 
        if ( $\langle$ the character is a newline 4b $\rangle$ )
            ++nl;

    return nl;
}
```

Defines:
line_count, never used.

4b \langle the character is a newline 4b $\rangle \equiv$
c = '\n'

This code is used in chunks 4 and 7.

Exercise 1-8

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

4e \langle the character is whitespace 4e $\rangle \equiv$
c = ' ' || \langle the character is a newline 4b \rangle || \langle the character is a tab 4d \rangle
This code is used in chunks 4f, 7, and 8a.

4d \langle the character is a tab 4d $\rangle \equiv$
c = '\t'

This code is used in chunks 4e and 6a.

4f $\langle wc.c\ 3f \rangle + \equiv$

```
bool is_whitespace(int c)
{
    return ( $\langle$ the character is whitespace 4e $\rangle$ );
}
```

Defines:
is_whitespace, used in chunk 5a.
Uses bool 8b.

5a $\langle wc.c\ 3f \rangle + \equiv$

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

     $\langle$ For each character c until EOF 3c $\rangle$ 
        if (is_whitespace(c))
            ++ns;

    return ns;
}
```

Defines:

`ws_count`, never used.

Uses `is_whitespace` 4f.

Exercise 1-9

5b $\langle catblanks.c\ 5b \rangle \equiv$

\langle Include the standard I/O functions. 8c \rangle

\langle Include the boolean type and values. 8b \rangle

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

     $\langle$ For each character c until EOF 3c $\rangle$  {
        if (!(prev_blank && c == ' '))
             $\langle$ Print the character. 3d $\rangle$ 
            prev_blank = (c == ' ');
    }

    return 0;
}
```

Uses `bool` 8b.

Root chunk (not used in this document).

Exercise 1-10

Process each character *c*.

5d $\langle unambiguous.c\ 5c \rangle + \equiv$

```
int c;

 $\langle$ For each character c until EOF 3c $\rangle$  {
```

5c $\langle unambiguous.c\ 5c \rangle \equiv$

\langle Include the standard I/O functions. 8c \rangle

```
int main()
{
```

This definition is continued in
chunks 5 and 6.

Root chunk (not used in this
document).

Replace each tab by `\t`.

```
6a <unambiguous.c 5c>+≡
    if (<the character is a tab 4d>)
        fputs("\\t", stdout);
```

Uses `fputs 8c` and `stdout 8c`.

Replace each backspace by `\b`.

```
6c <unambiguous.c 5c>+≡
    else if (<the character is a backspace 6b>)
        fputs("\\b", stdout);
```

Uses `fputs 8c` and `stdout 8c`.

Replace each backslash by `\\`.

```
6e <unambiguous.c 5c>+≡
    else if (<the character is a backslash 6d>)
        fputs("\\\\", stdout);
```

Uses `fputs 8c` and `stdout 8c`.

Otherwise print the character unchanged.

```
6f <unambiguous.c 5c>+≡
    else
        <Print the character. 3d>
```

Word Counting

```
6h <wc.c 3f>+≡
#define IN 1
#define OUT 0
```

Defines:

IN, used in chunks 7 and 8a.
OUT, used in chunks 7 and 8a.

```
6b <the character is a backspace 6b>≡
    c = '\b'
```

This code is used in chunk 6c.

```
6d <the character is a backslash 6d>≡
    c = '\\'
```

This code is used in chunk 6e.

Finally, close the `while` loop and exit.

```
6g <unambiguous.c 5c>+≡
    }

    return 0;
}
```

```

7  <wc.c 3f>+≡
    int main()
    {
        int c, nl, nw, nc, state;

        state = OUT;
        nl = nw = nc = 0;
        <For each character c until EOF 3c> {
            ++nc;
            if (<the character is a newline 4b>)
                ++nl;
            if (<the character is whitespace 4e>)
                state = OUT;
            else if (state == OUT) {
                state = IN;
                ++nw;
            }
        }

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

        return 0;
    }

```

Uses IN 6h, OUT 6h, and printf 8c.

Exercise 1-12

8a $\langle words.c\ 8a \rangle \equiv$
 $\langle Include\ the\ standard\ I/O\ functions.\ 8c \rangle$

