Decision Making in C

Dr Bhanu

Decision Making: Equality and Relational Operators

- Executable C statements either perform actions (such as calculations or input or output of data) or make decisions
    - If a student's grade in an exam is greater than or equal to 60 and if it is to print the message "Congratulations! You passed."
- We use if statement that allows a program to take a decision based on the truth or falsity of a statement of fact called a condition.

The if Selection Statement (single selection)

If statement may be written in C as

- *Version 1*

```
if ( grade >= 60 ) {
  printf( "Passed\n" );
}
```

- *Version 2*

```
if ( grade >= 60 )
{
  printf( "Passed\n" );
}
```

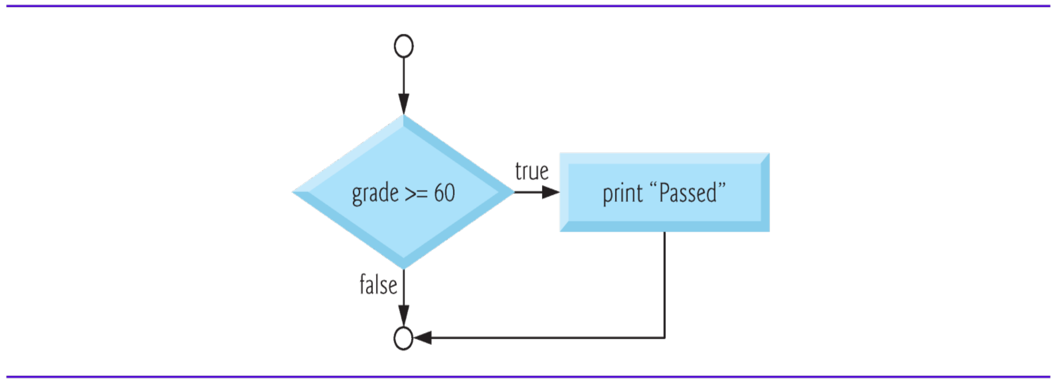


Fig. 3.2 | Flowcharting the single-selection `if` statement.

- The flowchart above illustrates the **single-selection** `if` statement.
- It contains what is perhaps the most important flowcharting symbol—the diamond symbol, also called the decision symbol, which indicates that a decision is to be made.
- The decision symbol contains an expression, such as a condition, that can be either true or false.

The if Selection Statement

- The decision symbol has two flowlines emerging from it.
    - One indicates the direction to take when the expression in the symbol is true;
    - The other indicates the direction to take when the expression is false.
- Decisions can be based on conditions containing relational or equality operators.
- In fact, a decision can be based on any expression
    - if the expression evaluates to zero, it’s treated as false
    - if it evaluates to nonzero, it’s treated as true.

Decision Making

- If the condition is met (i.e., the condition is true) the statement in the body of the `if` statement is executed.
- If the condition is not met (i.e., the condition is false) the body statement is not executed.
- Whether the body statement is executed or not, after the `if` statement completes, execution proceeds with the next statement after the `if` statement.
- Conditions in `if` statements are formed by using the equality operators and relational operators.

The if Selection Statement

- A left brace, `{`, begins the body of each `if` statement
- A corresponding right brace, `}`, ends each `if` statement’s body
- Any number of statements can be placed in the body of an `if` statement. They will be executed in the same sequential order in which they appear.
- In C programs, white space characters such as tabs, newlines and spaces are normally ignored.

