COMPREHENSIVE EDITION

PROGRAMMING AND PROBLEM SOLVING WITH C++

SIXTH EDITION

Nell Dale and Chip Weems

Chapter 4

Program Input and the Software Design Process

Chapter 4 Topics

- Object-Oriented Design Principles
- Functional Decomposition Methodology
- Software Engineering Tip Documentation

Functional Decomposition

- A technique for developing a program in which the problem is divided into more easily handled subproblems
- The solutions of these subproblems create a solution to the overall problem

Functional Decomposition

In functional decomposition, we work **from the abstract** (a list of the major steps in our solution) **to the particular** (algorithmic steps that can be translated directly into code in C++ or another language)

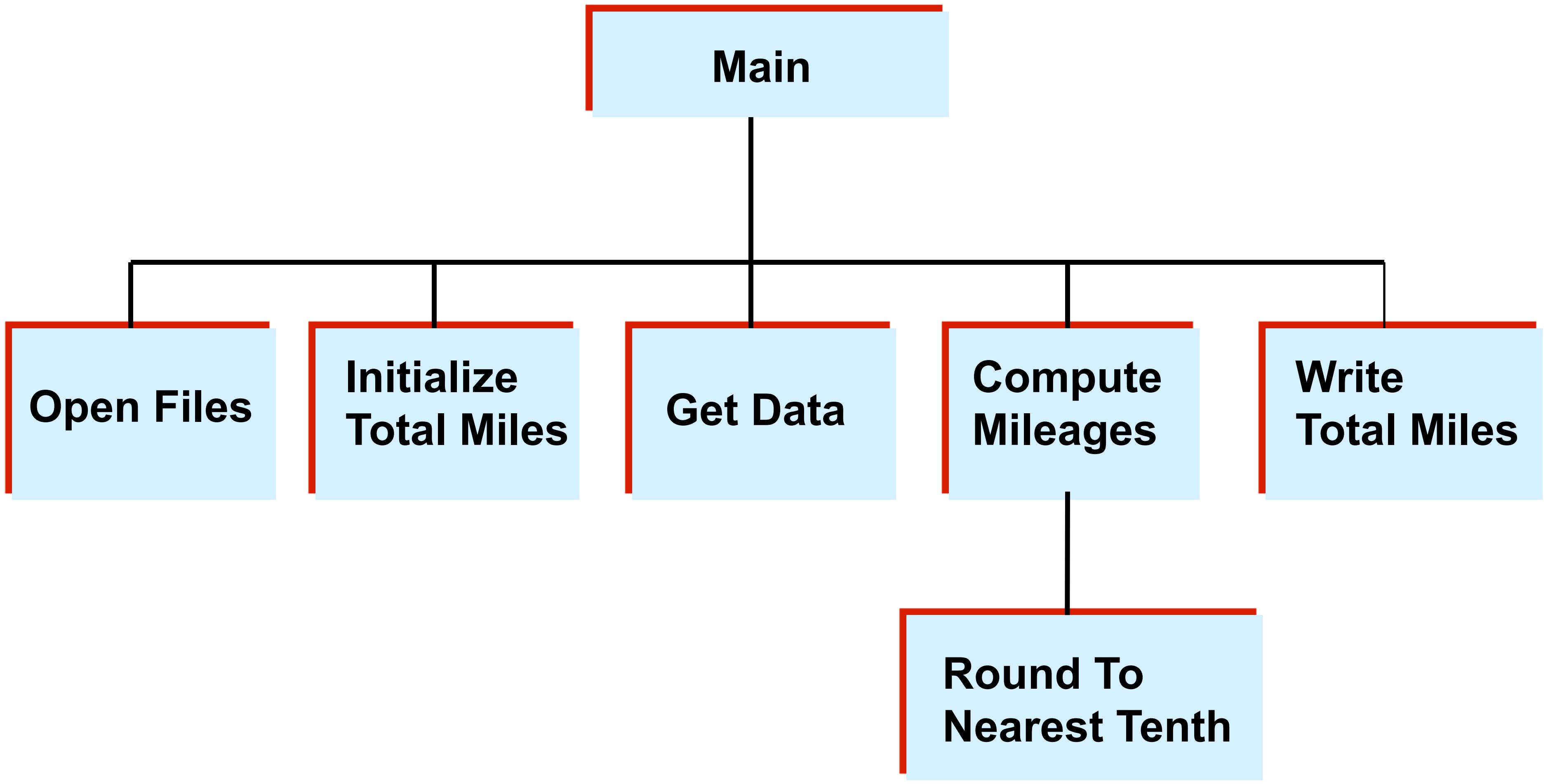
Functional Decomposition

- Focus is on actions and algorithms
- Begins by breaking the solution into a series of major steps; process continues until each subproblem cannot be divided further or has an obvious solution

