

Chapter 6 - Introduction to C Programming

Outline

1. Introduction
2. A Simple C Program: Printing a Line of Text
3. Another Simple C Program: Adding Two Integers
4. Arithmetic in C
5. Decision Making: Equality and Relational Operators



Objectives

- In this chapter, you will learn:
 - To be able to write simple computer programs in C.
 - To be able to use simple input and output statements.
 - To become familiar with fundamental data types.
 - To understand computer memory concepts.
 - To be able to use arithmetic operators.
 - To understand the precedence of arithmetic operators.
 - To be able to write simple decision making statements.



A Simple C Program: Printing a Line of Text

```
1  /* Fig. 2.1: fig02_01.c
2     A first program in C */
3  #include <stdio.h>
4
5  /* function main begins program execution */
6  int main()
7  {
8     printf( "Welcome to C!\n" );
9
10     return 0; /* indicate that program ended successfully */
11
12 } /* end function main */
```

```
Welcome to C!
```



A Simple C Program: Printing a Line of Text

- `int main()`
 - C++ programs contain one or more functions, exactly one of which must be `main`
 - Parenthesis used to indicate a function
 - `int` means that `main` "returns" an integer value
 - Braces (`{` and `}`) indicate a block
 - The bodies of all functions must be contained in braces



A Simple C Program: Printing a Line of Text

| Escape Sequence | Description |
|-----------------|---|
| <code>\n</code> | Newline. Position the cursor at the beginning of the next line. |
| <code>\t</code> | Horizontal tab. Move the cursor to the next tab stop. |
| <code>\a</code> | Alert. Sound the system bell. |
| <code>\\</code> | Backslash. Insert a backslash character in a string. |
| <code>\"</code> | Double quote. Insert a double quote character in a string. |

Fig. 2.2 Some common escape sequences.





```
1  /* Fig. 2.3: fig02_03.c
2     Printing on one line with two printf statements */
3  #include <stdio.h>
4
5  /* function main begins program execution */
6  int main()
7  {
8     printf( "welcome " );
9     printf( "to C!\n" );
10
11     return 0; /* indicate that program ended successfully */
12
13 } /* end function main */
```

```
Welcome to C!
```



```
1  /* Fig. 2.4: fig02_04.c
2     Printing multiple lines with a single printf */
3  #include <stdio.h>
4
5  /* function main begins program execution */
6  int main()
7  {
8     printf( "welcome\n\tto\n\tC!\n" );
9
10     return 0; /* indicate that program ended successfully */
11
12 } /* end function main */
Welcome
to
C!
```



Outline

fig02_05.c

```
1  /* Fig. 2.5: fig02_05.c
2     Addition program */
3  #include <stdio.h>
4
5  /* function main begins program execution */
6  int main()
7  {
8     int integer1; /* first number to be input by user */
9     int integer2; /* second number to be input by user */
10    int sum;      /* variable in which sum will be stored */
11
12    printf( "Enter first integer\n" ); /* prompt */
13    scanf( "%d", &integer1 );        /* read an integer */
14
15    printf( "Enter second integer\n" ); /* prompt */
16    scanf( "%d", &integer2 );        /* read an integer */
17
18    sum = integer1 + integer2;        /* assign total to sum */
19
20    printf( "Sum is %d\n", sum );     /* print sum */
21
22    return 0; /* indicate that program ended successfully */
23
24 } /* end function main */
```

```
Enter first integer
45
Enter second integer
72
Sum is 117
```



Outline

9

Program Output

Arithmetic

- Arithmetic operators:

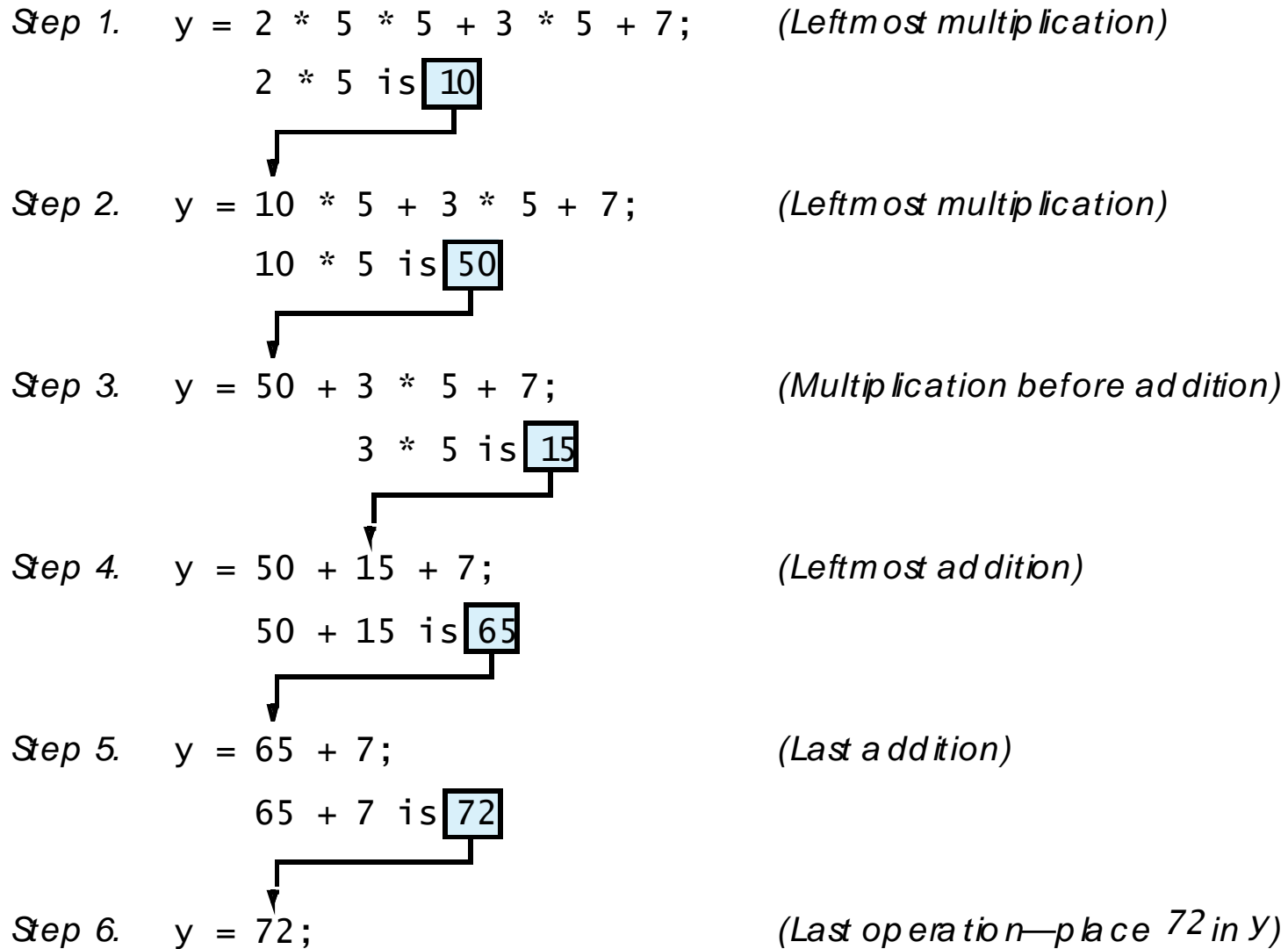
| C operation | Arithmetic operator | Algebraic expression | C expression |
|----------------|---------------------|----------------------|--------------------|
| Addition | + | $f + 7$ | <code>f + 7</code> |
| Subtraction | - | $p - c$ | <code>p - c</code> |
| Multiplication | * | bm | <code>b * m</code> |
| Division | / | x / y | <code>x / y</code> |
| Modulus | % | $r \text{ mod } s$ | <code>r % s</code> |

- Rules of operator precedence:

| Operator(s) | Operation(s) | Order of evaluation (precedence) |
|-------------|-----------------------------------|--|
| \emptyset | Parentheses | Evaluated first. If the parentheses are nested, the expression in the innermost pair is evaluated first. If there are several pairs of parentheses “on the same level” (i.e., not nested), they are evaluated left to right. |
| *, /, or % | Multiplication, Division, Modulus | Evaluated second. If there are several, they are evaluated left to right. |
| + or - | Addition Subtraction | Evaluated last. If there are several, they are evaluated left to right. |