Equality and Relational Operators

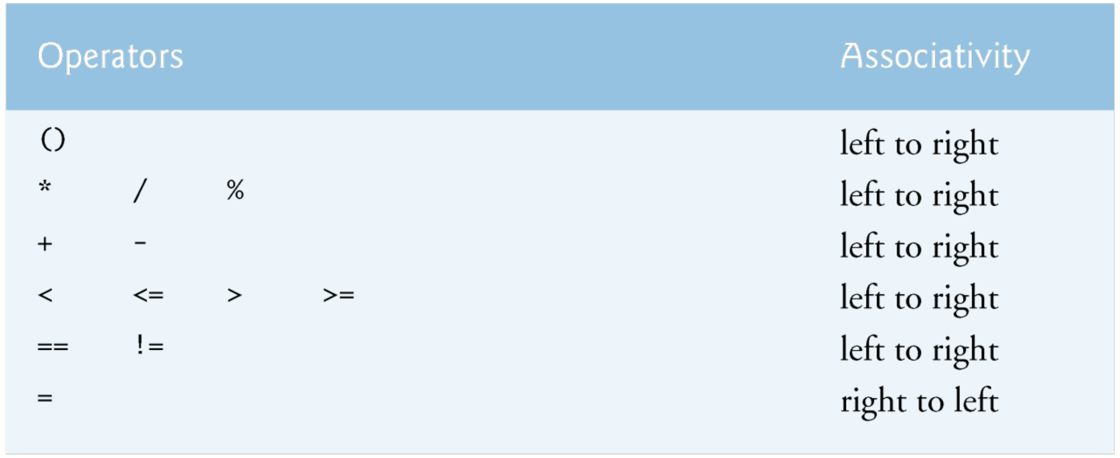
Algebraic equality or relational operator	C equality or relational operator	Example of C condition	Meaning of C condition
Equality operators			
=	`==`	`x == y`	x is equal to y
≠	`!=`	`x != y`	x is not equal to y
Relational operators			
>	`>`	`x > y`	x is greater than y
<	`<`	`x < y`	x is less than y
≥	`>=`	`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.

Relational Operators

- The relational operators all have the same level of precedence and they associate left to right.
- The equality operators have a lower level of precedence than the relational operators and they also associate left to right.
- In C, a condition may actually be any expression that generates a zero (false) or nonzero (true) value.

Precedence and Associativity



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.

= is an assignment operator.
== is an Equality operator

```c
/* Fig. 2.13: fig02_13.c
   Using if statements, relational
   operators, and equality operators */
#include <stdio.h>

/* function main begins program execution */
int main( void )
{
   int num1; /* first number to be read from user  */
   int num2; /* second number to be read from user */

   printf( "Enter two integers, and I will tell you\n" );
   printf( "the relationships they satisfy: " );

   scanf( "%d%d", &num1, &num2 ); /* read two integers */

   if ( num1 == num2 ) {
      printf( "%d is equal to %d\n", num1, num2 );
   } /* end if */

   if ( num1 != num2 ) {
      printf( "%d is not equal to %d\n", num1, num2 );
   } /* end if */
```

Fig. 2.13 | Using `if` statements, relational operators, and equality operators. (Part 1 of 3.)

```

    if ( num1 < num2 ) {
       printf( "%d is less than %d\n", num1, num2 );
    } /* end if */

    if ( num1 > num2 ) {
       printf( "%d is greater than %d\n", num1, num2 );
    } /* end if */

    if ( num1 <= num2 ) {
       printf( "%d is less than or equal to %d\n", num1, num2 );
    } /* end if */

    if ( num1 >= num2 ) {
       printf( "%d is greater than or equal to %d\n", num1, num2 );
    } /* end if */

    return 0; /* indicate that program ended successfully */
 } /* end function main */
```

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

```
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
22 is not equal to 12
22 is greater than 12
22 is greater than or equal to 12
```

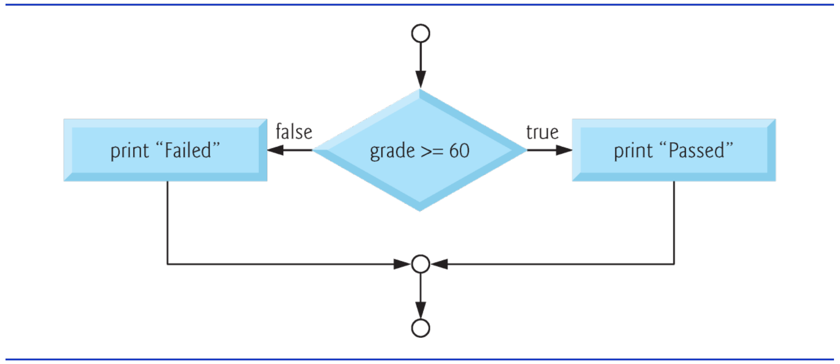
```
Enter two integers, and I will tell you
the relationships they satisfy: 7 7
7 is equal to 7
7 is less than or equal to 7
7 is greater than or equal to 7
```