Functional Decomposition

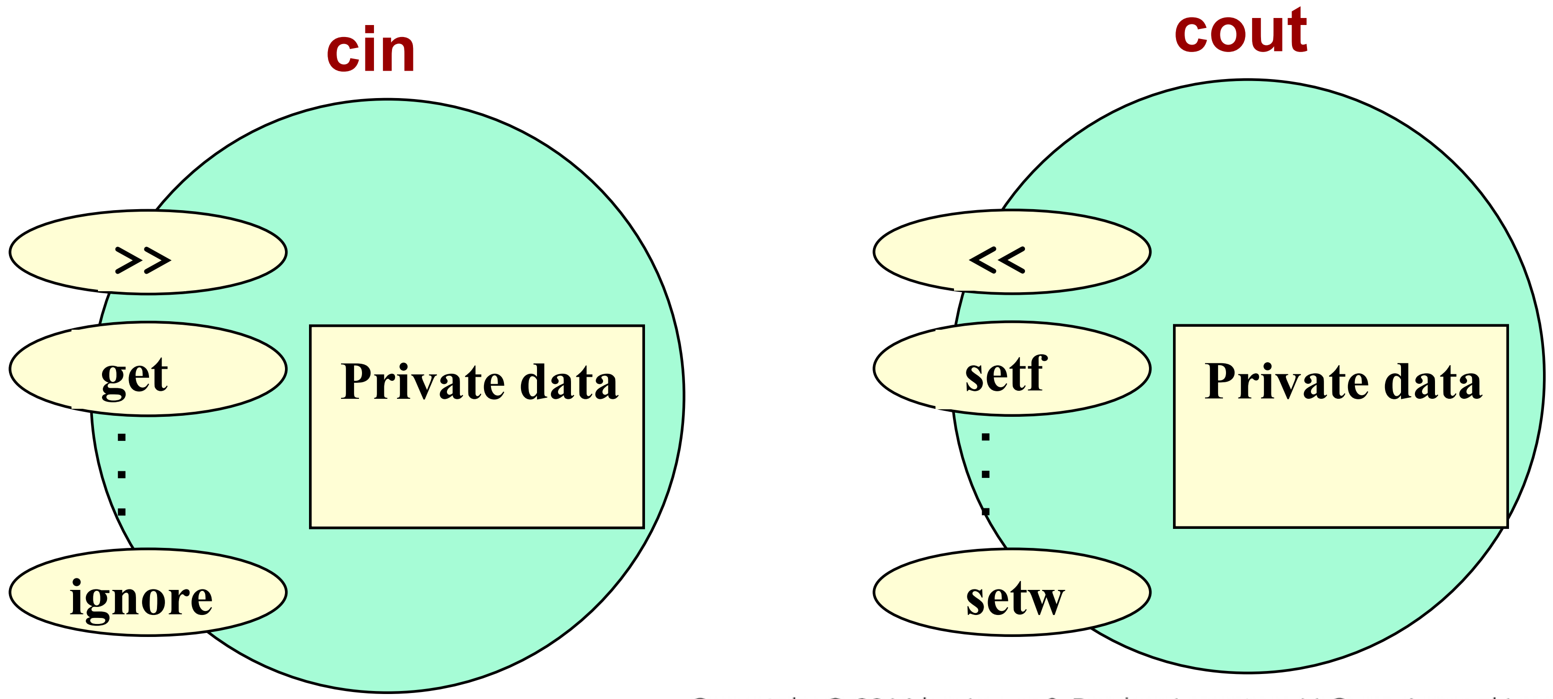
- **Units** are *modules* representing algorithms
  - A module is a collection of concrete and abstract steps that solves a subproblem
  - A module structure chart (hierarchical solution tree) is often created
- **Data** plays a secondary role in support of actions to be performed

Module Structure Chart



Object-Oriented Design

A technique for developing a program in which the solution is expressed in terms of objects -- self-contained entities composed of data and operations on that data



More about OOD

- Languages supporting OOD include: C++, Java, Smalltalk, Eiffel, CLOS, and Object-Pascal

- A ***class*** is a programmer-defined data type and objects are variables of that type

More about OOD

- In C++, **cin** is an object of a data type (class) named istream, and **cout** is an object of a class ostream.
- Header files iostream and fstream contain definitions of stream classes
- A class generally contains **private** data and **public** operations (called ***member functions***)

Object-Oriented Design (OOD)

