4

Control Statements Part 1

东南大学软件学院

1

OBJECTIVES

In this chapter you'll learn:

- **Basic problem-solving techniques.**
- **To develop algorithms through the process of top-down, stepwise refinement (自顶向下，逐步细化).**
- **To use the `if` and `if...else` selection （选择） statements to choose among alternative actions.**
- **To use the `while` repetition （循环） statement to execute statements in a program repeatedly.**
- **Counter-controlled repetition （计数器控制循环）and sentinel-controlled repetition （标志控制循环）.**
- **To use the increment （增量）, decrement（减量） and assignment （赋值）operators.**

2

4.1 Introduction

- **Before writing a program**
  - Have a thorough understanding of the problem
  - Carefully plan your approach for solving it
- **While writing a program**
  - Know what "building blocks" are available
  - Use good programming principles

3

4.2 Algorithms

- **Algorithms** 解决问题的步骤
  - Specify the actions to execute
  - Specify the order in which these actions execute
- **Program control** 程序控制
  - Specifies the order in which actions execute in a program
  - Are performed in C++ with control statements

4

4.3 Pseudocode 伪代码

- **Pseudocode**
  - Artificial, informal language used to develop algorithms
    - Used to "think out" a program before coding it
    - Easy to convert into a C++ program
  - Similar t
    - Only
    - No n
  - Not exe

```
Prompt the user to enter the first integer
Input the first integer

Prompt the user to enter the second integer
Input the second integer

Add first integer and second integer, store result
Display result
```

4.4 Control Structures

- **Sequential execution (Normally) 顺序执行**
  - Statements executed in sequential order
- **Transfer of control (Sometimes) 控制转移**
  - Next statement executed is *not* the next one in sequence
- **Structured programming 结构化编程**
  - Eliminated `goto` statements ("Spaghetti code")

4.4 Control Structures (Cont.)

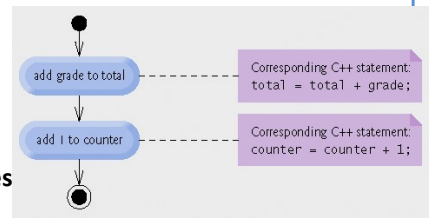
- Only three control structures are needed*
  - No `goto` statements
  - Demonstrated by Böhm and Jacopini
  - Three control structures
    - Sequence structure 顺序型结构
      - Programs executed sequentially by default
    - Selection structures 选择型结构
      - `if`, `if…else`, `switch`
    - Repetition structures 循环型结构
      - `while`, `do…while`, `for`

* Böhm and Jacopini, "Flow Diagrams, Turing Machinges and Languages with Only Two Formation Rules," Communications of the ACM, 1966

7

4.4 Control Structures (Cont.)

- UML activity diagram
  - Models the workflow
    - Action state symbols
      - Rectangles with curved sides
    - Small circles
      - Solid circle is the initial state
      - Solid circle in a hollow circle is the final state
    - Transition arrows
      - Represent the flow of activity
    - Comment notes
      - Connected to the diagram by dotted lines



8

4.4 Control Structures (Cont.)

- Single-entry/single-exit control statements 单入口/单出口控制语句（便于程序构建）
  - Three types of control statements
    - Sequence statement C++内置的
    - Selection statements `if`, `if...else`, `switch`
    - Repetition statements `while`, `do...while`, `for`
  - Combined in one of two ways
    - Control statement stacking 堆栈式
      - Connects exit point of one to entry point of the next
    - Control statement nesting 嵌套式

9

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).

10

C++ Keywords C++关键词不能用于标识符

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.

11

4.5 `if` Selection Statement

- **Selection statements**
  - **Choose among alternative courses of action**
  - **Pseudocode example**
    - ***If student's grade is greater than or equal to 60 Print "Passed"***
      - **If the condition is `true`**
        - » **Print statement executes, program continues to next statement**
      - **If the condition is `false`**
        - » **Print statement ignored, program continues**

12

4.5 if Selection Statement (Cont.)