Decision Making: Equality and Relational Operators



Decision Making: Equality and Relational Operators

- Executable statements
 - Perform actions (calculations, input/output of data)
 - Perform decisions
 - May want to print "pass" or "fail" given the value of a test grade
- `if` control statement
 - Simple version in this section, more detail later
 - If a condition is `true`, then the body of the `if` statement executed
 - 0 is `false`, non-zero is `true`
 - Control always resumes after the `if` structure
- Keywords
 - Special words reserved for C
 - Cannot be used as identifiers or variable names



Decision Making: Equality and Relational Operators

| Standard algebraic equality operator or relational operator | C equality or relational operator | Example of C condition | Meaning of C condition |
|---|-----------------------------------|------------------------|---------------------------------|
| <i>Equality Operators</i> | | | |
| = | == | $x == y$ | x is equal to y |
| ≠ | != | $x != y$ | x is not equal to y |
| <i>Relational Operators</i> | | | |
| > | > | $x > y$ | x is greater than y |
| < | < | $x < y$ | x is less than y |
| >= | >= | $x >= y$ | x is greater than or equal to y |
| <= | <= | $x <= y$ | x is less than or equal to y |





```
1  /* Fig. 2.13: fig02_13.c
2     Using if statements, relational
3     operators, and equality operators */
4  #include <stdio.h>
5
6  /* function main begins program execution */
7  int main()
8  {
9     int num1, /* first number to be read from user */
10    int num2; /* second number to be read from user */
11
12    printf( "Enter two integers, and I will tell you\n" );
13    printf( "the relationships they satisfy: " );
14
15    scanf( "%d%d", &num1, &num2 ); /* read two integers */
16
17    if ( num1 == num2 ) {
18        printf( "%d is equal to %d\n", num1, num2 );
19    } /* end if */
20
21    if ( num1 != num2 ) {
22        printf( "%d is not equal to %d\n", num1, num2 );
23    } /* end if */
24
```

**fig02_13.c (Part 1
of 2)**



Outline

fig02_13.c (Part 2 of 2)

```
25  if ( num1 < num2 ) {
26      printf( "%d is less than %d\n", num1, num2 );
27  } /* end if */
28
29  if ( num1 > num2 ) {
30      printf( "%d is greater than %d\n", num1, num2 );
31  } /* end if */
32
33  if ( num1 <= num2 ) {
34      printf( "%d is less than or equal to %d\n", num1, num2 );
35  } /* end if */
36
37  if ( num1 >= num2 ) {
38      printf( "%d is greater than or equal to %d\n", num1, num2 );
39  } /* end if */
40
41  return 0; /* indicate that program ended successfully */
42
43 } /* end function main */
```

```
Enter two integers, and I will tell you
the relationships they satisfy: 3 7
3 is not equal to 7
3 is less than 7
3 is less than or equal to 7
```

Program Output



Outline

Program Output (continued)

```
Enter two integers, and I will tell you
the relationships they satisfy: 22 12
22 is not equal to 12
22 is greater than 12
22 is greater than or equal to 12
```

```
Enter two integers, and I will tell you
the relationships they satisfy: 7 7
7 is equal to 7
7 is less than or equal to 7
7 is greater than or equal to 7
```

Decision Making: Equality and Relational Operators

| Operators | | | | Associativity |
|-----------|----|---|----|---------------|
| * | / | % | | left to right |
| + | - | | | left to right |
| < | <= | > | >= | left to right |
| == | != | | | left to right |
| = | | | | right to left |

Fig. 2.14 Precedence and associativity of the operators discussed so far.



Decision Making: Equality and Relational Operators

| Keywords | | | |
|----------|--------|----------|----------|
| auto | double | int | struct |
| break | else | long | switch |
| case | enum | register | typedef |
| char | extern | return | union |
| const | float | short | unsigned |
| continue | for | signed | void |
| default | goto | sizeof | volatile |
| do | if | static | while |

Fig. 2.15 C's reserved keywords.



Les types de données

| Data types | printf conversion specifications | scanf conversion specifications |
|-------------------|----------------------------------|---------------------------------|
| long double | %Lf | %Lf |
| double | %f | %lf |
| float | %f | %f |
| unsigned long int | %lu | %lu |
| long int | %ld | %ld |
| unsigned int | %u | %u |
| int | %d | %d |
| short | %hd | %hd |
| char | %c | %c |

Fig. 5.5 Promotion hierarchy for data types.

