Fundamentals of Computer Programming

C Programming
2. Statements

Contents

- True and False in C
- Selection Statements
  - if, switch
- Iteration Statements
  - while, for, do-while
- Jump Statements
  - break, continue, goto, return
- Expression Statements
- Block Statements

True and False in C

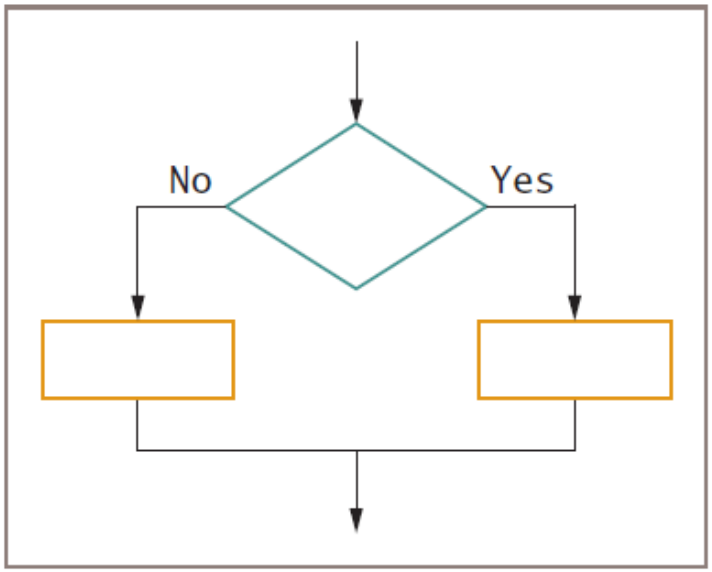
- Many C statements rely upon a *conditional expression* that determines what course of action is to be taken.
- A conditional expression evaluates to either a *true* or *false* value.
- In C, true is any *nonzero* value, including negative numbers. A false value is *0*.

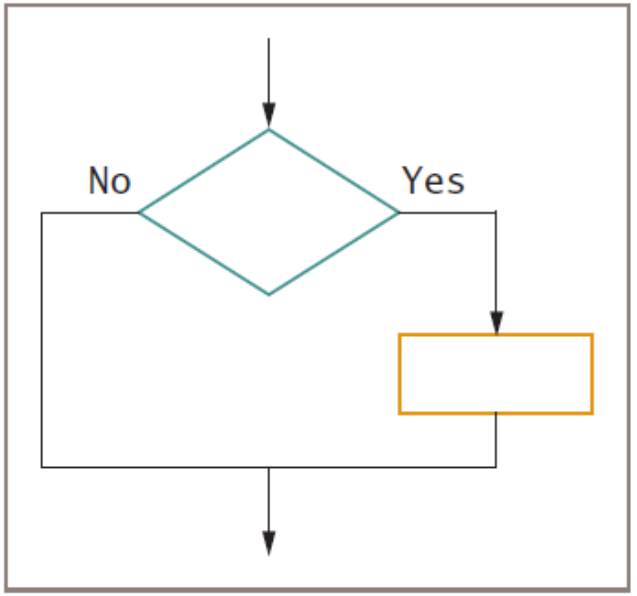
Selection Statements

- C supports two selection statements: **if** and **switch**.
- In addition, the ? operator is an alternative to **if** in certain circumstances.

if

- `if (expression) statement;`
  `else statement;`
- where a *statement* may consist of a single statement, a block of statements, or nothing (in the case of empty statements). The *else* clause is optional.





Exercise – Overtime Payroll Program

- Draw a flowchart and write a program that computes pay for employees.
- The program displays the weekly pay for each employee at the same hourly rate ($10.00) and assumes that there are no payroll deductions.
- The program contains a typical dual-alternative selection that determines whether an employee has worked more than a standard worksheet (40 hours), and pays one and one-half times the employee's usual hourly rate hours worked in excess of 40 per week.

[2] pp. 64-65

Nested ifs

- A **nested if** is an if that is the target of another **if** or **else**.
- In a nested if, an **else** statement always refers to the nearest **if** statement that is *within the same block* as the **else** and that is not already associated with an **else**.

The ? Alternative

- Ref: [2] pp.69-71

switch

- Ref: [2] pp.72-76

Iteration Statements

- In C, and all other modern programming languages, **iteration** statements (also called **loops**) allow a set of **instructions** to be repeatedly executed until a certain **condition** is reached.
- This **condition** may be predetermined (as in the **for** loop) or open ended (as in the **while** and **do-while** loops).

The for Loop

- Ref: [2] pp. 76-83

The while Loop

```
while (condition) statement;
```

- where *statement* is either an empty statement, a single statement, or a block of statements.
- The *condition* may be any expression, and true is any nonzero value.
- The loop iterates while the condition is *true*.
- When the condition becomes *false*, program control passes to the line of code immediately following the loop.

[2] pp. 83-86

The do-while Loop

- Ref: [2] pp. 86-87