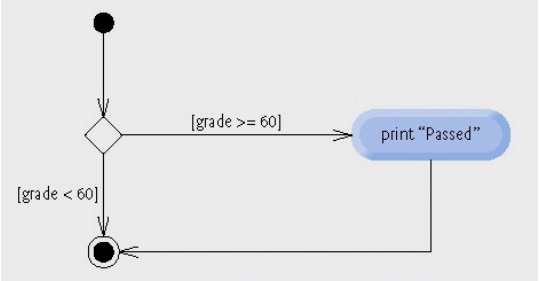
- Selection statements (Cont.)
  - Translation into C++
    - `if ( grade >= 60 )`
      `cout << "Passed";`
  - Any expression can be used as the condition
    - If it evaluates to zero, it is treated as false
    - If it evaluates to non-zero, it is treated as true

```
if ( 1 )
    cout << "Passed";
if ( 0 )
    cout << "Passed";
```

13

4.5 if Selection Statement (Cont.)

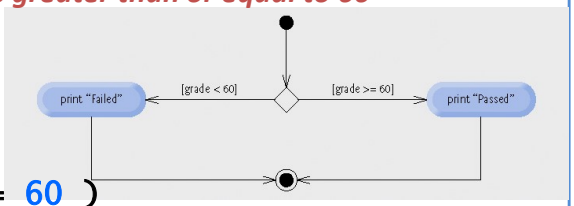
- Diamond symbol in UML modeling
  - Indicates decision is to be made
  - Contains guard conditions
    - Test condition
    - Follow correct path



14

4.6 `if…else` Double-Selection Statement

- `if…else`
  - Performs one action if the condition is true, and a different action if the condition is false
- Pseudocode
  - *If student's grade is greater than or equal to 60*
    print "Passed"
    Else
    print "Failed"



- C++ code

```
if ( grade >= 60 )
   cout << "Passed";
else
   cout << "Failed";
```

15

4.6 `if…else` Double-Selection Statement (Cont.)

- Ternary conditional operator (`?:`)
  - Three arguments (condition, value if `true`, value if `false`)
- Code could be written:

```
cout << ( grade >= 60 ? "Passed" : "Failed" );
```

Condition Value if true Value if false

```
grade >= 60 ? cout << "Passed" : cout << "Failed" ;
```

16

4.6 `if…else` Double-Selection Statement (Cont.)

- **Nested `if…else` statements**
  - One inside another, test for multiple cases
  - Once a condition met, other statements are skipped
  - Example
    - *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"

17

4.6 `if…else` Double-Selection Statement (Cont.)

- **Nested `if…else` statements (Cont.)**
  - Written In C++

```
if ( studentGrade >= 90 )
    cout << "A";
else
    if (studentGrade >= 80 )
        cout << "B";
    else
        if (studentGrade >= 70 )
            cout << "C";
        else
            if ( studentGrade >= 60 )
                cout << "D";
            else
                cout << "F";
```

18

4.6 if…else Double-Selection Statement (Cont.)

- Nested if…else statements (Cont.)
  - Written In C++ (indented differently)

```
if ( studentGrade >= 90 )
    cout << "A";
else if (studentGrade >= 80 )
    cout << "B";
else if (studentGrade >= 70 )
    cout << "C";
else if ( studentGrade >= 60 )
    cout << "D";
else
    cout << "F";
```

19

Performance Tip 4.1

- A nested if...else statement can perform much faster than a series of single-selection if statements because of the possibility of early exit after one of the conditions is satisfied.
- 嵌套型的if…else语句会比一系列单独的if…else语句效率要高。

20

Performance Tip 4.2

•In a nested `if... else` statement, test the conditions that are more likely to be `true` at the beginning of the nested `if...else` statement. This will enable the nested `if...else` statement to run faster and exit earlier than testing infrequently occurring cases first.

•将可能性最大的情况放在嵌套if…else语句的最外层，可以提高程序执行的效率。

21

4.6 `if…else` Double-Selection Statement (Cont.)

- Dangling-`else` problem （else 摇摆问题）
  - Example

```
• if ( x > 5 )
      if ( y > 5 )
         cout << "x and y are > 5";
  else
      cout << "x is <= 5";
```

```
• if ( x > 5 )
      if ( y > 5 )
         cout << "x and y are > 5";
      else
         cout << "x is <= 5";
```

  - Compiler associates `else` with the immediately preceding `if` 与最近的if语句相匹配

22

4.6 if…else Double-Selection Statement (Cont.)

