Selection Structures

CEN111 Algorithms and Programming I

Control Structures

- Sequence structure
- Selection structure
- Iteration structure

Control Structures

- Selection Statements in C
  - The `if` single-selection statement selects (performs) an action (or group of actions) only if a condition is true
  - The `if…else` double-selection statement performs one action (or group of actions) if a condition is true and a different action (or group of actions) if the condition is false.
  - The `switch` multiple-selection statement performs one of many different actions, depending on the value of an expression.

Control Structures

- Iteration Statements in C
  - `while`
  - `do…while`
  - `for`
- These statements perform tasks repeatedly

Decision Making

- Executable statements perform actions like calculations, input and output, or make decisions
- A condition is an expression that can be true or false
- This section introduces the if statement
  - Makes a decision based on a condition’s value
  - If true, the statement in the if statement’s body executes
  - Otherwise, it does not

Decision Making: Equality and Relational Operators

- Equality and Relational Operators
  - Conditions are formed using the equality and relational operators

Decision Making: Relational Operators

Algebraic equality or relational operator	C relational operator	Sample C condition	Meaning of C condition
>	>	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
=	==	x == y	x is equal to y
≠	!=	x != y	x is not equal to y

Relational Operators

- The relational operators are `<`, `>`, `<=`, and `>=`.
- Take 2 expressions as operands
- Yield either the int value 0 (false) or the int value 1 (true).

- Example: assume a = 1, b=2.

Expression	Value
`a <= b`	1
`a < b-5`	0
`a + 10 / b <= -3 + 8`	0

Equality Operators

- The equality operators are == and !=.
- Yield either the int value 0 or the int value 1.

Examples: assume `a=1, b=2, ch='A'`

Expression	Value
`a == b`	0
`a != b`	1
`ch < 'B'`	1
`a+b == -2 * 3`	0

Equality Operators

- Note carefully that the two expressions `a == b` and `a = b` are visually similar.

- The expressions
  - `a == b` is a test for equality.
  - `a = b` is an assignment expression

Logical Operators

- logical expression: an expression that uses one or more of the logical operators
  - `&&` (and)
  - `||` (or)
  - `!` (not)

Logical Operators

- Expressions connected by `&&` or `||` are evaluated left to right.
- logical complement (negation): !
  - the complement of a condition had the value 1 (true) when the condition's value is 0 (false)
  - the complement of a condition has the value 0 (false) when the condition's value is nonzero (true)

Logical Operators

Expression	**Value**
`!5`	0
`!!5`	1
`!(6 < 7)`	0
`!6 < 7`	1
`!(3-4)`	0

Selection Statements in C

making decisions

The `if` Selection Statement

- Selection statements choose among alternative courses of action
  - If student's grade is greater than or equal to 60

    Print "Passed"
- If **true**, then "Passed" is printed
- If **false**, the printing is ignored
- Written in C as

