Chapter 1
Problem Solving with C++

3rd Edition

Computing Fundamentals with C++

Rick Mercer
Franklin, Beedle & Associates

Goals

- Understand an example of program development
- Understand the characteristics of a good algorithm
- Understand how algorithmic patterns can help design programs

Program Development

- One program development strategy has these three steps:
  - Analysis: Understand the problem
  - Design: Organize the solution
  - Implementation: Get the solution running
- *Program development* is the progression from analysis to design to implementation
- We'll see deliverables from each phase

Analysis

- Synonyms
  - inquiry, examination, study
- Activities
  - Read and understand the problem statement
  - Name the pieces of information necessary to solve the problem
    - these data names are part of the solution

Use good names

- Using this grade scale, compute a course grade

Item	Weight
Projects	50%
Midterm	20%
Final Exam	30%

- Name the input data:

```
projects    midterm    finalExam
```

- Name the output:

```
courseGrade
```

Object Attributes

- The data things are called *objects* and have these three important characteristics:
  - a reference to the object like `myAccount`
  - state (values) like `ID` and `balance`
  - a set of operations to manipulates the values like `deposit(double)` and `withdraw(double)`

To input or output?

- It helps to distinguish objects that are either input or output
  - ***Output***: Information the computer must display after the processing has occurred
  - ***Input***: Information the user must supply to solve the problem.

Sample problems help

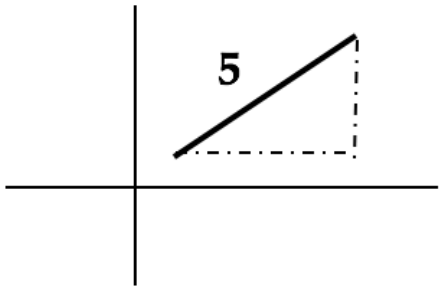
- It helps to provide sample problems
  - Given specific input data, determine the output

```
x1   1.0
y1   1.0
x2   5.0
y2   4.0
length 5.0
```

Input

Output

$$\text{Length} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$



Other Sample Problems

Mini Problem Description	*Object Names*	*Sample Problem*	*Input or Output?*
Compute the average of three test scores	test1	70.0	Input
	test2	80.0	"
	test3	93.0	"
	testAverage	81.0	Output
Compute the roots of a quadratic equation (ax^2+bx+c)	a	1.0	Input
	b	0.0	"
	c	-1.0	"
	root1	1.0	Output
	root2	-1.0	"
Compute a monthly loan payment	amount	12500.00	Input
	rate	0.08	"
	months	48	"
	payment	303.14	Output

Summary of Analysis

- Activities performed during analysis
  - Read and understand the problem
  - Decide what object(s) represent the answer—the *output*
  - Decide what object(s) the user must enter to get the answer—the *input*

Design

- Synonyms of design: model, think, plan, devise, pattern, propose, outline
- We'll use these design tools:
  - algorithms
  - algorithmic patterns
  - algorithm walkthroughs

Algorithms

- An algorithm is a set of activities that solves a problem
- An algorithm must:
  - list activities that must be performed
  - list the activities in the proper order

Bake a Cake

- A recipe (a.k.a. an algorithm)
  - Preheat Oven
  - Grease Pan
  - Mix ingredients
  - Place ingredients into pan
  - place pan in oven
  - remove pan after 35 minutes
- Switch some activities around
- What's missing?

Algorithmic Patterns

- Pattern: Anything shaped or designed to serve as a model or guide in making something else
- Algorithmic Pattern: A pattern that occurs frequently during program development.
- The Input/Process/Output (IPO) Pattern is used during the case study of Chapter 1

IPO Algorithmic Pattern

Pattern:	Input/Process/Output (IPO)
Problem:	The program requires input to generate the desired info
Outline:	1. obtain input data from user 2. process input data 3. output the results

Patterns ala Alexander

"Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice."

From A Pattern Language, Christopher Alexander, Oxford University Press

Example of Algorithm Design

- The deliverable from this design phase will be an algorithm.
- The IPO patterns provides a guide to design this more specific algorithm (that is a bit sketchy):

IPO Model	***One Specific IPO Case Study***
I)nput	Obtain `projects midTerm finalExam`
P)rocess	Compute the `courseGrade`
O)utput	Show the `courseGrade`

Refining steps in algorithms

- We often need to refine some steps
  - For example, "Compute the course grade" might now be refined with the C++ mathematical addition + and multiplication * symbols and names for the objects:

```
// Compute the courseGrade
courseGrade =   projects * 0.50
              + midterm * 0.20
              + finalExam * 0.30;
```