- Dangling-else problem (Cont.)
  - Rewrite with braces ({})

```
if ( x > 5 )
{
    if ( y > 5 )
        cout << "x and y are > 5";
}
else
    cout << "x is <= 5";
```

23

4.6 if…else Double-Selection Statement (Cont.)

- Compound statement 复合语句
  - Also called a block 语句块
    - Set of statements within a pair of braces
    - Used to include multiple statements in an if body
  - Example

```
if ( studentGrade >= 60 )
    cout << “Passed.\n”;
else
{
    cout << “Failed.\n”;
    cout << “You must take this course again.\n”;
}
```

  - Without braces,

```
cout << "You must take this course again.\n";
```

always executes

24

Software Engineering Observation 4.2

- A block can be placed anywhere in a program that a single statement can be placed.
- 任何单条语句可以出现的地方都可以用语句块替代。

25

4.6 `if…else` Double-Selection Statement (Cont.)

- Empty statement 空语句
  - A semicolon (`;`) where a statement would normally be
  - Performs no action
  - Also called a null statement

26

Common Programming Error

•Placing a semicolon after the condition in an `if` statement leads to a logic error in single-selection `if` statements and a syntax error in double-selection `if...else` statements (when the `if` part contains an actual body statement).

```
if (score >= 60);
    cout<<"Pass!";
```

27

4.7 while Repetition Statement

- Repetition statement 循环语句
  - Action repeated while some condition remains true
  - `while` (condition){
    ... (循环体)
    }
  - Example 输出小于等于100所有的3^n (n>=0)

```
int product = 1;
while ( product <= 100 ){
    cout<<product<<"\t";
    product = 3 * product;
}
```

28

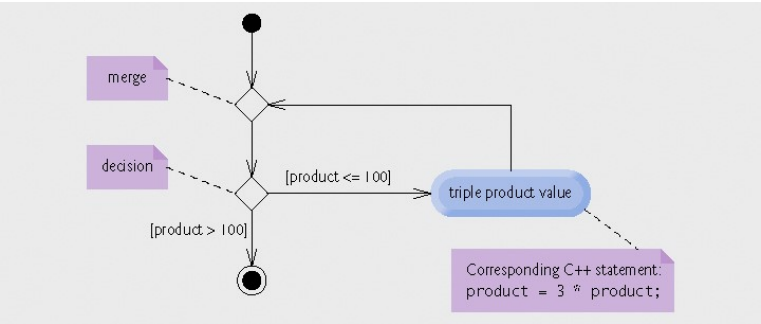
Common Programming Error 4.3

•Not providing, in the body of a `while` statement, an action that eventually causes the condition in the `while` to become false normally results in a logic error called an **infinite loop**, in which the repetition statement never terminates. 注意要使得循环条件能够变为false，否则将会是无限循环。

29

4.7 `while` Repetition Statement (Cont.)

- UML merge symbol
  - Joins two or more flows of activity into one flow of activity
  - Represented as a diamond



30

Performance Tip 4.3

- A small performance improvement for code that executes many times in a loop can result in substantial overall performance improvement. 即便是微小的性能改进，也有可能带来循环语句块整体性能的提升。

31

4.8 Formulating Algorithms: Counter-Controlled Repetition（计数器控制的循环）

- Problem statement

  A class of ten students took a quiz. The grades (integers in the range 0 to 100) for this quiz are available to you. Calculate and display the total of all student grades and the class average on the quiz.

- Counter-controlled repetition
  - Loop repeated until counter reaches certain value
  - Also known as definite repetition 定数循环
    - Number of repetitions is known beforehand

32

4.8 Formulating Algorithms: Counter-Controlled Repetition

- **Top-down, stepwise refinement 自顶向下，逐步求精**
  - Development technique for well-structured programs
  - Top step
    - Single statement conveying overall function of the program
    - *Determine the class average for the quiz*
  - First refinement
    - Multiple statements using only the sequence structure
    - Example
      - *Initialize variables*
      - *Input, sum and count the quiz grades*
      - *Calculate and print the total of all student grades and the class average*

33

Software Engineering Observation 4.5

- Many programs can be divided logically into three phases: an initialization phase that initializes the program variables; a processing phase that inputs data values and adjusts program variables (such as counters and totals) accordingly; and a termination phase that calculates and outputs the final results.