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

The if…else Selection Statement

- Different condition…**Double selection**
  - If student's grade is greater than or equal to 60
    Print "Passed"
    else
    Print "Failed"

```
if ( grade >= 60 ) {
   printf( "Passed\n" );
} /* end if */
else {
   printf( "Failed\n" );
} /* end else */
```



Flowcharting the double-selection if…else statement.

The if…else (Alternate version)

```
printf( "%s\n", grade >= 60 ? "Passed" : "Failed" );
```

- C provides the conditional operator (?:) which is closely related to the if…else statement.
- The conditional operator is C's only ternary operator—it takes three operands.
- The operands together with the conditional operator form a conditional expression.
- The first operand is a condition.
- The second operand is the value for the entire conditional expression if the condition is true and the third operand is the value if the condition is false.

The `if…else` (Multiple selection)

- Nested if…else statements test for multiple cases by placing `if…else` statements inside `if…else` statements.
- For example, the following statement will print `A` for exam grades greater than or equal to `90`, `B` for grades greater than or equal to `80`, `C` for grades greater than or equal to `70`, `D` for grades greater than or equal to `60`, and `F` for all other grades.

  -
    ```
    If student’s grade is greater than or equal to 90
    Print “A”
    else
    If student’s grade is greater than or equal to 80
    Print “B”
    else
    If student’s grade is greater than or equal to 70
    Print “C”
    else
    If student’s grade is greater than or equal to 60
    Print “D”
    else
    Print “F”
    ```

The `if...else` (Multiple selection)

-
  ```
  if ( grade >= 90 )
     printf( "A\n" );
  else
     if ( grade >= 80 )
        printf("B\n");
     else
        if ( grade >= 70 )
           printf("C\n");
        else
           if ( grade >= 60 )
              printf( "D\n" );
           else
              printf( "F\n" );
  ```

The `if…else` (Multiple selection)

- Many C programmers prefer to write the preceding `if` statement as

  -
    ```
    if ( grade >= 90 )
       printf( "A\n" );
    else if ( grade >= 80 )
       printf( "B\n" );
    else if ( grade >= 70 )
       printf( "C\n" );
    else if ( grade >= 60 )
       printf( "D\n" );
    else
       printf( "F\n" );
    ```

The `if…else` (Multiple selection)

- As far as the C compiler is concerned, both forms are equivalent.
- The latter form is popular because it avoids the deep indentation of the code to the right.
- The `if` selection statement expects only one statement in its body.
- To include several statements in the body of an `if`, enclose the set of statements in braces (`{` and `}`).
- A set of statements contained within a pair of braces is called a compound statement or a block.

The `if…else` Selection Statement

- The following example includes a compound statement in the `else` part of an `if…else` statement.

  -
    ```
    if ( grade >= 60 ) {
       printf( "Passed.\n" );
    } /* end if */
    else {
       printf( "Failed.\n" );
       printf( "You must take this course again.\n" );
    } /* end else */
    ```

The `if…else` Selection Statement

- In this case, if grade is less than `60`, the program executes both `printf` statements in the body of the `else` and prints

  -
    ```
    Failed.
    You must take this course again.
    ```
- Notice the braces surrounding the two statements in the `else` clause.
- These braces are important. Without the braces, the statement

```
printf( "You must take this course again.\n" );
```

- would be outside the body of the `else` part of the `if`, and would execute regardless of whether the grade was less than 60.