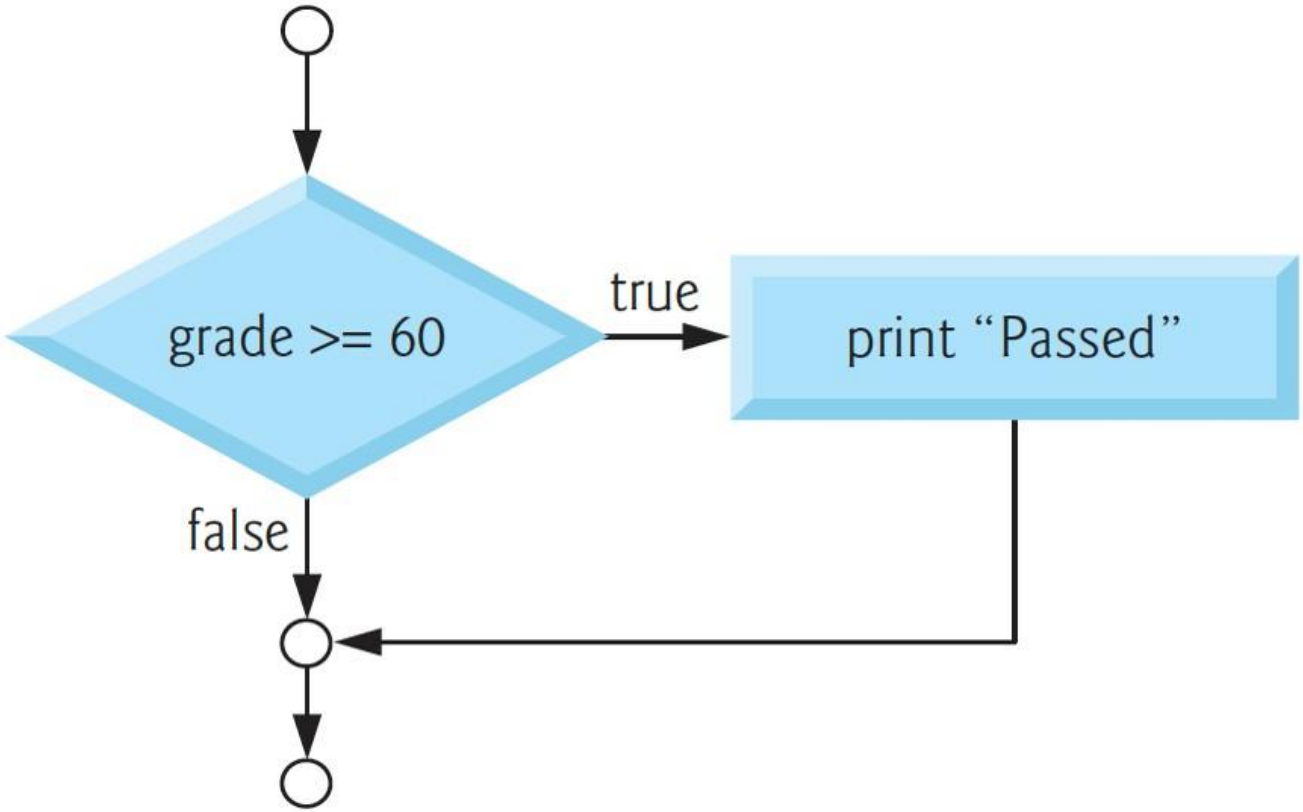
```
if (grade >= 60) { // grade must be declared before this
   printf("Passed");
} // end if
```

- The C if statement code corresponds closely to the pseudocode

The `if` Selection Statement

- **Indentation in the `if` Statement**
  - The indentation in the second line is optional but highly recommended
  - Emphasizes the inherent structure of structured programs
  - The compiler ignores white-space characters used for indentation and vertical spacing
    - blanks, tabs and newlines

The `if` Selection Statement



The `if…else` Selection Statement

- Specifies different actions to perform when the condition is **true** or **false**
  - If student's grade is greater than or equal to 60

    Print "Passed"

  else

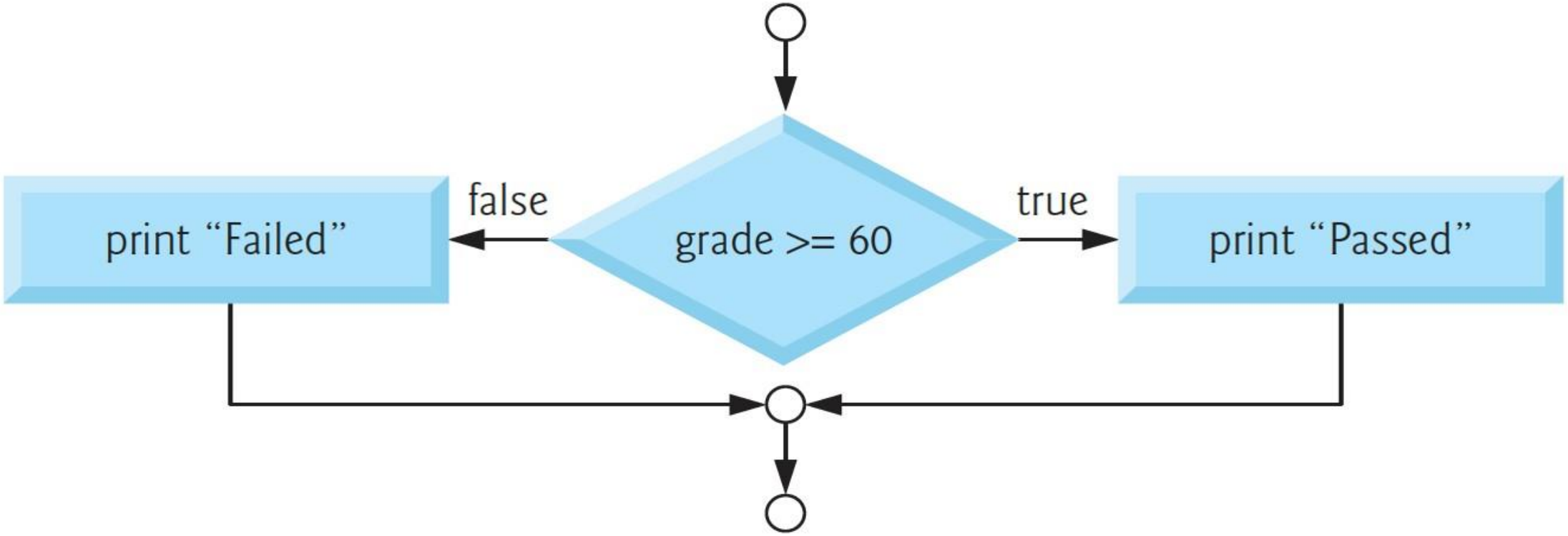
    Print "Failed"
- Prints "Passed" if the student's grade is greater than or equal to 60
- Otherwise, it prints "Failed"
- The else's body also is indented

