An Introduction to Programming with C++

Fifth Edition

CHAPTER 3

COMPLETING THE PROBLEM-SOLVING PROCESS AND GETTING STARTED WITH C++

Objectives

- Code an algorithm into a program
- Desk-check a program
- Evaluate and modify a program
- Understand the components of a C++ program
- Create a C++ program

Objectives (continued)

- Save, build, and execute a C++ program
- Locate and fix an error in a C++ program
- Print a C++ program
- Make a backup copy of a solution

Concept Lesson

- More on the Problem-Solving Process
- Coding the Algorithm into a Program
- Desk-Checking the Program
- Evaluating and Modifying the Program
- Creating a C++ Program

More on the Problem-Solving Process

1. Analyze the problem.
2. Plan the algorithm.
3. Desk-check the algorithm.
4. Code the algorithm into a program.
5. Desk-check the program.
6. Evaluate and modify (if necessary) the program.

Figure 3-1: The process for creating a computer program

Coding the Algorithm into a Program

Sarah Martin has been working for Quality Builders for four years. Last year, Sarah received a 4% raise, which brought her current weekly pay to $250. Sarah is scheduled to receive a 3% raise next week. She wants you to write a program that will display, on the computer screen, the amount of her new weekly pay.

Figure 3-2: Problem specification for Sarah Martin

Input	Processing	Output
current weekly pay raise rate	Processing items: weekly raise Algorithm: 1. enter the current weekly pay and raise rate 2. calculate the weekly raise by multiplying the current weekly pay by the raise rate 3. calculate the new weekly pay by adding the weekly raise to the current weekly pay 4. display the new weekly pay	new weekly pay

Figure 3-3: IPO chart for Sarah Martin's problem

Assigning Names, Data Types, and Initial Values to the IPO Items

- To code algorithm, first assign a name to each input, processing, and output item in IPO chart
  - Names can contain only letters, numbers, and _
  - Cannot contain punctuation characters or spaces
  - Examples:
    - `raise` ← usually in lowercase letters
    - `newPay` ← use **camel case** if name contains multiple words
  - Each input/processing/output item must have a data type
  - You may **initialize** each item

Assigning Names, Data Types, and Initial Values to the IPO Items (continued)

double is a **keyword**

IPO chart information	C++ instructions
Input current weekly pay raise rate	`double currentPay = 0.0;` `double raiseRate = 0.0;`
Processing weekly raise	`double raise = 0.0;`
Output new weekly pay	`double newPay = 0.0;`
Algorithm 1. enter the current weekly pay and raise rate 2. calculate the weekly raise by multiplying the current weekly pay by the raise rate 3. calculate the new weekly pay by adding the weekly raise to the current weekly pay 4. display the new weekly pay	

this is a **statement**

all C++ statements must end with a semicolon

Figure 3-4: C++ instructions corresponding to the input, processing, and output items

Translating the Algorithm Steps into C++ Code

IPO chart information	C++ instructions
Input current weekly pay raise rate	`double currentPay = 0.0;` `double raiseRate = 0.0;`
Processing weekly raise	`double raise = 0.0;`
Output new weekly pay	`double newPay = 0.0;`

Algorithm
1. enter the current weekly pay and raise rate
	`cout << "Enter current weekly pay: ";`
	`cin >> currentPay;`
	`cout << "Enter raise rate: ";`
	`cin >> raiseRate;`

2. calculate the weekly raise by multiplying the current weekly pay by the raise rate	`raise = currentPay * raiseRate;`
3. calculate the new weekly pay by adding the weekly raise to the current weekly pay	`newPay = raise + currentPay;`
4. display the new weekly pay	`cout << "New pay: " << newPay << endl;`

`cout`: standard output stream
`cin`: standard input stream
<<: **insertion operator**
>>: **extraction operator**

stream: sequence of characters, to perform standard I/O operations

a **stream manipulator**

Figure 3-5: C++ instructions for the Sarah Martin algorithm

Desk-Checking the Program

current weekly pay	raise rate	weekly raise	new weekly pay
~~250~~	~~.03~~	~~7.50~~	~~257.50~~
100	.10	10	110