34

1 *Set total to zero*
2 *Set grade counter to one*

初始化

3
4 *While grade counter is less than or equal to ten*
5 *Prompt the user to enter the next grade*
6 *Input the next grade*
7 *Add the grade into the total*
8 *Add one to the grade counter*
9

处理阶段

10 *Set the class average to the total divided by ten*
11 *Print the total of the grades for all students in the class*
12 Print the class average

结束

35

4.8 Formulating Algorithms: Counter-Controlled Repetition (Cont.)

- Counter-controlled repetition
  - Counter variable 计数器变量
    - Used to count
      - In example, indicates which of the 10 grades is being entered
  - Total variable 总和变量
    - Used to accumulate the sum of several values
    - Normally initialized to zero beforehand
      - Otherwise it would include the previous value stored in that memory location

36

```
int main()
{
    int total = 0;
    int gradeCounter = 1;
    int grade; // grade value entered by user
    int average; // average of grades
    while ( gradeCounter <= 10 )
    {
      cout << "Enter grade: ";
      cin >> grade;
      total = total + grade;
      gradeCounter = gradeCounter + 1;
    }
    average = total / 10;
    cout << "\nTotal of all 10 grades is " << total << endl;
    cout << "Class average is " << average << endl;
}
```

初始化

处理阶段

结束

37

```
int main()
{
    int total = 0;
    int gradeCounter = 1;
    int grade; // grade value entered by user
    int average; // average of grades
    while ( gradeCounter <= 10 )
    {
      cout << "Enter grade: ";
      cin >> grade;
      total = total + grade;
      gradeCounter = gradeCounter + 1;
    }
    average = total / 10;
    cout << "\nTotal of all 10 grades is " << total << endl;
    cout << "Class average is " << average << endl;
    return 0;
}
```

total和gradeCounter为什么需要初始化？而其他两个变量不需要？能否省略total和gradeCounter变量初始化步骤？

总和变量

计数器变量

Uninitialized variables Contain "garbage" (or undefined) values

改变计数器数值

38

```
Welcome to the grade book for
CS101 C++ Programming
Enter grade: 67
Enter grade: 78
Enter grade: 89
Enter grade: 67
Enter grade: 87
Enter grade: 98
Enter grade: 93
Enter grade: 85
Enter grade: 82
Enter grade: 100
Total of all 10 grades is 846
Class average is 84
```

为什么不是84.6?

average = total / 10;

int 数据类型

➢ 整型除法

When dividing two integers performs truncation (截尾)

846 / 10 = 84 ~~.6~~

39

Common Programming Error 4.6

•Using a loop's counter-control variable in a calculation after the loop often causes a common logic error called an off-by-one-error (差1错误).

40

4.9 Formulating Algorithms: Sentinel-Controlled Repetition （标记控制循环）

- **Problem statement**

  Develop a class average program that processes grades for an arbitrary number （任意数目） of students each time it is run.

- **Sentinel-controlled repetition 标记量控制循环**
  - **Also known as indefinite repetition 不定数循环**
  - **Use a sentinel value 使用标记量**
    - **Indicates "end of data entry"**
    - **A sentinel value cannot also be a valid input value （不能是一个合法的输入值）**

41

4.9 Formulating Algorithms: Sentinel-Controlled Repetition (Cont.)

- **Top-down, stepwise refinement 自顶向下，逐步求精**
  - **Development technique for well-structured programs**
  - **Top step**
    - **Single statement conveying overall function of the program**
    - ***Determine the class average for the quiz***
  - **First refinement**
    - **Multiple statements using only the sequence structure**
    - **Example**
      - ***Initialize variables***
      - ***Input, sum and count the quiz grades***
      - ***Calculate and print the total of all student grades and the class average***