The `if…else` Selection Statement

- Written in C as

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

The `if…else` Selection Statement



The `if…else` Selection Statement: Conditional Expressions

- The conditional operator (? :) is C's only ternary operator (3 operands)
- ? : and its three operands form a conditional expression
  - First operand is a condition
  - Second is the conditional expression's value if the condition is true
  - Third is the conditional expression's value if the condition is false

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

Nested `if…else` Statements

- Test for multiple cases by placing if…else statements inside if…else statements
- For example, the following pseudocode prints:
  - A for grades greater than or equal to 90
  - B for grades greater than or equal to 80 (but less than 90)
  - C for grades greater than or equal to 70 (but less than 80)
  - D for grades greater than or equal to 60 (but less than 70)
  - F for all other grades.

Nested `if…else` Statements

```
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"
```

Nested `if…else` Statements

- Written in C as

```
if (grade >= 90) {
   printf("A");
} // end if
else {
   if (grade >= 80) {
      printf("B");
   } // end if
   else {
      if (grade >= 70) {
         printf("C");
      } // end if
      else {
         if (grade >= 60) {
            printf("D");
         } // end if
         else {
            printf("F");
         } // end else
      } // end else
   } // end else
} // end else
```

Nested `if…else` Statements

- Most programmers write the preceding if statement as

```
if (grade >= 90) {
   printf("A");
} // end if
else if (grade >= 80) {
   printf("B");
} // end else if
else if (grade >= 70) {
   printf("C");
} // end else if
else if (grade >= 60) {
   printf("D");
} // end else if
else {
   printf("F");
} // end else
```

Blocks and Compound Statements

- To include several statements in an `if`, enclose them in braces ({ and })
- Braces form a **compound statement** or a **block**
  - Can be placed anywhere a single statement can be placed
- The following **`if…else`** statement's **`else`** part includes a compound statement containing two statements:

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

Blocks and Compound Statements

- The braces surrounding the two statements in the else clause are important; otherwise, the second statement would be outside the else's body and would execute regardless of whether the grade was less than 60
- Always include your control statements' bodies in braces

The `switch` Multiple-Selection Structure

- Occasionally, an algorithm will contain a series of decisions that test a variable or expression separately for each of the integer values it may assume, then perform different actions
- This is called multiple selection
- C provides the `switch` multiple-selection statement
- Consists of a series of `case` labels, an optional `default` case and statements to execute for each case

The `switch` Multiple-Selection Structure

```
switch ( a_variable ){
case value1:
      actions;
      break;
case value2 :
      actions;
      break;
...
default:
      actions;
      }
```

Exercise

- Write a simple calculator using the switch multiple-selection structure

Reference

- C How to Program, Ninth Edition by Deitel & Deitel, Pearson, 2022.