

ERIC BAILEY

# THE C PROGRAMMING LANGUAGE



# *Contents*

<i>Chapter One</i>	5
--------------------	---



# Chapter One

## *Hello, world!*

Covers Exercises 1-1 and 1-2.

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

5a `<hello.c 5a>≡`

This definition is continued in chunk 5b.  
Root chunk (not used in this document).

*<Include the standard I/O functions. 19b>*

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

5b `<hello.c 5a>+≡`

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

Uses `printf` 19b.

## *Fahrenheit-Celsius table*

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

5c `<fahrcls.c 5c>≡`

This definition is continued in chunks 5-7.  
Root chunk (not used in this document).

*<Include the standard I/O functions. 19b>*

*<Include the standard string functions. 19c>*

Declare some useful constants.

5d `<fahrcls.c 5c>+≡`

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

Defines:

LOWER, used in chunk 6.

STEP, used in chunk 6.

UPPER, used in chunk 6.

*Exercise 1-3*

6a `<fahrrels.c 5c>+≡`

```
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 6.

Uses `printf` 19b, `putchar` 19b, `puts` 19b, and `strlen` 19c.

*Exercise 1-4*

6b `<fahrrels.c 5c>+≡`

```
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 7a.

Uses `LOWER` 5d, `print_header` 6a, `printf` 19b, `STEP` 5d, and `UPPER` 5d.

*Exercise 1-5*

6c `<fahrrels.c 5c>+≡`

```
void fahrrels()
{
    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:

`fahrrels`, used in chunk 7a.

Uses `LOWER` 5d, `print_header` 6a, `printf` 19b, `STEP` 5d, and `UPPER` 5d.

*The main function*

7a  $\langle \text{fahrrels.c } 5c \rangle + \equiv$

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

    return 0;
}
```

Uses `celsfahr` 6b, `fahrrels` 6c, and `puts` 19b.

*Copy*

Covers Exercises 1-6 and 1-7.

7b  $\langle \text{copy.c } 7b \rangle \equiv$

This definition is continued in chunk 7c.  
Root chunk (not used in this document).

$\langle \text{Include the standard I/O functions. } 19b \rangle$

7c  $\langle \text{copy.c } 7b \rangle + \equiv$

```
int main()
{
    int c;
     $\langle \text{For each character } c \text{ until EOF } 19d \rangle$ 
     $\langle \text{Print the character. } 20a \rangle$ 

    return 0;
}
```

*Exercise 1-9*

7d  $\langle \text{catblanks.c } 7d \rangle \equiv$

This definition is continued in chunk 8a.  
Root chunk (not used in this document).

$\langle \text{Include the standard I/O functions. } 19b \rangle$

$\langle \text{Include the boolean type and values. } 19a \rangle$

8a  $\langle \text{catblanks.c 7d} \rangle + \equiv$

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

     $\langle \text{For each character } c \text{ until EOF 19d} \rangle \{$ 
        if (!(prev_blank && c == ' '))
             $\langle \text{Print the character. 20a} \rangle$ 
            prev_blank = (c == ' ');
    }

    return 0;
}
```

Uses bool 19a.

### Exercise 1-10

Process each character `c`.

8c  $\langle \text{unambiguous.c 8b} \rangle + \equiv$

```
int c;

 $\langle \text{For each character } c \text{ until EOF 19d} \rangle \{$ 

    Replace each tab by \t.


8d  $\langle \text{unambiguous.c 8b} \rangle + \equiv$



```
    if ( $\langle \text{the character is a tab 20d} \rangle$ )
        fputs("\\t", stdout);
```



Uses fputs 19b and stdout 19b.


```

Replace each backspace by `\b`.

8e  $\langle \text{unambiguous.c 8b} \rangle + \equiv$

```
    else if ( $\langle \text{the character is a backspace 20e} \rangle$ )
        fputs("\\b", stdout);
```

Uses fputs 19b and stdout 19b.

Replace each backslash by `\\`.

8f  $\langle \text{unambiguous.c 8b} \rangle + \equiv$