42

Initialize total to zero
Initialize counter to zero

初始化

Prompt the user to enter the first grade
Input the first grade (possibly the sentinel)

处理阶段

While the user has not yet ente
Add this grade into the run
Add one to the grade count
Prompt the user to enter th
Input the next grade (possibly t

Division by zero is normally a fatal logic error

If the counter is not equal to zero
Set the average to the total divided by the counter
Print the total of the grades for all students in the class
Print the class average
else
Print "No grades were entered"

结束

43

4.9 Formulating Algorithms: Sentinel-Controlled Repetition (Cont.)

- Floating-point numbers 浮点数
  - A real number with a decimal point
  - C++ provides data types `float` and `double`
    - `double` numbers can have larger magnitude and finer detail
    - Floating-point constant values are treated as `double` values by default
  - Floating-point values are often irrational numbers（1 / 3, 1 / 9, pi）

Float 只能保证有6位有效数字，double至少可以保证有10位有效数字

44

基本数据类型取值范围

类型	大小/字节	值
bool	1	true 或者false
unsigned short int	2	0~65535
short int	2	-32768~32767
unsigned long int	4	0~4294967395
long int	4	-2147483648~2147483647
int	4	-2147483648~2147483647
unsigned int	4	0~4294967395
char	1	256 个字符
float	4	10^{-37}~10^{38}
double	8	10^{-307}~10^{308}
long double	16	10^{-4931}~10^{4932}

45

```
int main()
  {
    int total;
    int gradeCounter;
    int grade;
    double average;
    // initialization phase
    total = 0; // initialize total
    gradeCounter = 0; // initialize loop counter
    // processing phase
    cout << "Enter grade or -1 to quit: ";
    cin >> grade;
    while ( grade != -1 )
    {
      total = total + grade;
      gradeCounter = gradeCounter + 1;
      cout << "Enter grade or -1 to quit: ";
      cin >> grade; // input grade or sentinel value
    } // end while
```

46

```
// termination phase
  if ( gradeCounter != 0 ) // if user entered at least one grade...
  {
    // calculate average of all grades entered
    average = static_cast< double >( total ) / gradeCounter;

    // display total and average (with two digits of precision)
    cout << "\nTotal of all " << gradeCounter << " grades entered is "
      << total << endl;
    cout << "Class average is " << setprecision( 2 ) << fixed <<
average
      << endl;
  } // end if
  else // no grades were entered, so output appropriate message
    cout << "No grades were entered" << endl;
  return 0;
}
```

```
Welcome to the grade book for
CS101 C++ Programming
Enter grade or -1 to quit: 97
Enter grade or -1 to quit: 88
Enter grade or -1 to quit: 72
Enter grade or -1 to quit: -1
Total of all 3 grades entered is 257
Class average is 85.67
```

47

4.9 基本数据类型之间的显式及隐式转换

- Explicit conversion 显式转换
  - Unary cast operator 一元强制类型转换运算符
    - Creates a temporary copy of its operand with a different data type
      - `static_cast< double > ( total )`
        - » Creates temporary floating-point copy of `total`
- Implicit conversion 隐式转换
  - Promotion 升级(原则是不损失数据)
    - Converting a value (e.g. `int`) to another data type (e.g. `double`) to perform a calculation（int的取值范围比double的取值范围要小）

48

4.9 基本数据类型之间的显式及隐式转换

```
average = static_cast< double >( total ) /gradeCounter;
```

