Procedural Programming

Programming Process

1. Understand the problem
2. Outline a general solution
3. Decompose the general solution into manageable component parts
4. Develop a specific solution for each component
5. Test the solution to each component for correctness
6. Implement the solution to each component in a specific programming language

Programming Process

7. Test implementation of each component
8. Assemble all of the components
9. Test the system of components
10. Document the system of components and the user related features of the program
11. Install the program on the target system of computer hardware and software
12. Test the program in the target environment
13. Train support personnel and users
14. Maintain and support the program

A First Example

Assume that you have been asked to write a computer program that allows the user to enter 2 numbers from the keyboard, adds, subtracts, multiplies and divides the numbers and displays the results to the screen.

Understand the Problem

- How can you solve a problem if you don’t know exactly what the problem is?
- Is there anything about the problem description that is unclear?
- Can you describe the input and output specifically?

Outline a General Solution

Input	Processing	Output
2 integers Num1 Num2	Display instructions Get input Do 4 calculations Display results	Sum Difference Product Quotient

Decompose the General Solution

- Breaking a complex problem into simpler problems is a useful problem solving strategy.
- We'll come back to this step when the problems get more complex.

Develop a Specific Solution

- This is HARD because people tend to "skip" steps that seem obvious. Computers can't do that.
- As an illustration, try to list the steps that you take to brush your teeth! Did you miss anything?

Develop a Specific Solution

ArithmeticCalculations

- Display instructions
- Get Num1 and Num2
- Sum = Num1 + Num2
- Difference = Num1 – Num2
- Product = Num1 * Num2
- Quotient = Num1/Num2
- Display appropriate labels and Sum, Difference, Product, Quotient

End

Test the Solution for Correctness

	Input		Expected Output			
	Num1	Num2	Sum	Diff.	Product	Quotient
Test Data 1	10	5	15	5	50	2
Test Data 2 (Not yet tested)	2	4	6	-2	8	.5

Test the Solution for Correctness

	Num1	Num2	Sum	Diff.	Product	Quotient
ArithmeticCalculations Display instructions Get Num1 and Num2 Sum = Num1 + Num2 Difference = Num1 – Num2 Product = Num1 * Num2 Quotient = Num1/Num2 Display appropriate labels and Sum, Difference, Product, Quotient End						

Develop a Specific Solution

- Some key terminology
  - Algorithm
  - Pseudocode
  - Variable
  - Assignment statement

Another Example

Assume that you have been asked by your local elementary school to create a program that tells students how many quarters, dimes, nickels and pennies should be received as change from a purchase under $1. The program should allow a student to enter the purchase price and should display the change.

Another Example

- Follow the same process
  - Can you ask clarifying questions?
  - Can you create an IPO chart?
  - Can you create an algorithm?
    - Can you do an example?
    - Can you generalize from the example?
  - Does the algorithm work?

Another Example

CalculatingChange

1. Display instructions
2. Get Price
3. Change = 100 – Price
4. Display Change
5. NumQtrs = integer part of (Change/25)
6. Change = Change – NumQtrs * 25
7. NumDimes = integer part of (Change/10)
8. Change = Change – NumDimes * 10
9. NumNickels = integer part of (Change/5)
10. Change = Change – NumNickels * 5
11. NumPennies = Change
12. Display labels and NumQtrs, NumDimes, NumNickels, NumPennies

End

Another Example

	Price	Change	Num-Qtrs	Num-Dimes	Num-Nickels	Num-Pennies
CalculatingChange Display instructions Get Price Change = 100 – Price Display Change NumQtrs = integer part of (Change/25) Change = Change – NumQtrs * 25 NumDimes = integer part of (Change/10) Change = Change – NumDimes * 10 NumNickels = integer part of (Change/5) Change = Change – NumNickels * 5 NumPennies = Change Display labels and NumQtrs, NumDimes, NumNickels, NumPennies End						

Practice Example

Assume that you have been asked to write a program to determine the sales tax on an item and display the tax and the total due to the user. The user will enter the price of the item in dollars and cents. The tax rate is 5%.

Practice Example

Assume that you have been asked by a friend, who is a beginning programming student, to write a program that converts 4 bit unsigned binary numbers into the decimal equivalent.