```
    else if ( $\langle \text{the character is a backslash 20f} \rangle$ )
        fputs("\\\\", stdout);
```

Uses fputs 19b and stdout 19b.

Otherwise print the character unchanged.

8g  $\langle \text{unambiguous.c 8b} \rangle + \equiv$

```
    else
         $\langle \text{Print the character. 20a} \rangle$ 
```

8b  $\langle \text{unambiguous.c 8b} \rangle \equiv$

This definition is continued in chunk 8.

Root chunk (not used in this document).

$\langle \text{Include the standard I/O functions. 19b} \rangle$

```
int main()
{
```

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

8h  $\langle \text{unambiguous.c 8b} \rangle + \equiv$

```
    }

    return 0;
}
```



*Character Counting***9a**  $\langle wc.c\ 9a \rangle \equiv$ 

This definition is continued in chunks **9** and **10**.  
 Root chunk (not used in this document).

$\langle Include\ the\ standard\ I/O\ functions.\ 19b \rangle$   
 $\langle Include\ the\ boolean\ type\ and\ values.\ 19a \rangle$

**9b**  $\langle wc.c\ 9a \rangle + \equiv$ 

```
double char_count()
{
    double nc;

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

    return nc;
}
```

Defines:

char\_count, never used.

*Line Counting***9c**  $\langle wc.c\ 9a \rangle + \equiv$ 

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

    nl = 0;
     $\langle For\ each\ character\ c\ until\ EOF\ 19d \rangle$ 
    if ( $\langle the\ character\ is\ a\ newline\ 20c \rangle$ )
        ++nl;

    return nl;
}
```

Defines:

line\_count, never used.

*Exercise 1-8***9d**  $\langle wc.c\ 9a \rangle + \equiv$ 

```
bool is_whitespace(int c)
{
    return ( $\langle the\ character\ is\ whitespace\ 20b \rangle$ );
}
```

Defines:

is\_whitespace, used in chunk **10a**.

Uses bool **19a**.

10a  $\langle wc.c\ 9a \rangle + \equiv$

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

     $\langle$ For each character c until EOF 19d $\rangle$ 
        if (is_whitespace(c))
            ++ns;

    return ns;
}
```

Defines:

ws\_count, never used.

Uses is\_whitespace 9d.

### Word Counting

10b  $\langle wc.c\ 9a \rangle + \equiv$

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

Defines:

IN, used in chunks 10–12.

OUT, used in chunks 10–12.

10c  $\langle wc.c\ 9a \rangle + \equiv$

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

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

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

    return 0;
}
```

Uses IN 10b, OUT 10b, and printf 19b.

*Exercise 1-12***11** *<words.c 11>*≡

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 10b, OUT 10b, and putchar 19b.

*Arrays**Exercise 1-13*

Vertical histogram

12 *<wordlength.c 12>*≡

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, w1;
    int length[MAX_WORD_LENGTH+1];

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

    state = OUT;
    w1 = 0;
    <For each character c until EOF 19d> {
        if (<the character is whitespace 20b>) {
            if (state == IN) {
                state = OUT;
                ++length[w1 ≤ MAX_WORD_LENGTH ? w1-1 : MAX_WORD_LENGTH];
            }
        } else {
            if (state == OUT) {
                state = IN;
                w1 = 0;
            }
            ++w1;
        }
    }

    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 10b, OUT 10b, printf 19b, and putchar 19b.

### *Exercise 1-14*

13a *<charfreq.c 13a>*≡

This definition is continued in chunks 13b and 14a.  
Root chunk (not used in this document).

*<Include the standard I/O functions. 19b>*

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

13b *<charfreq.c 13a>*+≡

```
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 14a.

Uses printf 19b.

14a `<charfreq.c 13a>+≡`

```

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 13b`, `putchar 19b`, and `stdout 19b`.

## Functions

### Exercise 1-16

14b `<longestline.c 14b>≡`  
 This definition is continued in chunks 14–16.  
 Root chunk (not used in this document).

```

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

```

14c `<longestline.c 14b>+≡`  
*<Include the standard I/O functions. 19b>*  
`#include "get_line.h"`

Define the maximum line length to read into memory.

