Lecture 7:
Control Statements

Ioan Raicu
Department of Electrical Engineering & Computer Science
Northwestern University

EECS 211
Fundamentals of Computer Programming II
April 7th, 2010

4.2 Algorithms

- Any solvable computing problem can be solved by the execution of a series of actions in a specific order.
- An algorithm is procedure for solving a problem in terms of
  - the actions to execute and
  - the order in which the actions execute
- Specifying the or-der in which statements (actions) execute in a computer program is called program control.
- This chapter investigates program control using C++'s control statements.

4.3 Pseudocode

- Pseudocode (or "fake" code) is an artificial and informal language that helps you develop algorithms.
- Similar to everyday English
- Convenient and user friendly.
- Helps you "think out" a program before attempting to write it.
- Carefully prepared pseudocode can easily be converted to a corresponding C++ program.
- Normally describes only executable statements.
- Declarations (that do not have initializers or do not involve constructor calls) are not executable statements.
- Fig. 4.1 corresponds to the algorithm that inputs two integers from the user, adds these integers and displays their sum.

4.3 Pseudocode

```
1  Prompt the user to enter the first integer
2  Input the first integer
3
4  Prompt the user to enter the second integer
5  Input the second integer
6
7  Add first integer and second integer, store result
8  Display result
```

Fig. 4.1 | Pseudocode for the addition program of Fig. 2.5.

4.4 Control Structures

- Normally, statements in a program execute one after the other in the order in which they're written.
  - Called sequential execution.
- Various C++ statements enable you to specify that the next statement to execute may be other than the next one in sequence.
  - Called transfer of control.
- All programs could be written in terms of only three control structures
  - the sequence structure
  - the selection structure and
  - the repetition structure
- When we introduce C++'s implementations of control structures, we'll refer to them in the terminology of the C++ standard document as "control statements."

4.4 Control Structures (cont.)

- Unless directed otherwise, the computer executes C++ statements one after the other in the order in which they're written—that is, in sequence.
- C++ allows us to have as many actions as we want in a sequence structure.
- Anywhere a single action may be placed, we may place several actions in sequence.

4.4 Control Structures (cont.)

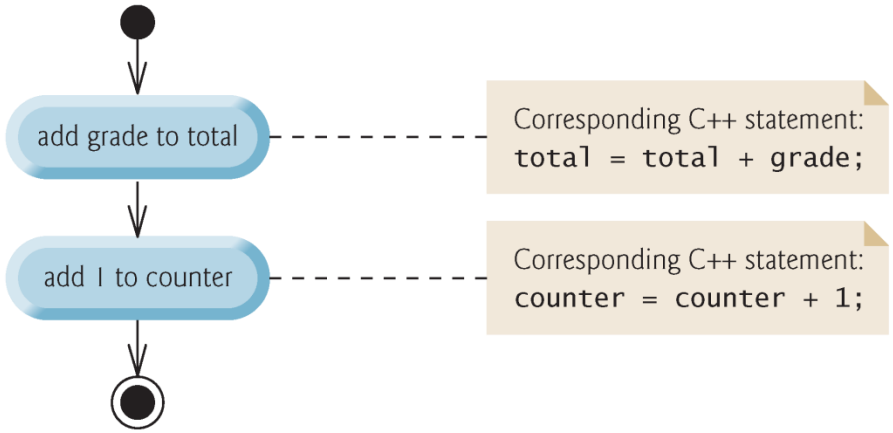

Fig. 4.2 | Sequence-structure activity diagram.

4.4 Control Structures (cont.)

- An activity diagram models the workflow (also called the activity) of a portion of a software system.
- Activity diagrams are composed of special-purpose symbols, such as action state symbols (a rectangle with its left and right sides replaced with arcs curving outward), diamonds and small circles; these symbols are connected by transition arrows, which represent the flow of the activity.
- Activity diagrams help you develop and represent algorithms, but many programmers prefer pseudocode.
- Activity diagrams clearly show how control structures operate.
- Action states represent actions to perform.
  - Each contains an action expression that specifies a particular action to perform.

4.4 Control Structures (cont.)

- The arrows in the activity diagram are called transition arrows.
  - Represent transitions, which indicate the order in which the actions represented by the action states occur.
- The solid circle at the top of the diagram represents the activity's initial- state—the beginning of the workflow before the program performs the modeled activities.
- The solid circle surrounded by a hollow circle that appears at the bottom of the activity diagram represents the final state—the end of the workflow after the program performs its activities.

4.4 Control Structures (cont.)

- C++ provides three types of selection statements
- The `if` selection statement either performs (selects) an action if a condition (predicate) is true or skips the action if the condition is false.
- The `if...else` selection statement performs an action if a condition is true or performs a different action if the condition is false.
- The `switch` selection statement performs one of many different actions, depending on the value of an integer expression.

4.4 Control Structures (cont.)

- The `if` selection statement is a single-selection statement because it selects or ignores a single action (or, as we'll soon see, a single group of actions).
- The `if`…`else` statement is called a double-selection statement because it selects between two different actions (or groups of actions).
- The `switch` selection statement is called a multiple-selection statement because it selects among many different actions (or groups of actions).