```

#define IN    1
#define OUT   0

int main()
{
    int c, state;

    state = OUT;
     $\langle For\ each\ character\ c\ until\ EOF\ 3c \rangle$  {
         $\langle (the\ character\ is\ whitespace\ 4e) \rangle$  {
            if (state == IN)
                putchar('\n');
            state = OUT;
        } else {
            state = IN;
        }

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

    return 0;
}

```

Uses IN 6h, OUT 6h, and putchar 8c.
 Root chunk (not used in this document).

Common Headers

8b $\langle Include\ the\ boolean\ type\ and\ values.\ 8b \rangle \equiv$
`#include <stdbool.h>`

Defines:

`bool`, used in chunks 4f and 5b.

This code is used in chunks 3f and 5b.

8c $\langle Include\ the\ standard\ I/O\ functions.\ 8c \rangle \equiv$
`#include <stdio.h>`

Defines:

`fputs`, used in chunk 6.

`printf`, used in chunks 1-3 and 7.

`putchar`, used in chunks 2c, 3d, and 8a.

`puts`, used in chunks 2c and 3b.

`stdout`, used in chunk 6.

This code is used in chunks 1-3, 5, and 8a.

9 *<Include the standard string functions. 9>*≡
 #include <string.h>

Defines:

strlen, used in chunk 2c.

This code is used in chunk 2a.

Chunks

〈For each character *c* until EOF 3c〉 [3c](#), [3e](#), [4c](#), [5a](#), [5b](#), [5d](#), [7](#), [8a](#)
 〈Include the boolean type and values. 8b〉 [3f](#), [5b](#), [8b](#)
 〈Include the standard I/O functions. 8c〉 [1](#), [2a](#), [3e](#), [3f](#), [5b](#), [5c](#), [8a](#), [8c](#)
 〈Include the standard string functions. 9〉 [2a](#), [9](#)
 〈Print the character. 3d〉 [3d](#), [3e](#), [5b](#), [6f](#)
 〈catblanks.c 5b〉 [5b](#)
 〈copy.c 3e〉 [3e](#)
 〈fahr.cels.c 2a〉 [2a](#), [2b](#), [2c](#), [2d](#), [3a](#), [3b](#)
 〈hello.c 1〉 [1](#)
 〈the character is a backslash 6d〉 [6d](#), [6e](#)
 〈the character is a backspace 6b〉 [6b](#), [6c](#)
 〈the character is a newline 4b〉 [4b](#), [4c](#), [4e](#), [7](#)
 〈the character is a tab 4d〉 [4d](#), [4e](#), [6a](#)
 〈the character is whitespace 4e〉 [4e](#), [4f](#), [7](#), [8a](#)
 〈unambiguous.c 5c〉 [5c](#), [5d](#), [6a](#), [6c](#), [6e](#), [6f](#), [6g](#)
 〈wc.c 3f〉 [3f](#), [4a](#), [4c](#), [4f](#), [5a](#), [6h](#), [7](#)
 〈words.c 8a〉 [8a](#)

Index

IN: [6h](#), [7](#), [8a](#)
 LOWER: [2b](#), [3a](#)
 OUT: [6h](#), [7](#), [8a](#)
 STEP: [2b](#), [3a](#)
 UPPER: [2b](#), [3a](#)
 bool: [4f](#), [5b](#), [8b](#)
 celsfahr: [2d](#), [3b](#)
 char_count: [4a](#)
 fahr.cels: [3a](#), [3b](#)
 fputs: [6a](#), [6c](#), [6e](#), [8c](#)
 is_whitespace: [4f](#), [5a](#)
 line_count: [4c](#)
 printf: [1](#), [2c](#), [2d](#), [3a](#), [7](#), [8c](#)
 print_header: [2c](#), [2d](#), [3a](#)
 putchar: [2c](#), [3d](#), [8a](#), [8c](#)
 puts: [2c](#), [3b](#), [8c](#)
 stdout: [6a](#), [6c](#), [6e](#), [8c](#)
 strlen: [2c](#), [9](#)
 ws_count: [5a](#)