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

<table>
<thead>
<tr>
<th>Item</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Projects</td>
<td>50%</td>
</tr>
<tr>
<td>Midterm</td>
<td>20%</td>
</tr>
<tr>
<td>Final Exam</td>
<td>30%</td>
</tr>
</tbody>
</table>

• 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

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

\[
\begin{align*}
  x_1 & = 1.0 \\
  y_1 & = 1.0 \\
  x_2 & = 5.0 \\
  y_2 & = 4.0 \\
  \text{length} & = 5.0
\end{align*}
\]
Other Sample Problems

<table>
<thead>
<tr>
<th>Mini Problem Description</th>
<th>Object Names</th>
<th>Sample Problem</th>
<th>Input or Output?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compute the average of three test scores</td>
<td>test1, test2, test3, testAverage</td>
<td>70.0, 80.0, 93.0, 81.0</td>
<td>Input, &quot;&quot;, &quot;&quot;, Output</td>
</tr>
<tr>
<td>Compute the roots of a quadratic equation ((ax^2+bx+c))</td>
<td>a, b, c, root1, root2</td>
<td>1.0, 0.0, -1.0, 1.0, -1.0</td>
<td>Input, &quot;&quot;, &quot;&quot;, Output</td>
</tr>
<tr>
<td>Compute a monthly loan payment</td>
<td>amount, rate, months, payment</td>
<td>12500.00, 0.08, 48, 303.14</td>
<td>Input, &quot;&quot;, &quot;&quot;, Output</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Pattern:</th>
<th>Input/Process/Output (IPO)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem:</td>
<td>The program requires input to generate the desired info</td>
</tr>
</tbody>
</table>
| 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 ModelOne Specific IPO Case Study**

  I) Input
     - Obtain projects midTerm finalExam

  P) Process
     - Compute the courseGrade

  O) Output
     - 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)

Input  Retrieve some example values from the user and store them into the objects as shown:

projects 92  midterm 82  finalExam 78

Process  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

Output  Display the course grade
## Implementation

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

<table>
<thead>
<tr>
<th>Activity</th>
<th>Deliverable</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) Translate algorithm into a programming language</td>
<td>Source Code</td>
</tr>
<tr>
<td>2) Compile source code into object code</td>
<td>Machine Language</td>
</tr>
<tr>
<td>3) Link together the object code files</td>
<td>Running program</td>
</tr>
<tr>
<td>4) Verify the program does what it is supposed to do</td>
<td>Correct program</td>
</tr>
</tbody>
</table>
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
#include <iostream>
using namespace std;

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

    // I)input
    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;
}
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:
  
  \[
  0.25 \times \text{test1} + 0.25 \times \text{test2} + 0.50 \times \text{finalExam}
  \]

- There is an input operation applied to the keyboard object named \text{cin}
  
  \text{cin} \gg \text{test1}; \quad // \text{This alters test1}

- The state of \text{courseGrade} is examined through an output operation (\text{cout} is the object that represents the output console)
  
  \text{cout} \ll \text{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 _____________