4.4 Control Structures (cont.)

- C++ provides three types of repetition statements (also called looping statements or loops) for performing statements repeatedly while a condition (called the loop-continuation condition) remains true.
- These are the while, do…while and for statements.
- The `while` and `for` statements perform the action (or group of actions) in their bodies zero or more times.
- The `do…while` statement performs the action (or group of actions) in its body at least once.

4.4 Control Structures (cont.)

- Each of the words `if`, `else`, `switch`, `while`, `do` and `for` is a C++ keyword.
- These words are reserved by the C++ programming language to implement various features, such as C++'s control statements.
- Keywords must not be used as identifiers, such as variable names.
- Figure 4.3 provides a complete list of C++ keywords-.

4.4 Control Structures (cont.)

C++ Keywords

Keywords common to the C and C++ programming languages

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

C++-only keywords

and	and_eq	asm	bitand	bitor
bool	catch	class	compl	const_cast
delete	dynamic_cast	explicit	export	false
friend	inline	mutable	namespace	new
not	not_eq	operator	or	or_eq
private	protected	public	reinterpret_cast	static_cast
template	this	throw	true	try
typeid	typename	using	virtual	wchar_t
xor	xor_eq			

Fig. 4.3 | C++ keywords.

4.4 Control Structures (cont.)

Common Programming Error 4.1

Using a keyword as an identifier is a syntax error.

4.4 Control Structures (cont.)

Common Programming Error 4.2

Spelling a keyword with any uppercase letters is a syntax error. All of C++'s keywords contain only lowercase letters.

4.4 Control Structures (cont.)

Software Engineering Observation 4.1

Any C++ program we'll ever build can be constructed from only seven different types of control statements (sequence, `if`, `if...else`, `switch`, `while`, `do...while` and `for`) combined in only two ways (control-statement stacking and control-statement nesting).

4.5 `if` Selection Statement

- Programs use selection statements to choose among alternative courses of action.
- The following pseudocode determines whether "student's grade is greater than or equal to 60" is `true` or `false`.

  – *If student's grade is greater than or equal to 60*
  Print "Passed"

  – If `true`, "Passed" is printed and the next pseudocode statement in order is "performed" (remember that pseudocode is not a real programming language).
  – If `false`, the print statement is ignored and the next pseudocode statement in order is performed.
  – The indentation of the second line is optional, but it's recommended because it emphasizes the inherent structure of structured programs.

4.5 if Selection Statement (cont.)

- The preceding pseudocode *If* statement can be written in C++ as
  - ```
    if ( grade >= 60 )
       cout << "Passed";
    ```
- Figure 4.4 illustrates the single-selection `if` statement.
- The diamond or decision symbol indicates that a decision is to be made.
  - The workflow will continue along a path determined by the symbol's associated guard conditions, which can be true or false.
  - Each transition arrow emerging from a decision symbol has a guard condition in square brackets above or next to the transition arrow.
  - If a guard condition is true, the workflow enters the action state to which that transition arrow points.
    ```

4.5 if Selection Statement (cont.)

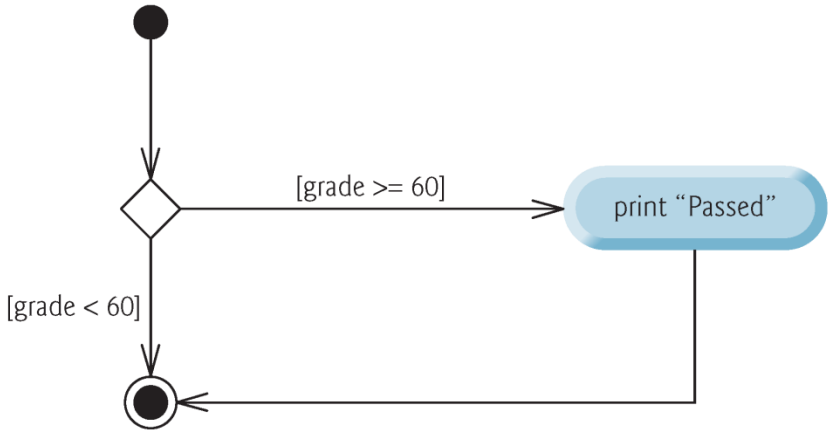

Fig. 4.4 | `if` single-selection statement activity diagram.

4.5 if Selection Statement (cont.)

- A decision can be based on any expression—if the expression evaluates to zero, it's treated as false; if the expression evaluates to nonzero, it's treated as true.
- C++ provides the data type bool for variables that can hold only the values true and false—each of these is a C++ keyword.

4.5 `if` Selection Statement (cont.)

Portability Tip 4.1

For compatibility with earlier versions of C, which used integers for Boolean values, the `bool` *value* `true` *also can be represented by any nonzero value (compilers typically use 1) and the* `bool` *value* `false` *also can be represented as the value zero.*

Questions