

Chapter 3

Developing a Program

PRELUDE TO PROGRAMMING, 4TH EDITION BY ELIZABETH ORANGE

3.1 The Program Development Cycle

Problem solving principles

- Completely understand the problem
- Devise a plan to solve it
- Carry out the plan
- Review the results

Writing a program

- 1) Analyze the problem
- 2) Design the program
- 3) Code the program
- 4) Test the program

PRELUDE TO PROGRAMMING, 4TH EDITION BY ELIZABETH ORANGE

1. Analyze the problem

Identify desired results (**output**)

Determine **input** needed to produce those results

Example: Create a program to generate 6 numbers to play the lottery

- Is 7, 7, 7, 7, 7, 7 ok?
- Is -3, 0, 8, 9, 689, 689 ok?
- Is 1, 2, 6, 47.98765, 88, 93.45 ok?
- These are all 6 numbers but we see we must be more specific
- Desired results: 6 different positive integers within the range of 1 to 40

PRELUDE TO PROGRAMMING, 4TH EDITION BY ELIZABETH ORANGE

2. Design the program

Create a detailed description of program

- Use charts, models, or ordinary language (**pseudocode**)

Identify **algorithms** needed

- Algorithm: a step-by-step method to solve a problem or complete a task

Algorithms must be:

- Well defined
- Well ordered
- Must produce some result
- Must terminate in a finite time

PREFACE TO PROGRAMMING, 4TH EDITION BY ELIZABETH ORANGE

3. Code the program

Translate charts, models, pseudocode, or ordinary language into **program code**

Add statements to document what the code does

- Internal documentation
- External documentation

Each programming language uses its specific **syntax**

PREFACE TO PROGRAMMING, 4TH EDITION BY ELIZABETH ORANGE

Syntax

Correct syntax for telling your friend where you put a cheese sandwich is:

"I have put it on the table."

Incorrect use of English syntax to say:

"I have it on the table put."

All the right words are there, but without proper syntax, the sentence is gibberish in English.

But translated word for word, the second sentence is correct syntax in German.

PREFACE TO PROGRAMMING, 4TH EDITION BY ELIZABETH ORANGE

4. Test the program

In analysis phase: continually **ask questions**

- Did I interpret data correctly?
- Does program fulfill requirements?
- Are my formulas or procedures correct? Etc...

In design phase: use **desk-checking** to walk through the program

In coding phase: software will alert you to **errors in syntax** but not in the **logic** of the program

Finally, **test your program** with as many sets of test data as possible

- Use good data, bad data, data you know the answers for, etc.

PREFACE TO PROGRAMMING, 4TH EDITION BY ELIZABETH ORANGE

Additional Steps in the Cycle

- Create an outline of the program so that it is apparent what major tasks and subtasks have to be accomplished and the relationships among these tasks
- Describe in detail how each of these tasks is to be carried out

To put a commercial program (produced by a software publishing company) you may need to:

- Create a user's guide
 - to help users can understand the intricacies of the program
- Create help files
 - installed with the software for users to get on-screen help
- Train employees to provide telephone or web-based customer support
- Duplicate disks and accompanying materials for distribution
- Advertise the program to attract buyers

PREFACE TO PROGRAMMING, 4TH EDITION BY ELIZABETH ORANGE

Program development is a process

- Program development is a **cyclical process** that often requires returning to earlier steps and, with complex programs, may take many months
- The design process may uncover flaws in the analysis
- Coding may find problems leading to modifications or additions to the design
- Testing inevitably uncovers problems that require returning to previous phases

PREFACE TO PROGRAMMING, 4TH EDITION BY ELIZABETH ORANGE

The Sale Price Example

A local department store wants to develop a program which, when given an item's original price and the percentage it is discounted, will compute the sale price, with sales tax.