- Focus is on entities called objects and operations on those objects, all bundled together

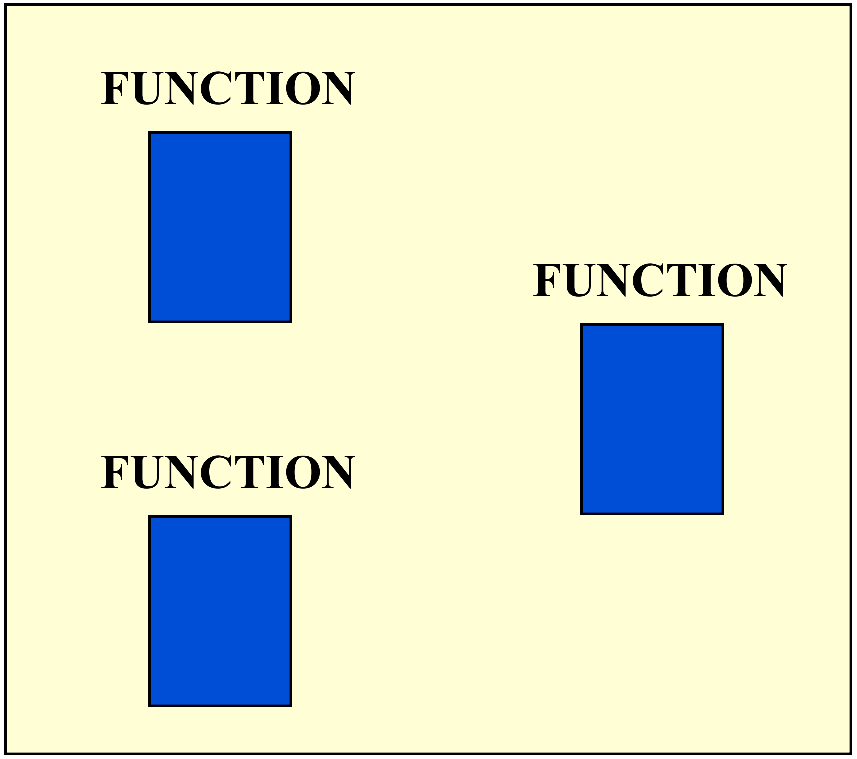
- Begins by identifying the major objects in the problem, and choosing appropriate operations on those objects

Object-Oriented Design (OOD)

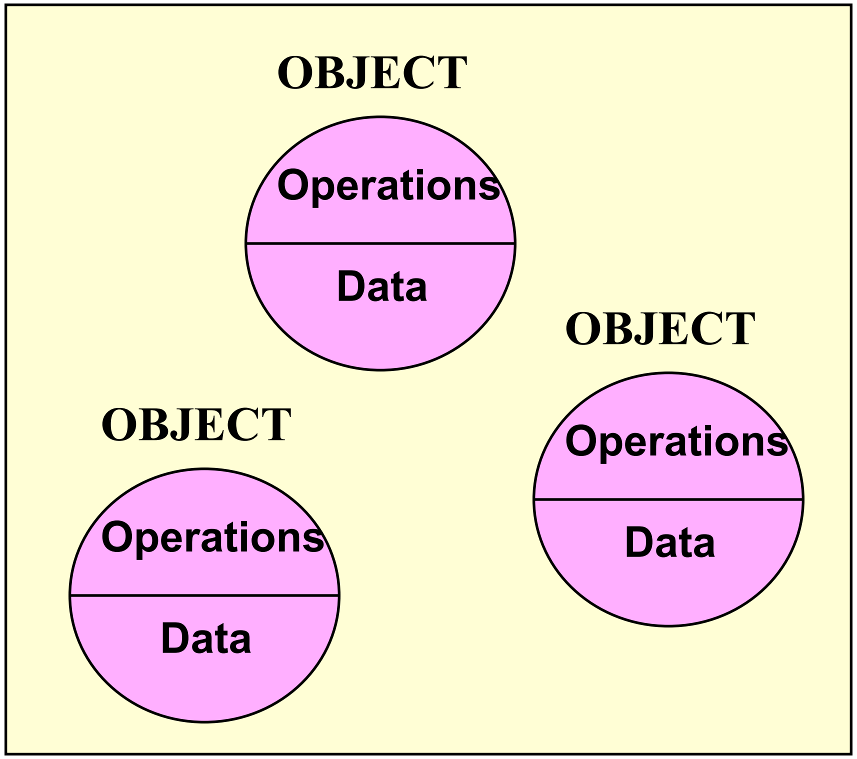
- Units are *objects*; programs are collections of objects that communicate with each other
- Data plays a leading role; algorithms are used to implement operations on the objects and to enable object interaction