Algorithm Walkthrough

- Suggestion: Use an algorithm walkthrough to review the algorithm and find a test case
- Simulate what the computer would do if given the instructions.
  - If input occurs, copy values by object names
  - if processing occurs, change an object's value
  - if output occurs, write that output

Input/Process/Output (IPO)

I)nput Retrieve some example values from the user and store them into the objects as shown:

```
projects 92      midterm 82   finalExam 78
```

P)rocess Use this input data to compute `courseGrade`

```
courseGrade = 0.5*projects + 0.2*midterm + 0.3*finalExam
              0.5 * 92     + 0.2 * 82    + 0.3 * 78
                46.0       +    16.4     +   23.4
courseGrade = 85.8
```

O)utput Display the course grade

Implementation

- Synonyms for Implementation
  - accomplishment, making good, execution
- Implementation deliverable: computer program

Activity	Deliverable
1) Translate algorithm into a programming language	Source Code
2) Compile source code into object code	Machine Language
3) Link together the object code files	Running program
4) Verify the program does what it is supposed to do	Correct program

Translation into Code

- Pseudo code algorithm

  Display the value of the course grade

- Our programming language translation

```
cout << "Course grade: " << courseGrade;
```

- Once the algorithm is translated into a programming language abstraction:
  - use the compiler to generate machine code
  - use the linker to create executable program
  - run the program
  - test the program

Preview of C++

```
#include <iostream>
using namespace std;

int main()  {
  // Declare the objects to be given values
  int projects, midterm, finalExam;

  // I)nput
  cout << "Enter projects score: ";
  cin >> projects;
  cout << "Enter midterm: ";
  cin >> midterm;
  cout << "Enter final exam: ";
  cin >> finalExam;

  // P)rocess
  double courseGrade = (0.5 * projects) +
                       (0.2 * midterm)  +
                       (0.3 * finalExam);

  // O)utput
  cout << "Course grade: " << courseGrade << "%" << endl;
}
```

One Dialog:

```
Enter projects score: 92
Enter midterm: 82
Enter final exam: 78
Course grade: 85.8%
```

Testing

- Testing occurs at any point in program development:
  - Analysis: example problems
  - Design: algorithm walkthroughs
  - Implementation: run program with several sets of input data
- A running program isn't always right
  - We can gain confidence that it is correct by running the program with many test cases
    - Try all 100s, all 0s, all the same, several sets where all are different values

Objects Types, and Variables

- To input something that can be used by a program, there must be a place to store it in the memory of the computer
- These "places" are objects, which is a region of memory (a bunch of bits)
- variable: a named object that can have changing values

Objects

- We understand objects by the
  - the value(s) they store
  - the operations that can be applied
- The Course Grade problem used four numeric objects (`double` that has *double* the precision of `float`)
  - *values*: each object of the double class stores one floating point number
  - *operations*: operations such as input with `cin >>`, output `cout <<`, assignment with `courseGrade =`, addition with `+` and multiplication with `*`

Characteristics of Objects

- Name
  - All four objects have their own unique name
- Values (State)
  - The state of the double class objects was set either through an input operation:

    ```
    cin >> projects;
    ```
  - or through an assignment operation:

    ```
    courseGrade = 0.0;
    ```

Operations applied to objects

- Addition and multiplication operations are applied to some double objects:

```
0.25 * test1 + 0.25 * test2 + 0.50 * finalExam
```

- There is an input operation applied to the keyboard object named `cin`

```
cin >> test1; // This alters test1
```

- The state of `courseGrade` is examined through an output operation (`cout` is the object that represents the output console)

```
cout << courseGrade;
```

Types

- **type**: a set of values and the operations on those values
- C++ has **fundamental types**
  - `int` stores integers
    - operations + - / * =
  - `float` stores floating-point numbers like `1.234`
    - operations + - / * =
  - `double` stores floating-point numbers like `1.234`
    - operations + - / * =

Compound Types

- **compound type**: a type composed of several other types
  - `string` stores a literal string like `"Kim Baker"`
    - operations: `size append []`
  - `ostream`: sends values to an output stream such as the console or a file
    - operations: `width precision <<`
  - `istream`: sends values to an output stream such as the console or a file
    - operations: `peek getline >>`
  - `bankAccount`: store data about an account at a bank
    - operations: `deposit withdraw getBalance`

Pick the right type

- Which type of object and what name would you use to represent each of the following?
  - The number of students in a course _____
  - An effective annual percentage rate _____
  - A person's name _____
  - Obtain keyboard input _____