Output required: name of item, discounted price, amount of sales tax, total price

Variables needed: **ItemName, SalePrice, Tax, TotalPrice**

Input required: name of item, original price, percent discounted

More variables: **OriginalPrice, DiscountRate**

Formulas required:

New variable needed: **AmountSaved**

SalePrice = **OriginalPrice** - **AmountSaved**

AmountSaved = **OriginalPrice** * (**DiscountRate**/100)

Tax = **SalePrice** * .065

TotalPrice = **SalePrice** + **Tax**

PREFACE TO PROGRAMMING, 4TH EDITION BY ELIZABETH ORANGE

Design: Input → Processing → Output

Input	Perform Calculations (Process)	Output
Input variables:	Computations:	Display:
ItemName	AmountSaved = OriginalPrice * DiscountRate /100	TotalPrice
DiscountRate	SalePrice = OriginalPrice - AmountSaved	ItemName
OriginalPrice	Tax = SalePrice * .065	Tax
	TotalPrice = SalePrice + Tax	SalePrice

PREFACE TO PROGRAMMING, 4TH EDITION BY ELIZABETH ORANGE

3.2 Program Design

Modular Programming

To begin designing a program: identify the major tasks the program must accomplish.

Each of these tasks becomes a **program module**.

- if needed, break each of these fundamental "high-level" tasks into **submodules**
- Some submodules might be divided into submodules of their own
- this process can be continued as long as necessary
- Identifying the tasks and subtasks is called **modular programming**

PREFACE TO PROGRAMMING, 4TH EDITION BY ELIZABETH ORANGE

Using Modules and Submodules

- A module performs a single task.
- A module is self-contained and independent of other modules.
- A module is relatively short. Ideally, statements should not exceed one page.

Benefits of Modular Programming

- program is easier to read
- easier to design, code, and test the program one module at a time
- different program modules can be designed and/or coded by different programmers
- a single module may be used in more than one place in the program
- modules that perform common programming tasks can be used in more than one program

PREFACE TO PROGRAMMING, 4TH EDITION BY ELIZABETH ORRANE

Pseudocode: uses short, English-like phrases to describe the outline of a program

Example: pseudocode for the Sale Price Program with modules:

```
Input Data module
    Prompt for ItemName, OriginalPrice, DiscountRate
    Input ItemName, OriginalPrice, DiscountRate
Perform Calculations module
    Set AmountSaved = OriginalPrice * (DiscountRate/100)
    Set SalePrice = OriginalPrice - AmountSaved
    Set Tax = SalePrice * .065
    Set TotalPrice = SalePrice + Tax
Output Results module
    Write ItemName
    Write OriginalPrice
    Write DiscountRate
    Write SalePrice
    Write Tax
    Write TotalPrice
```

PREFACE TO PROGRAMMING, 4TH EDITION BY ELIZABETH ORRANE

Refined Pseudocode for the Sale Price Program

```
Input Data module
    Write "What is the item's name?"
    Input ItemName
    Write "What is its price and the percentage discounted?"
    Input OriginalPrice
    Input DiscountRate
Perform Calculations module
    Set AmountSaved = OriginalPrice * (DiscountRate/100)
    Set SalePrice = OriginalPrice - AmountSaved
    Set Tax = SalePrice * .065
    Set TotalPrice = SalePrice + Tax
Output Results module
    Write "The item is: " + ItemName
    Write "Pre-sale price was: " + OriginalPrice
    Write "Percentage discounted was: " + DiscountRate + "%"
    Write "Sale price: " + SalePrice
    Write "Sales tax: " + Tax
    Write "Total: $" + TotalPrice
```

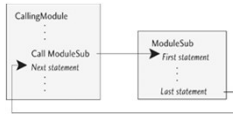
PREFACE TO PROGRAMMING, 4TH EDITION BY ELIZABETH ORRANE

Calling Modules

A **call statement** causes a submodule to be executed.

After a call statement, program control is transferred to the first line of the called module.

After all statements in the submodule have been executed, control returns to the line of code immediately below the call statement.