Two Programming Methodologies

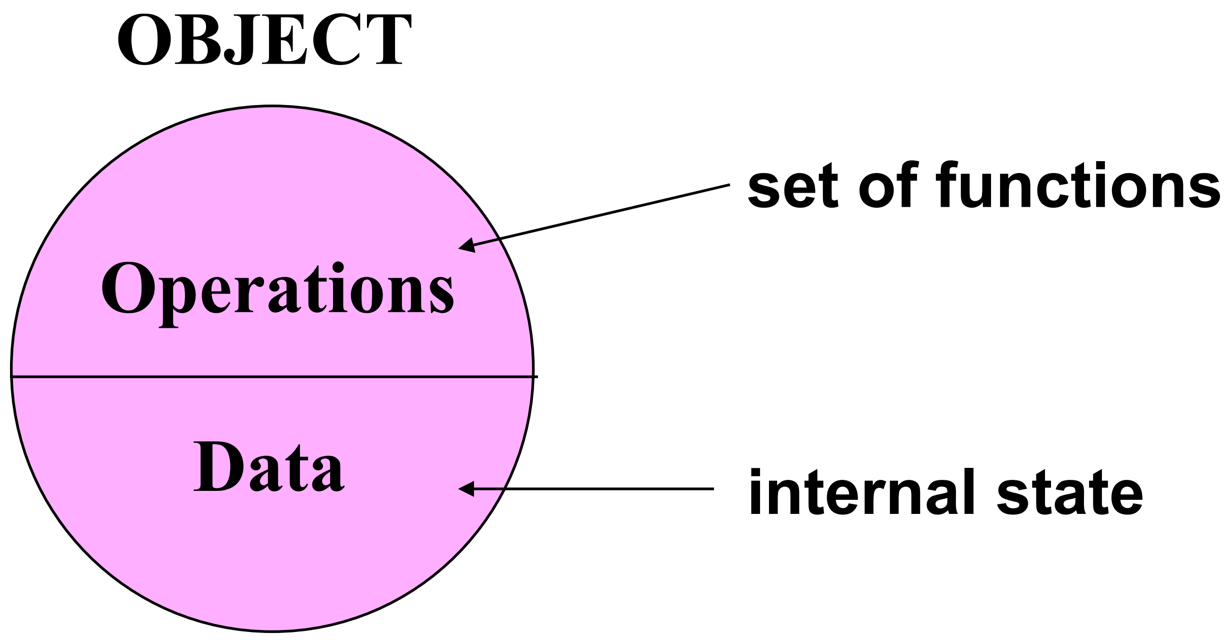
Functional Decomposition



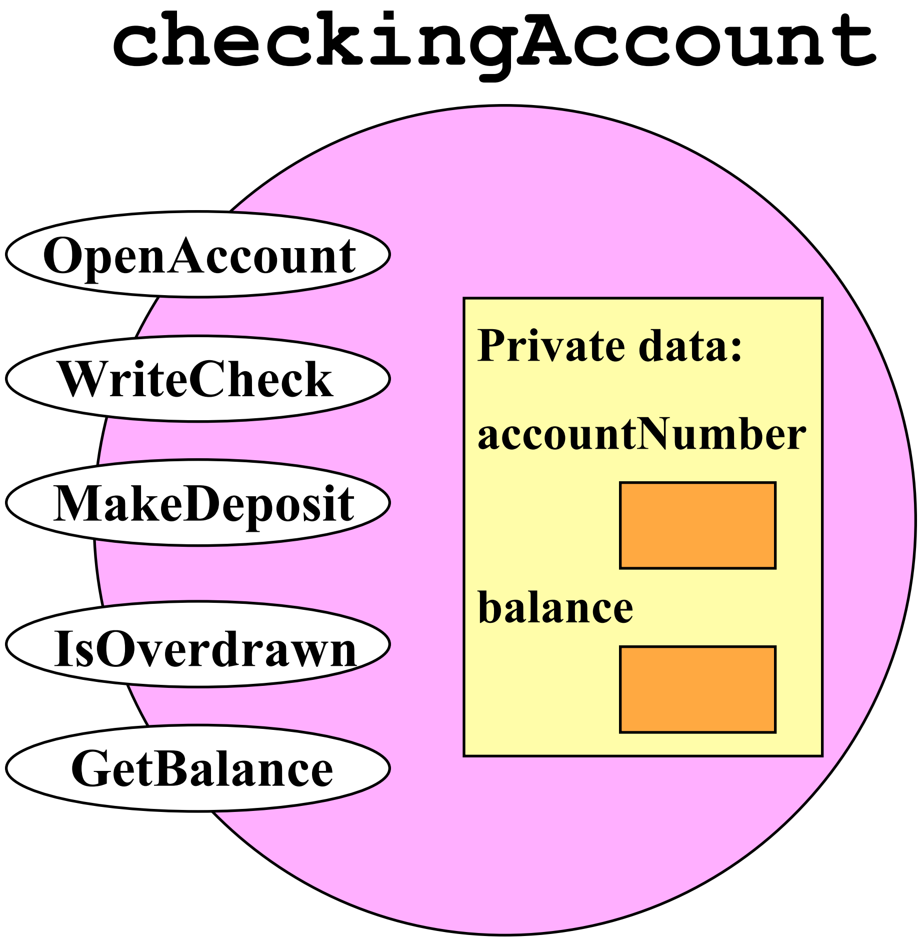
Object-Oriented Design



What is an object?



An object contains data and operations



OOD Used with Large Software Projects

- Objects within a program often model real-life objects in the problem to be solved

- Many libraries of pre-written classes and objects are available as-is for re-use in various programs

OOD Used with Large Software Projects

- The OOD concept of inheritance allows the customization of an existing class to meet particular needs without having to inspect and modify the source code for that class
- This can reduce the time and effort needed to design, implement, and maintain large systems

Software Engineering Tip
Documentation

- **Documentation includes the written problem specification, design, development history, and actual code of a problem**
- **Good documentation helps other programmers read and understand a program**
- **Good documentation invaluable when software is being debugged and modified (maintained)**

Software Engineering Tip
Documentation

- Documentation is both external and internal to the program
- External documentation includes the specifications, development history, and the design documents
- Internal documents includes the program format and self-documenting code--meaningful identifiers and comments

Software Engineering Tip Documentation

- Comments in your programs may be sufficient for someone reading or maintaining your programs
- However, if the program is to be used by non-programmers, then you must also provide a user's manual
- Keep documentation up-to-date and indicate any changes you made in pertinent documentation

Lab I/O

- Username Generator
- or
- Phone Number Exchange

Names in Multiple Formats

Problem

You are beginning to work on a problem that needs to output names in several formats along with the corresponding social security number.

As a start, you decide to write a short C++ program that inputs a social security number and a single name and displays it in the different formats, so you can be certain that all of your string expressions are correct.

Algorithm

```
Main Module                              Level 0
    Open files
    Get social security number
    Get name
    Write data in proper formats
    Close files
Open Files                               Level 1
    inData.open("name.dat")
    outData.open("name.out")
```

Get Name

```
Get first name