14d `<longestline.c 14b>+≡`  
`/// The maximum line length to read into memory.`  
`#define MAXLINE 80`

Defines:

`MAXLINE`, used in chunks 15 and 17.

15a  $\langle \text{longestline.c 14b} \rangle + \equiv$   
`void copy(char to[], char from[]);`

Uses copy 16a.

15b  $\langle \text{longestline.c 14b} \rangle + \equiv$   
`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) {`

Uses copy 16a and MAXLINE 14d 16d.

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

15c  $\langle \text{longestline.c 14b} \rangle + \equiv$   
`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 new-line.

15d  $\langle \text{longestline.c 14b} \rangle + \equiv$   
`if (max ≥ MAXLINE && longest[MAXLINE-1] ≠ '\n')  
 fputs("...\n", stdout);`

Uses fputs 19b, MAXLINE 14d 16d, and stdout 19b.

15e  $\langle \text{longestline.c 14b} \rangle + \equiv$   
`}  
  
 return 0;  
}`

16a  $\langle \text{longestline.c 14b} \rangle + \equiv$

```

/* 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 15.

### *Exercise 1-17*

16b  $\langle \text{longlines.c 16b} \rangle \equiv$

This definition is continued in chunks 16 and 17.

Root chunk (not used in this document).

```

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

```

16c  $\langle \text{longlines.c 16b} \rangle + \equiv$

```

<Include the standard I/O functions. 19b>
#include "get_line.h"

```

Define the maximum line length to read into memory.

16d  $\langle \text{longlines.c 16b} \rangle + \equiv$

```

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

```

Defines:

MAXLINE, used in chunks 15 and 17.

Define the minimum line length to be considered long.

16e  $\langle \text{longlines.c 16b} \rangle + \equiv$

```

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

```

Defines:

MINLENGTH, used in chunk 17.



```
17  <longlines.c 16b>+≡
    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;
    }
```

Uses fputs 19b, MAXLINE 14d 16d, MINLENGTH 16e, printf 19b, and stdout 19b.



# Common

## Headers

**19a**    *⟨Include the boolean type and values. 19a⟩*≡  
This code is used in chunks **7d** and **9a**.

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

Defines:

`bool`, used in chunks **8a** and **9d**.

**19b**    *⟨Include the standard I/O functions. 19b⟩*≡  
This code is used in chunks **5**, **7–9**, **11–14**, and **16c**.

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

Defines:

`fputs`, used in chunks **8**, **14a**, **15d**, and **17**.

`printf`, used in chunks **19b**, **5**, **6**, **10c**, **12**, **13b**, **15c**, and **17**.

`putchar`, used in chunks **6a**, **11**, **12**, **14a**, and **20a**.

`puts`, used in chunks **6a** and **7a**.

`stdout`, used in chunks **8**, **14a**, **15d**, and **17**.

**19c**    *⟨Include the standard string functions. 19c⟩*≡  
This code is used in chunk **5c**.

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

Defines:

`strlen`, used in chunk **6a**.

## Patterns

### Control

**19d**    *⟨For each character `c` until EOF 19d⟩*≡  
This code is used in chunks **7–12** and **14a**.

```
while ((c = getchar()) ≠ EOF)
```

*I/O*

**20a**  $\langle \textit{Print the character. 20a} \rangle \equiv$   
 This code is used in chunks **7** and **8**.

```
    putchar(c);
```

Uses `putchar` **19b**.

*Predicates*

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

**20b**  $\langle \textit{the character is whitespace 20b} \rangle \equiv$   
 This code is used in chunks **9–12**.