Control Structures

- Control structures are program building blocks that determine the order in which statements in a program are executed
- The structure theorem states that all programming problems can be solved by using 3 control structures
  - Sequence
  - Selection
  - Repetition

Selection

- Allows programs to “select” a set of instructions to execute based on the presence (or absence) of a condition
- Most programming languages have 2 selection statements
  - If statement (if in C++)
  - Case statement (switch in C++)

If Statement Example

```
If Price > 100 Then
   Display Error Message
Else
   Change = 100 – Price
   Display Change
   …
End if
```

If Statement in General

If condition Then

    Statements to execute when true

Else

    Statements to execute when false

End If

Selection Problem

Assume that you have been asked to write a program to find the largest of a set of 3 numbers. The user will enter 3 integers between 0 and 100. The program will display appropriate messages and display the largest of the 3 numbers.

Selection Problem

- The first 2 steps are exactly the same
  - Can you ask clarifying questions to make sure you understand the problem?
  - Can you create an IPO chart to outline a general solution?
- Start the algorithm development step the same way too
  - Can you describe in English the processing steps that you take when you order 3 numbers?

Selection Problem

```
FindLargest (Version1)
    Display instructions
    Get Num1, Num2 and Num3
    If Num1 > Num2 Then
            Largest = Num1
    Else
            Largest = Num2
    End If
    If Num3 > Largest Then
            Largest = Num3
    End If
    Display label and Largest
End
```

Selection Problem

```
FindLargest (Version 2)
        Display instructions
        Get Num1, Num2 and Num3
        If Num1 > Num2 Then
                If Num3 > Num1 Then
                        Largest = Num3
                Else
                        Largest = Num1
                End If
        Else
                If Num3 > Num2 Then
                        Largest = Num3
                Else
                        Largest = Num2
                End If
        End If
        Display label and Largest

End
```

Selection Problem

FindLargest (Version 3)

1. Display instructions
2. Get Num1, Num2 and Num3
3. If Num1 > Num2 and Num1 > Num3 Then
   Largest = Num1
4. Else If Num1 > Num2 and Num3 > Num1 Then
   Largest = Num3
5. Else If Num2 > Num1 and Num2 > Num3 Then
   Largest = Num2
6. Else If Num2 > Num1 and Num3 > Num2 Then
   Largest = Num3
7. End If
8. Display label and Largest

End

Selection Problem

	Num 1	Num 2	Num 3	Largest

FindLargest (Test Set 3)
Display instructions
Get Num1, Num2 and Num3
If Num1 > Num2 and Num1 > Num3 Then
Largest = Num1
Else If Num1 > Num2 and Num3 > Num1 Then
Largest = Num3
Else If Num2 > Num1 and Num2 > Num3 Then
Largest = Num2
Else If Num2 > Num1 and Num3 > Num2 Then
Largest = Num3
End If
Display label and Largest
End
	5	5	7	

Let's Generalize

- An if statement is a selection statement
- Many if statements look like this

```
If condition Then
    Statements to execute when true
Else
    Statements to execute when false
End If
```

- But, there are all kinds of "variations"
  - This one doesn't have an "else" part

```
If condition Then
    Statements to execute when true
End If
```

Let's Generalize

- But, there are all kinds of variations
  - This one has more than one "else" part

```
If condition1 Then
    Statements to execute if condition1 is true
Else If condition2 Then
    Statements to execute if condition2 is true
Else If condition3 Then
    Statements to execute if condition3 is true
...
Else
    Statements to execute if all conditions are false
End If
```

Let's Generalize

- But, there are all kinds of variations
  - This one is "nested"

```
If condition1 Then
    Statements to execute if condition1 is true
Else
    If condition2 Then
        Statements to execute if condition2 is true and condition 1 is false
    Else
        Statements to execute if condition2 is false and condition1 is false
    End If
End If
```

Let's Generalize

- Conditions can be more complex too
  - Most programming languages have 2 logical operators that can be used to make compound conditions
    - And
      - Gender = M and Age < 25
      - X > 1 and x < 10
    - Or
      - Gender = F or Age >= 25
      - X < 1 or X > 10

Let's Generalize

- Many programming languages have another kind of selection statement – the case statement
- Case Statements
  - Are very different in structure and function in different programming languages.
  - You'll see specific examples when we look at examples of actual procedural programs.

Practice Problem

Assume that you've been asked to write a program that calculates an employee's weekly pay. The user enters the pay rate and the hours worked. The program displays regular pay, overtime pay and weekly pay.

Practice Problem

Assume that you've been asked to write a program that determines the letter grade received by a student in a course. The total number of points in the term is 400. Earning 90% of the points gives students an A, 80% is a B, 70% is a C, 60% is a D and less than 60% is an F.