double　　显式转换成double　　隐式转换成double

(1)
```
int a = 5, b = 10;
double c;
c = a / static_cast<double>(b);
```

(2)
```
int a = 5, b = 10;
double c;
c = a / b;
```

(3)
```
int b = 10;
double a = 5, c;
c = a / b;
```

(4)
```
double a = 5, b = 10;
int c;
c = a / b;
```

warning C4244: “=”: 从“double”转换到“int”，可能丢失数据

(5)
```
double a = 5, b = 10, c;
c = static_cast<int>(a) / b;
```

Answers:
(1) 0.5
(2) 0
(3) 0.5
(4) 0
(5) 0.5

4.9 基本数据类型之间的显式及隐式转换

(1)
```
double c, a = 2.2;
int d = 2, f = 10;
c = a * d / f;
```

(2)
```
double c, a = 2.2;
int d = 2, f = 10;
c = a + d / f;
```

输出：
(1)　0.44
(2)　2.2

4.9 浮点数的格式化

- **Formatting floating-point numbers**
  - Parameterized stream manipulator `setprecision`
    - Specifies number of digits of precision to display
    - Default precision is six digits(不使用setprecision)
    - Need to include header filer <iomanip>
  - Nonparameterized stream manipulator `fixed`
    - Indicates that floating-point values should be output in fixed-point format 以定点数形式输出浮点数
      - 不采用科学计数法形式，如3.1×10^3
  - Nonparameterized stream manipulator `showpoint`
    - Forces decimal point to display 强制显示小数点
    - 相反的有`noshowpoint`

51

Setprecision的使用

- 在用fixed流运算符进行定点表示的输出中，setprecision(n)表示小数位数。

```
double a = 1000, b = 9.1256;
cout<<"a = "<< setprecision(2)<<fixed<<a<<"\n";
cout<<"b = "<<b<<"\n";
cout<<"a = "<< setprecision(5)<<a<<"\n";
cout<<"b = "<<b<<"\n";
```

Output:
a = 1000.00
b = 9.13
a = 1000.00000
b = 9.12560

setprecision和fixed流运算符：
- 粘性流运算符

fixed流运算符：
- 会进行四舍五入
- 小数点位数不够时会自动进行补0

52

Setprecision的使用

- 在用showpoint流运算符进行浮点表示的输出中，setprecision(n)表示有效位数。

```
double a = 1000, b = 9.1256;
cout<<"a = "<< setprecision(3)<<showpoint<<a<<"\n";
cout<<"b = "<<b<<"\n";
cout<<"a = "<< setprecision(7)<<a<<"\n";
cout<<"b = "<<b<<"\n";
cout<<"a = "<< setprecision(4)<<a<<"\n";
cout<<"a = "<<noshowpoint<<a<<"\n";
```

Output:

```
a = 1.00e+003
b = 9.13
a = 1000.000
b = 9.125600
a = 1000.
a = 1000
```

showpoint流运算符：
- 粘性流运算符
- 会进行四舍五入
- 小数点位数不够时会自动进行补0

53

```
int main()
{
        double a = 1000.1256, b = 9.1256;
        cout<<"a = "<< setprecision(2)<<fixed<<a<<"\n";
        cout<<"b = "<<b<<"\n";
        cout<<"a = "<< setprecision(2)<<showpoint<<a<<"\n";
        cout<<"b = "<<b<<"\n";
}
```

```
Output:
a = 1000.13
b = 9.13
a = 1.0e+003
b = 9.1
```

```
cout.unsetf(ios_base::fixed);
```

```
Output:
a = 1000.13
b = 9.13
a = 1000.13
b = 9.13
```

Fixed流运算符会覆盖showpoint流运算符的设置，反之不会

54

Common Programming Error 4.9

Floating-point numbers are represented only approximately by most computers. Using floating-point numbers in a manner that assumes they are represented exactly (e.g., using them in comparisons for equality) can lead to incorrect results.

问题：在C++中该如何比较两个浮点数是否相等？

```
float  a = 5, b = 5;
if (fabs(a-b) < 1E-5)
  cout << "a is equal to b.";
```

55

4.10 Formulating Algorithms: Nested Control Statement

- **Problem statement**

You have been asked to write a program to summarize the results. You have been given a list of these 10 students. Next to each name is written a 1 if the student passed the exam or a 2 if the student failed.

Your program should analyze the results of the exam as follows:

1. Input each test result (i.e., a 1 or a 2). Display the prompting message "Enter result" each time the program requests another test result.