```
c = ' ' ||  $\langle \textit{the character is a newline 20c} \rangle$  ||  $\langle \textit{the character is a tab 20d} \rangle$ 
```

**20c**  $\langle \textit{the character is a newline 20c} \rangle \equiv$   
 This code is used in chunks **9c**, **10c**, and **20b**.

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

**20d**  $\langle \textit{the character is a tab 20d} \rangle \equiv$   
 This code is used in chunks **8d** and **20b**.

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

**20e**  $\langle \textit{the character is a backspace 20e} \rangle \equiv$   
 This code is used in chunk **8e**.

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

**20f**  $\langle \textit{the character is a backslash 20f} \rangle \equiv$   
 This code is used in chunk **8f**.

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

# Chunks

*<catblanks.c 7d>* [7d](#), [8a](#)  
*<charfreq.c 13a>* [13a](#), [13b](#), [14a](#)  
*<copy.c 7b>* [7b](#), [7c](#)  
*<fahrceles.c 5c>* [5c](#), [5d](#), [6a](#), [6b](#), [6c](#), [7a](#)  
*<For each character c until EOF 19d>* [7c](#), [8a](#), [8c](#), [9c](#), [10a](#), [10c](#), [11](#), [12](#),  
[14a](#), [19d](#)  
*<hello.c 5a>* [5a](#), [5b](#)  
*<Include the boolean type and values. 19a>* [7d](#), [9a](#), [19a](#)  
*<Include the standard I/O functions. 19b>* [5a](#), [5c](#), [7b](#), [7d](#), [8b](#), [9a](#), [11](#),  
[12](#), [13a](#), [14c](#), [16c](#), [19b](#)  
*<Include the standard string functions. 19c>* [5c](#), [19c](#)  
*<longestline.c 14b>* [14b](#), [14c](#), [14d](#), [15a](#), [15b](#), [15c](#), [15d](#), [15e](#), [16a](#)  
*<longlines.c 16b>* [16b](#), [16c](#), [16d](#), [16e](#), [17](#)  
*<Print the character. 20a>* [7c](#), [8a](#), [8g](#), [20a](#)  
*<the character is a backslash 20f>* [8f](#), [20f](#)  
*<the character is a backspace 20e>* [8e](#), [20e](#)  
*<the character is a newline 20c>* [9c](#), [10c](#), [20b](#), [20c](#)  
*<the character is a tab 20d>* [8d](#), [20b](#), [20d](#)  
*<the character is whitespace 20b>* [9d](#), [10c](#), [11](#), [12](#), [20b](#)  
*<unambiguous.c 8b>* [8b](#), [8c](#), [8d](#), [8e](#), [8f](#), [8g](#), [8h](#)  
*<wc.c 9a>* [9a](#), [9b](#), [9c](#), [9d](#), [10a](#), [10b](#), [10c](#)  
*<wordlength.c 12>* [12](#)  
*<words.c 11>* [11](#)



# *Index*

bool: [8a](#), [9d](#), [19a](#)  
celsfahr: [6b](#), [7a](#)  
char\_count: [9b](#)  
copy: [15a](#), [15b](#), [16a](#)  
fahrccels: [6c](#), [7a](#)  
fputs: [8d](#), [8e](#), [8f](#), [14a](#), [15d](#), [17](#),  
    [19b](#)  
IN: [10b](#), [10c](#), [11](#), [12](#)  
is\_whitespace: [9d](#), [10a](#)  
line\_count: [9c](#)  
LOWER: [5d](#), [6b](#), [6c](#)  
MAXLINE: [14d](#), [15b](#), [15d](#), [16d](#), [17](#)  
MINLENGTH: [16e](#), [17](#)  
OUT: [10b](#), [10c](#), [11](#), [12](#)  
prchar: [13b](#), [14a](#)  
print\_header: [6a](#), [6b](#), [6c](#)  
printf: [19b](#), [5b](#), [6a](#), [6b](#), [6c](#), [10c](#),  
    [12](#), [13b](#), [15c](#), [17](#), [19b](#)  
putchar: [6a](#), [11](#), [12](#), [14a](#), [19b](#),  
    [20a](#)  
puts: [6a](#), [7a](#), [19b](#)  
stdout: [8d](#), [8e](#), [8f](#), [14a](#), [15d](#), [17](#),  
    [19b](#)  
STEP: [5d](#), [6b](#), [6c](#)  
strlen: [6a](#), [19c](#)  
UPPER: [5d](#), [6b](#), [6c](#)  
ws\_count: [10a](#)