

Outlines

- Equality and Relational Operators
- A Simple C Program: Relations
- Keywords
- Algorithm
- Flow Chart
- Pseudo Code

Equality and Relational Operators

Algebraic equality 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

Fig. 2.12 | Equality and relational operators.

Equality and Relational Operators ...



Common Programming Error 2.17

A syntax error occurs if the two symbols in any of the operators $==$, $!=$, $>=$ and $<=$ are separated by spaces.



Common Programming Error 2.18

A syntax error occurs if the two symbols in any of the operators $!=$, $>=$ and $<=$ are reversed as in $=!$, $=>$ and $=<$, respectively.

Equality and Relational Operators ...



Common Programming Error 2.19

Confusing the equality operator `==` with the assignment operator.



Common Programming Error 2.20

Placing a semicolon immediately to the right of the right parenthesis after the condition in an `if` statement.

Equality and Relational Operators ...

Operators				Associativity
()				left to right
*	/	%		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.

A Simple C Program: Relations

```
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( void )
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 */
```

Fig. 2.13 | Using if statements, relational operators, and equality operators. (Part I of 3.)

A Simple C Program: Relations ...

```
24
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 } /* end function main */
```

Fig. 2.13 | Using if statements, relational operators, and equality operators. (Part 2 of 3.)

A Simple C Program: Relations ...

```
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
```

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

Fig. 2.13 | Using `if` statements, relational operators, and equality operators. (Part 3 of 3.)

A Simple C Program: Relations ...



Good Programming Practice 2.12

Indent the statement(s) in the body of an `if` statement.



Good Programming Practice 2.13

Place a blank line before and after every `if` statement in a program for readability.

A Simple C Program: Relations ...



Good Programming Practice 2.14

Although it is allowed, there should be no more than one statement per line in a program.



Common Programming Error 2.21

Placing commas (when none are needed) between conversion specifiers in the format control string of a scanf statement.

Keywords

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

Keywords added in C99

`_Bool _Complex _Imaginary inline restrict`

Fig. 2.15 | C's keywords.

Algorithm

- Before writing a program to solve a particular problem, it's essential to have a thorough understanding of the problem and a carefully planned approach to solving the problem.
- The solution to any computing problem involves executing a series of actions in a specific order.

Algorithm ...

- A **procedure** for solving a problem in terms of
 - the **actions** to be executed, and
 - the **order** in which these actions are to be executedis called an **algorithm**.
- Correctly specifying the order in which the actions are to be executed is important.
- Specifying the order in which statements are to be executed in a computer program is called **program control**.

Flow Chart

- A **flowchart** is a graphical representation of an algorithm or of a portion of an algorithm.
- Flowcharts are drawn using certain special-purpose symbols such as **rectangles**, **diamonds**, **ovals**, and **small circles**; these symbols are connected by arrows called **flowlines**.
- Like pseudocode, flowcharts are useful for developing and representing algorithms, although pseudocode is preferred by most programmers.

Flow Chart ...

- We use the **rectangle** symbol, also called the **action** symbol, to indicate any type of action including a calculation or an input/output operation.
- The **flowlines** in the figure indicate the order in which the actions are performed.
- C allows us to have as many actions as we want in a sequence structure.

Flow Chart ...

- When drawing a flowchart that represents a complete algorithm, an **oval** symbol containing the word “**Begin**” is the first symbol used in the flowchart; an **oval** symbol containing the word “**End**” is the last symbol used.
- When drawing only a portion of an algorithm, the oval symbols are omitted in favor of using small circle symbols, also called connector symbols.
- Perhaps the most important flowcharting symbol is the **diamond** symbol, also called the **decision symbol**, which indicates that a decision is to be made.

Pseudo Code

- Pseudocode is an artificial and informal language that helps you develop algorithms.
- Pseudocode is similar to everyday English; it's convenient and user friendly although it's not an actual computer programming language.
- Pseudocode programs are not executed on computers.
- Rather, they merely help you “think out” a program before attempting to write it in a programming language such as C.

Pseudo Code ...

- Pseudocode consists purely of characters, so you may conveniently type pseudocode programs into a computer using an editor program.
- A carefully prepared pseudocode program may be converted easily to a corresponding C program.
- Pseudocode consists only of action statements—those that are executed when the program has been converted from pseudocode to C and is run in C.

Pseudo Code ...

- Definitions are not executable statements.
- They're messages to the compiler.
- For example, the definition

int i;

simply tells the compiler the type of variable i and instructs the compiler to reserve space in memory for the variable.

- But this definition does not cause any action—such as input, output, or a calculation—to occur when the program is executed.

Pseudo Code ...

- Some programmers choose to list each variable and briefly mention the purpose of each at the beginning of a pseudocode program.