Figure 3-6: Completed desk-check table for the Sarah Martin algorithm

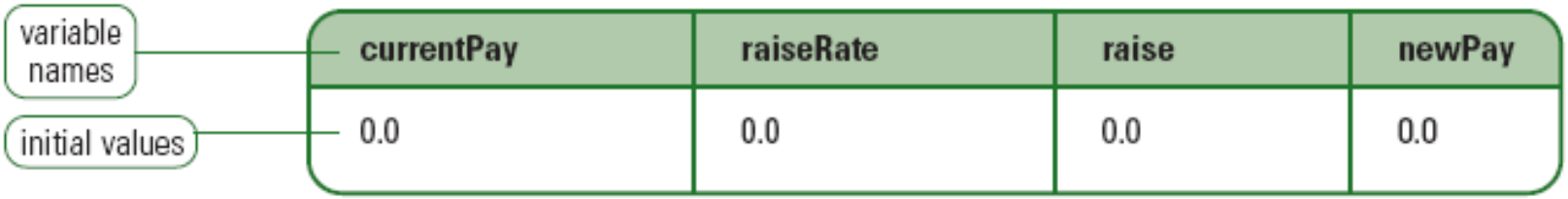


Figure 3-7: Variable names and initial values shown in the program's desk-check table

Desk-Checking the Program (continued)

currentPay	raiseRate	raise	newPay
~~0.0~~ 250.0	~~0.0~~ .03	0.0	0.0

Figure 3-8: Current status of the desk-check table

currentPay	raiseRate	raise	newPay
~~0.0~~ 250.0	~~0.0~~ .03	~~0.0~~ 7.5	0.0

Figure 3-9: Desk-check table showing the results of the raise calculation

Desk-Checking the Program (continued)

currentPay	raiseRate	raise	newPay
~~0.0~~	~~0.0~~	~~0.0~~	~~0.0~~
250.0	.03	7.5	257.5

Figure 3-10: Desk-check table showing the results of the new weekly pay calculation

currentPay	raiseRate	raise	newPay
~~0.0~~	~~0.0~~	~~0.0~~	~~0.0~~
~~250.0~~	~~.03~~	~~7.5~~	~~257.5~~
~~0.0~~	~~0.0~~	~~0.0~~	~~0.0~~
100.0	.10	10.0	110.0

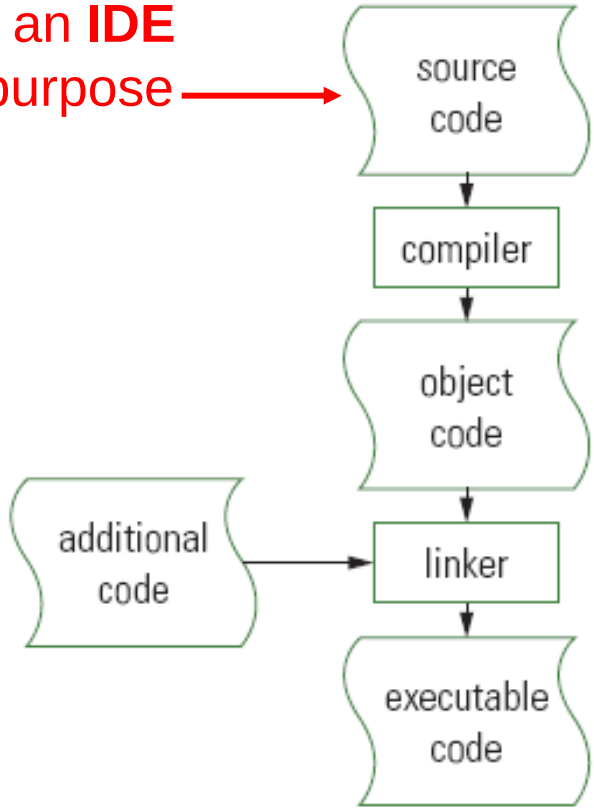
Figure 3-11: Desk-check table showing the results of the second desk-check

Evaluating and Modifying the Program

- **Testing** is running the program, with sample data
  - Results should agree with desk-check ones
- **Debugging** is locating/removing errors in program
  - Program errors are called **bugs**
- A **syntax error** occurs if an instruction violates the programming language's **syntax** (set of rules)
  - E.g., `cout < "Hello";`
- A **logic error** occurs if an instruction does not give the expected results
  - E.g., `average = number1 + number2 / 2;`