PREFACE TO PROGRAMMING, 4TH EDITION BY ELIZABETH CHORRAGE

The Main Module

- The **main module** is where program execution begins and normally ends.
- The main module is not a submodule of another.
- It is the parent module of the program's highest-level modules.
- The highest-level modules are called into action by the main module.

In the Sale Price Program, we add a Main module to call others:

```

Main module
    Call Input Data module
    Call Perform Calculations module
    Call Output Results module
  
```

PREFACE TO PROGRAMMING, 4TH EDITION BY ELIZABETH CHORRAGE

Format Output

- Include information about what the output means

If a program calculates the temperature converted from Fahrenheit to Celsius, the following output is confusing:

```

27
  
```

But the following output makes more sense:

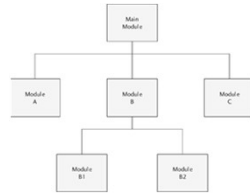
```

81 degrees Fahrenheit is
27 degrees Celsius
  
```

PREFACE TO PROGRAMMING, 4TH EDITION BY ELIZABETH CHORRAGE

Hierarchy Charts

- Like an organization chart
- Shows position of modules in the program
- Depicts what modules exist and how they are related
- Large programs need a “map” for documentation



PRELUDE TO PROGRAMMING, 4TH EDITION BY ELIZABETH CHURCH

3.3 Coding, Documenting, Testing

Coding

- Coding is done in a specific programming language. We will use pseudocode.
- This phase should only begin after a solid design exists.

Documenting

- Code needs to contain documentation that describes to the reader what the code is doing
- Two types of **comments** are used within the code
- **Internal documentation** is for the programmers to read
- **External documentation** is for the user

PRELUDE TO PROGRAMMING, 4TH EDITION BY ELIZABETH CHURCH

Comments: not processed by the computer, valued by other programmers

Header comments

- Appear at beginning of a program or a module
- Provide general information

Step comments or in-line comments

- Appear throughout program
- Explain purpose of specific portions of code

Often comments delineated by:

- `//`
- `/* comment goes here */`

PRELUDE TO PROGRAMMING, 4TH EDITION BY ELIZABETH CHURCH

Using comments for a program to find size of a room

```
// Program to calculate the area in square footage of a room
// Programmer: E. Drake, Santa Fe College
// Version 6.0 - January 1, 2015
// This program computes the area of a room, given its width and length
// Variables used: Width, Length, SquareFeet
// Declare the variables
  Declare Width As Float
  Declare Length As Float
  Declare SquareFeet As Float
// Get the values of the dimensions
Write "What are the length and width of the room in inches?"
  Input Length
  Input Width
// Calculate square footage
  Set SquareFeet = Width * Length
// Output the result
Write "Your room is " + SquareFeet + " square feet."
```

PREFACE TO PROGRAMMING, 6TH EDITION BY ELIZABETH DRANE

The Testing Phase

Testing

- Create **test data** that will be used to check the program's correctness.
 - Use **desk checking** (or walking through a program by hand with a set of data that you know the answer to).
- Check that the program will catch errors by using test data designed to create errors.
- The more testing of various types of data you can use, the more likely you are to have a program that is free of errors.

PREFACE TO PROGRAMMING, 6TH EDITION BY ELIZABETH DRANE

Types of Errors: Syntax Errors

Syntax errors: a violation of the programming language's rules for creating valid statements

- May be caused by incorrect grammar or punctuation, or misspelling a keyword
- The program will not run at all with syntax errors

PREFACE TO PROGRAMMING, 6TH EDITION BY ELIZABETH DRANE

Types of Errors: Logic Errors

Logic errors: the program runs, but does not produce the expected results

- May be caused by using an incorrect formula, or incorrect sequence of statements, etc.
- Sometimes called **runtime errors**
- Can be detected during the desk checking phase of the programming cycle

PREFACE TO PROGRAMMING, 4TH EDITION BY ELIZABETH ORRANGE

3.4 Commercial Programs: Testing and Documenting

External documentation

Purposes:

1. Documentation in a **user's guide** or on-screen **help** system provides information about the program for the end users
2. Documentation in a **maintenance manual** provides information about how the program code accomplishes its purposes

PREFACE TO PROGRAMMING, 4TH EDITION BY ELIZABETH ORRANGE

User's Guides:

- usually written during alpha or beta test phases by a **technical writer**

Documentation for other programmers:

- Program maintenance manual
 - For programming experts
 - Used to help them fix or enhance code written by other programmers
- Design documentation
 - Written by programmer to explain rationale behind methods and code used
- Trade Study documentation
 - A research tool
 - An attempt to find the best solution

PREFACE TO PROGRAMMING, 4TH EDITION BY ELIZABETH ORRANGE

3.5 Structured Programming

- A method for designing and coding programs in a systematic, organized manner
- It combines the principles of top-down design, modularity and the use of the three accepted control structures: **sequence**, **repetition** and **selection**
- Sequence, repetition and selection can be expressed in **pseudocode**, or with **flowcharts**

PRELUDE TO PROGRAMMING, 4TH EDITION BY ELIZABETH ORANGE






Flowcharts

A tool for programmers to design programs

- Describes the flow of a program module's execution with diagrams
- Completely different from hierarchy charts
- Connected symbols are used to describe sequence, repetition, and selection structures
- Some prefer to use flowcharting to learn how to express algorithms, and others prefer to use pseudocode
- Many programs are designed with a combination of pseudocode and flowcharts

PRELUDE TO PROGRAMMING, 4TH EDITION BY ELIZABETH ORANGE

Basic Flowcharting Symbols

Symbol	Name	Description
	Terminator	Represents the start or end of a program or module
	Process	Represents any kind of processing function; for example, a computation
	Input/output	Represents an input or output operation
	Decision	Represents a program branch point
	Connector	Indicates an entry to, or exit from, a program segment

PRELUDE TO PROGRAMMING, 4TH EDITION BY ELIZABETH ORANGE



Control Structures

In the 1960s computer scientists proved there are only 3 basic **control structures** (also called **constructs**) needed to create any program or algorithm!

Sequence – execute statements in sequential order

- The simplest of control structures – start at the beginning and continue in sequential order

Selection – selectively execute statements

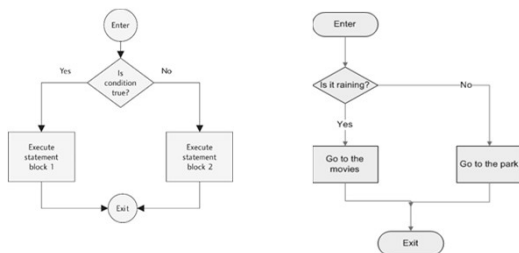
- Also called a **branch** or **decision**
- requires a **condition** to determine when to execute statements

Repetition – repeat statements more than once

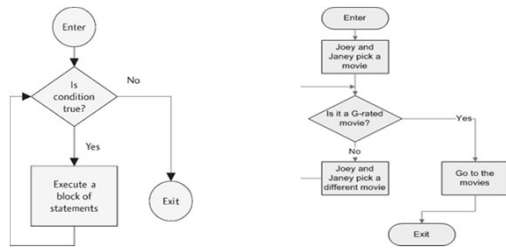
- Also called a **loop**
- needs a **stop condition**, i.e., the program will continue to loop until some condition is met

PRELUDE TO PROGRAMMING, 4TH EDITION BY ELIZABETH ORRANE

Flowchart for typical decision (selection) structures



Flowchart for typical repetition (loop) structures



PRELUDE TO PROGRAMMING, 6TH EDITION BY ELIZABETH GRABER

Style Pointers

- ☐ Write modular programs
- ☐ Use descriptive variable names
- ☐ Provide a welcome message for the user
- ☐ Use a prompt before an input
- ☐ Identify program output
- ☐ Document your programs

PRELUDE TO PROGRAMMING, 6TH EDITION BY ELIZABETH GRABER