Basic Programming Techniques

Problem Solving Series

Instructor's Guide

Table of Contents

Developed by the Teaching and Learning Laboratory at MIT
for the Singapore University of Technology and Design

Introduction

When to Use this Video

- In ISTD 102
- Prior knowledge: Introductory programming, simple data structures

Key Information

Duration: 14:05
Narrator: Niaja Farve, Ph.D. candidate
Materials Needed:
• paper
• pencil

Learning Objectives

After watching this video students will be able to:

- Divide a programming problem into simpler, analogous pieces.
- Solve the problem by combining solutions to simpler pieces.

Motivation

In computer programming, the process of constructing repeating elements, such as through recursive functions or loops, provides the foundation for writing any practical program. Obtaining this basic skill set and becoming comfortable with repetition is often one of the first major hurdles a student encounters in programming.

Student Experience

It is highly recommended that the video is paused when prompted so that students are able to attempt the activities on their own and then check their solutions against the video.

During the video, students will:

- Consider the idea of repetition in a simplified, non-programming example: eating cereal.
- Learn the basic code framework of recursive and iterative techniques.
- Formulate recursive and iterative solutions to a string manipulation example.
- Devise and check a recursive solution to the famous Towers of Hanoi problem.

Video Highlights

This table outlines a collection of activities and important ideas from the video.

Time	Feature	Comments
1:03	Introduction and motivation	Niaja Farve introduces the concept of repetitive techniques and breaking problems into simpler problems.
1:40	Motivating example: teaching a computer how to eat a bowl of cereal.	Students turn an everyday, human task that they are familiar with into a programming exercise.
3:06	General code framework for iteration	An outline of iterative code. Students can start with this framework and fill in the pieces needed for solving their particular problem.
3:30	General code framework for recursion	An outline of recursive code. Students can start with this framework and fill in the pieces needed for solving their particular problem.
5:16	Motivating example: "downup", string manipulation problem.	This is an opportunity for students to try repetitive programming techniques on a basic string manipulation problem.
8:34	Motivating example: Towers of Hanoi	This is an opportunity for students to apply repetitive techniques to a more complex problem.

Video Summary

In this video, Niaja Farve, doctoral student of Electrical Engineering and Computer Science, explains a very fundamental and essential programming skill - using repetitive techniques. First, using a light-hearted example of eating cereal, Niaja explains how to break problems into simpler yet similar pieces. She then explains how to use recursion and iteration to repetitively solve these simpler pieces, and consequently, the whole problem. Next, students are presented with a classic string manipulation problem to try their new skills. Finally, students are given a more complex programming problem: solving the Towers of Hanoi.

Pre-Video Materials

When appropriate, this guide is accompanied by additional materials to aid in the delivery of some of the following activities and discussions.

Students should have some prior exposure to programming before watching this video. The following pre-video activities could be used to reinforce this skill.

1. Suppose you have a function, **substring,** that when given a string, behaves as follows:
> substring("hippy",1,2)
hi
> substring("hippy",3,5)
ppy

What do you suppose is the output to substring("puppy",2,4)?

Suppose you have a function, **down,** that when given a string, behaves as follows:
> down("hippo")
hippo
hipp
hip
hi
h
> down("cat")
cat
ca
c

(a) What do you suppose is the output to **down**("puppy")?

(b) If we can assume the input string is 2 characters long, can you write the code for **down**? Use pseudocode or a language you are familiar with. Hint: use the substring function from question 1.

(c) Write the code for an input string 3 characters long.

Post-Video Materials

Use the following activities to reinforce and extend the concepts in the video.

1. Lets write a function, **sum**, which finds the sum of an input list of numbers.

(a) Fill in the following code:

```
Function sum( list L )
{
    answer = 0;
    for( each member m of list )
    {
        answer = _____
    }
    return answer;
}
```

(b) Now try to use recursion function calls rather than a loop to write the function.

2. We would like to write a function, **explode**. Given a string, it outputs the same string but with spaces between each character:

```
> explode( "dog" )
d o g
> explode( "hippo" )
h i p p o
```

(a) What is a smaller but similar version of this problem?

(b) Fill in the missing pieces to the recursive code below:

```
Function explode( string s )
{
  len = length(s);
  if( _______ )
  {
          print s;
  } else
  {
          print explode( substring( s,1,len-1 );
          print " ";
          print explode( substring( s, __ , __ );
  }
}
```

(c) Write the solution to **explode** using an iterative loop instead.

```
Function explode( string s )
{
  len = length(s);
  print substring(s,1,1)
  for( i=2 through len )
  {
        print " ";
        print substring(s,i,i);
  }
}
```

3. Suppose you are throwing a huge party and would like to invite not only all of your friends, but all of your friends' friends in the social network. As a helper function, we have **GetFriends**, which returns a list of friends of the input person:

```
> GetFriends( Jen )
{ Dipa, Janet, Xiao, Suzie }

> GetFriends( Dipa )
{ Jen, Frank }
```

(a) If you only have one friend, (it happens!) write the code to obtain the list of your one friend and all the friends of that friend.

(b) Now decide what type of repetitive technique you would like to use and write the function **GetPartyFriends**, which obtains a list of all your friends and your friend's friends.

(c) Finally, write the function **GetNStepFriends**, which takes a list of your friends, as well as a number *N* representing the number of levels of friends of friends to traverse.

Additional Resources

References

While every introductory computer programming textbook will discuss this topic, two well-written resources for learning about recursion and iteration are:

- Abelson, H., & Sussman, G. J. (1996). Structure and Interpretation of Computer Programs - 2nd Edition (MIT Electrical Engineering and Computer Science) (p. 683). The MIT Press. Free text online at http://www.mitpress.mit.edu/sicp/full-text/book/book.html, retrieved July 29, 2013.
- Harvey, B., & Wright, M. (1999). Simply Scheme - 2nd Edition: Introducing Computer Science (p. 611). The MIT Press. Free text online at http://www.eecs.berkeley.edu/~bh/ss-toc2.html, retrieved July 29, 2013.

A very thorough explanation of recursion and an extensive problem set can be found here:

- http://people.scs.carleton.ca/~lanthier/teaching/COMP1405/Notes/COMP1405_Ch7_Recursion.pdf, retrieved July 29, 2013.

Detailed descriptions of Towers of Hanoi solutions can be found on Wikipedia:

- http://en.wikipedia.org/wiki/Tower_of_Hanoi#Solution, retrieved July 29, 2013.