Creating a C++ Program

created using an **IDE**
or a general-purpose
editor



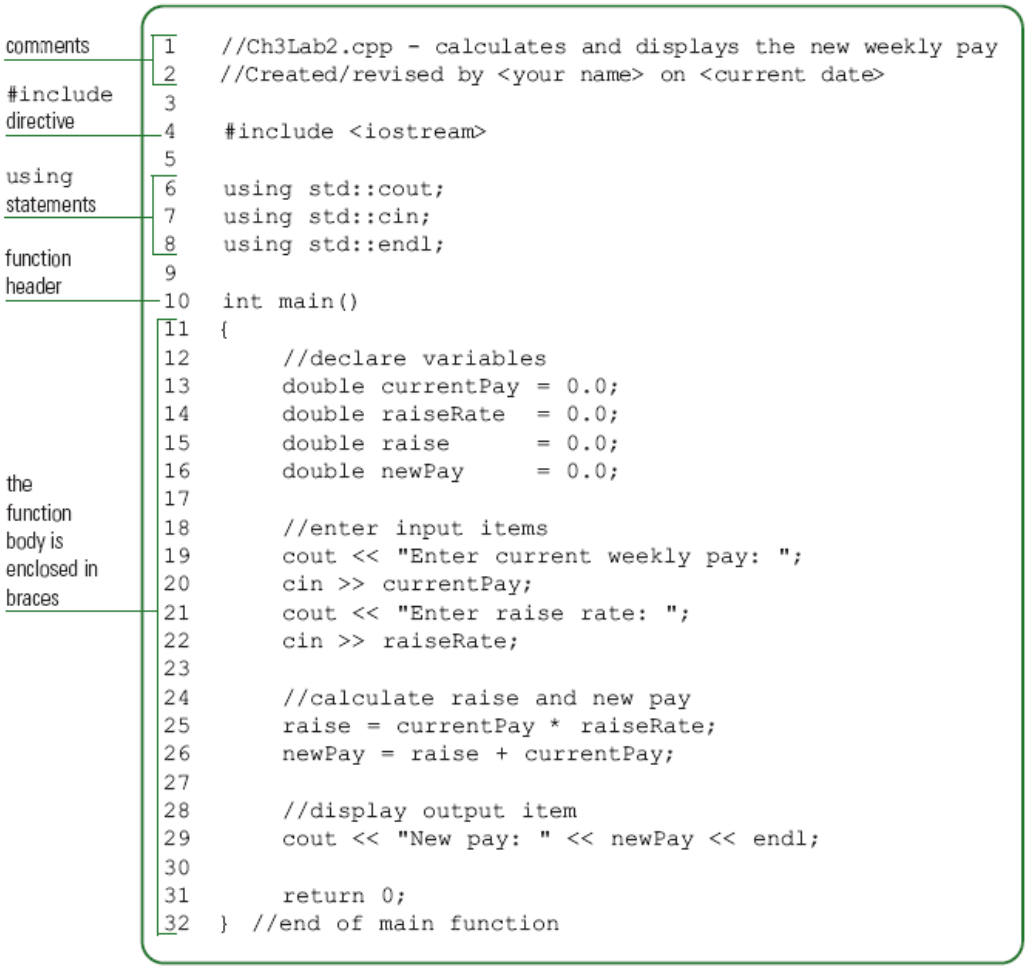
source file: Ch3Lab2.cpp

object file: Ch3Lab2.obj

executable file: Ch3Lab2.exe

Figure 3-12: Process by which source code is translated into executable code

Creating a C++ Program (continued)




```
1   //Ch3Lab2.cpp - calculates and displays the new weekly pay
2   //Created/revised by <your name> on <current date>
3
4   #include <iostream>
5
6   using std::cout;
7   using std::cin;
8   using std::endl;
9
10  int main()
11  {
12       //declare variables
13       double currentPay = 0.0;
14       double raiseRate  = 0.0;
15       double raise      = 0.0;
16       double newPay     = 0.0;
17
18       //enter input items
19       cout << "Enter current weekly pay: ";
20       cin >> currentPay;
21       cout << "Enter raise rate: ";
22       cin >> raiseRate;
23
24       //calculate raise and new pay
25       raise = currentPay * raiseRate;
26       newPay = raise + currentPay;
27
28       //display output item
29       cout << "New pay: " << newPay << endl;
30
31       return 0;
32  } //end of main function
```

Figure 3-13: C++ program designed to solve the Sarah Martin problem

Summary

- Fourth step in the problem-solving process is to code the algorithm into a program
- In C++, you perform standard I/O operations using streams (sequences of characters)
  - `cout` and `cin`
  - Insertion operator (<<) sends data to output stream
  - Extraction operator (>>) gets data from input stream
- After coding the algorithm, you desk-check program
- Final step in the problem-solving process is to evaluate and modify (if necessary) the program

Summary (continued)

- Some programs have errors, called bugs
  - Syntax error occurs when an instruction violates one of the rules of the programming language's syntax
  - Logic error occurs when you enter an instruction that does not give you the expected results
  - You debug to locate and remove errors
- To create and execute a C++ program, you need to have a text editor and a C++ compiler
  - Compiler translates source code into object code
  - Linker produces executable file

Summary (continued)