2. Count the number of test results of each type.

3. Display a summary of the test results indicating the number of students who passed and the number who failed.

56

4.10 Formulating Algorithms: Nested Control Statement (Cont.)

- **Notice that**
  - **Program processes 10 results**
    - **Fixed number, use counter-controlled loop**
  - **Each test result is 1 or 2**
    - **If not 1, assume 2**
  - **Two counters can be used**
    - **One counts number that passed**
    - **Another counts number that failed**

57

4.10 Formulating Algorithms: Nested Control Statement (Cont.)

- **Top level outline**
  - *Analyze exam results*
- **First refinement**
  - *Initialize variables*
    Input the 10 exam results and count passes and failures
    Print a summary of the exam results
- **Second Refinement**
  - *Initialize variables*
    to
    Initialize passes to zero
    Initialize failures to zero
    Initialize student counter to one

58

4.10 Formulating Algorithms: Nested Control Statement (Cont.)

- **Second Refinement (Cont.)**
  - *Input the ten exam results and count passes and failures*

    to

    While student counter is less than or equal to 10
    Prompt the user to enter the next exam result
    If the student passed
    Add one to passes
    Else
    Add one to failures
    Add one to student counter

59

4.10 Formulating Algorithms: Nested Control Statement (Cont.)

- **Second Refinement (Cont.)**
  - *Print a summary of the exam results*

    to

    Print the number of passes
    Print the number of failures
    If more than eight students passed
    Print "Raise tuition"

60

```
int main()
{
  int passes = 0;
  int failures = 0;
  int studentCounter = 1;
  int result; // one exam result (1 = pass, 2 = fail)
  while ( studentCounter <= 10 )
  {
    cout << "Enter result (1 = pass, 2 = fail): ";
    cin >> result;
    if ( result == 1 )
      passes = passes + 1;
    else
      failures = failures + 1;
    studentCounter = studentCounter + 1;
  } // end while
  cout << "Passed " << passes << "\nFailed " << failures << endl;

  return 0;
}
```

循环语句

选择语句

61

4.11 Assignment Operators

- Assignment expression abbreviations
  - Addition assignment operator
    - Example

      `c = c + 3;` abbreviates to `c += 3;`
- Other assignment operators

```
d -= 4      (d = d - 4)
e *= 5      (e = e * 5)
f /= 3      (f = f / 3)
g %= 9      (g = g % 9)
```

62

4.12 Increment and Decrement Operators

- **Increment operator ++**
  - Increments variable by one
    - Example

      C++
- **Decrement operator −−**
  - Decrements variable by one
    - Example

      C−−

63

4.12 Increment and Decrement Operators

- **Preincrement/ Predecrement 前自增/前自减**
  - When the operator is used before the variable (++V or −−V)
  - Variable is changed, then the expression it is in is evaluated using the new value
- **Postincrement/ Postdecrement 后自增/后自减**
  - When the operator is used after the variable (V++ or V−−)
  - Expression the variable is in executes using the old value, then the variable is changed

运算符和变量之间没有空格

64

4.12 Increment and Decrement Operators (Cont.)

- If `c = 5`, then

  `cout << ++c;`
  - `c` is incremented to 6
  - Then 6 is printed

  `cout << c++;`
  - Prints 5 (printing occurs before the increment)
  - Then `c` is incremented to 6

65

4.12 Increment and Decrement Operators (Cont.)

- When variable is not in an expression
  - Preincrementing and postincrementing have same effect
    - Example

      ```
      ++c;
      cout << c;
      ```

      and

      ```
      c++;
      cout << c;
      ```
    - print the same result

66

Common Programming Error 4.14

Attempting to use the increment or decrement operator on an expression other than a modifiable variable name or reference, e.g., writing ++(x + 1), is a syntax error.

67

Operators	Associativity	Type
`::`	left to right	scope resolution
`()`	left to right	parentheses
`++` `--` `static_cast<` *type* `>()`	left to right	unary (postfix)
`++` `--` `+` `-`	right to left	unary (prefix)
`*` `/` `%`	left to right	multiplicative
`+` `-`	left to right	additive
`<<` `>>`	left to right	insertion/extraction
`<` `<=` `>` `>=`	left to right	relational
`==` `!=`	left to right	equality
`?:`	right to left	conditional
`=` `+=` `-=` `*=` `/=` `%=`	right to left	assignment

Fig. 4.22 | Operator precedence for the operators encountered so far in the text.

68