Get middle name or initial
Get last name
```

Write Data in Proper Formats

Write first name, blank, middle name, blank, last name, blank, social security number

Write last name, comma, first name, blank, middle name, blank, social security number

Write last name, comma, blank, first name, blank, middle initial, period, blank, social security number

Write first name, blank, middle initial, period, blank, last name

```
Middle initial                              Level 2
    Set initial to middleName.substr(0, 1)
      + period
Close files
    inData.close()
    outData.close()
```

C++ Program

```
//*****************************************************************
// Format Names program
// This program reads in a social security number, a first name
// a middle name or initial, and a last name from file inData.
// The name is written to file outData in three formats:
//    1. First name, middle name, last name, and social security
//    number.
//    2. last name, first name, middle name, and social
//    security number
//    3. last name, first name, middle initial, and social
//    security number
//    4.  First name, middle initial, last name
//*****************************************************************
```

```
#include <fstream>                // Access ofstream
#include <string>                // Access string
using namespace std;

int main()
{
    // Declare and open files
    ifstream inData;
    ofstream outData;
    inData.open("name.dat");
    outData.open("name.out");
```

```
// Declare variables
    string socialNum;        // Social security number
    string firstName;        // First name
    string lastName;               // Last name
    string middleName;       // Middle name
    string initial;          // Middle initial
```

```
// Read in data from file inData
inData >> socialNum >> firstName >> middleName
       >> lastName;
// Access middle initial and append a period
initial = middleName.substr(0, 1) + '.';
```

```
// Output information in required formats
    outData << firstName << ' ' << middleName << ' '
            << lastName << ' ' << socialNum << endl;
    outData << lastName << ", " << firstName << ' '
            << middleName << ' ' << socialNum  << endl;
    outData << lastName << ", " << firstName << ' '
            << initial << ' ' << socialNum  << endl;
    outData << firstName << ' ' << initial << ' '
            << lastName;
```

```
  // Close files
     inData.close();
     outData.close();
     return 0;


}
```