- Comments are internal documentation of programs
- C++ programs typically include at least one directive
- `using` statements tell compiler where it can find the definition of certain keywords
- A function is a block of code that performs a task
  - `main()` is where the execution of program begins
- The first line in a function is called the header
  - After the header comes the body, enclosed in braces

Application Lesson: Completing the Problem-Solving Process

- Lab 3.1: Stop and Analyze
- Lab 3.2:
  - Use Microsoft Visual C++ 2005 Express Edition
- Lab 3.3:
  - Modify the program created in Lab 3.2 so that it doesn't use a processing item
- Lab 3.4: Desk-Check Lab
- Lab 3.5: Debugging Lab

Application Lesson: Completing the Problem-Solving Process (continued)

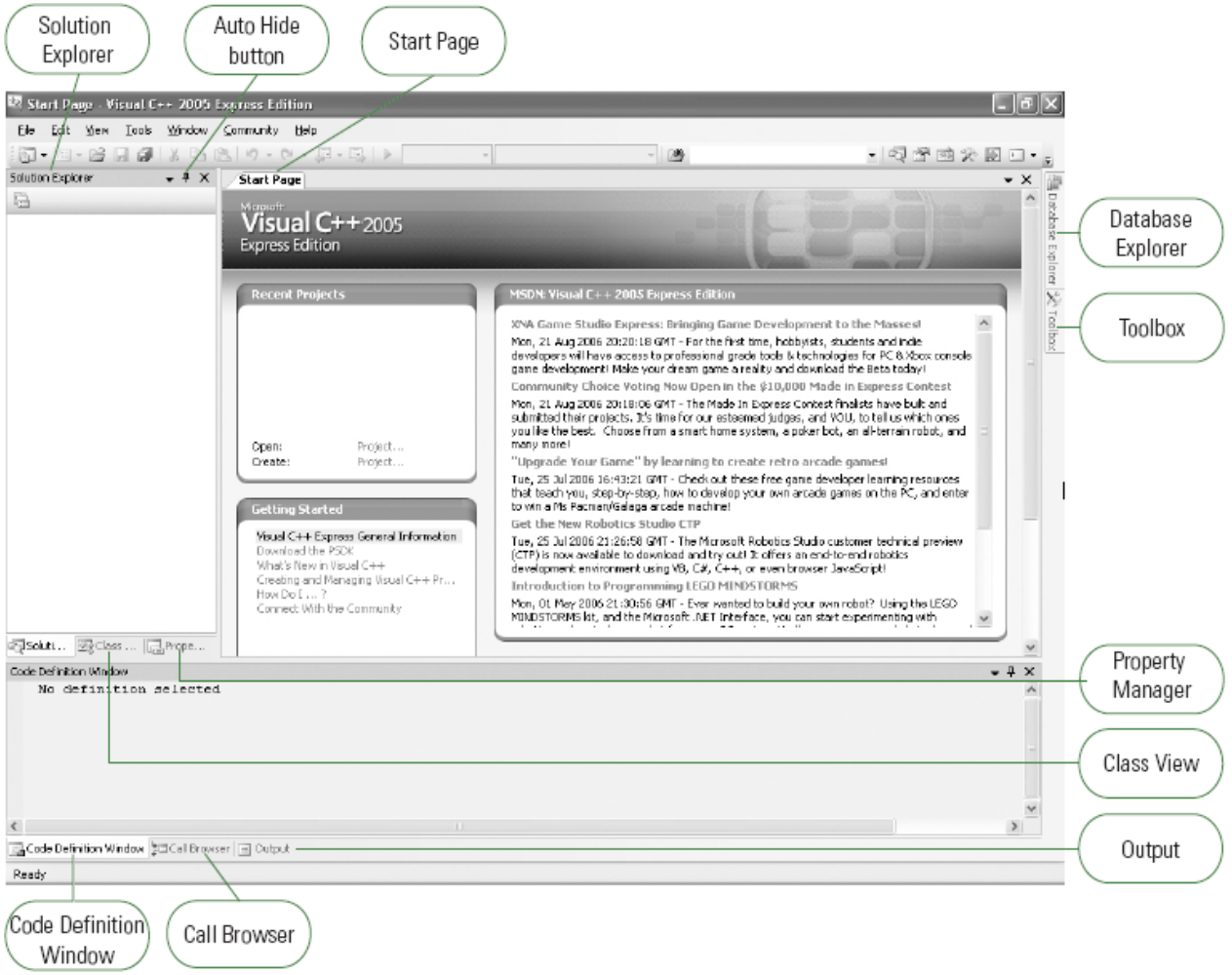


Figure 3-22: Default window layout for Visual C++ 2005 Express Edition