Selection

CS 133N/ CS 161N

Mari Good

Objectives

- Review the programming process and the IPO chart
- Introduce programming control structures
- Introduce algorithms and pseudocode
- Introduce you to C# syntax for
  - if statement
  - Logical operators
  - Relational operators
  - switch statement
- Practice with several examples

Programming is a Process

- You may recall these problem solving steps from the last topic
  - Understand the problem
  - Outline a general solution using an IPO chart
  - Develop an algorithm for solving the problem using pseudocode
  - Test the algorithm for correctness
  - Translate the algorithm into C# syntax
  - Test and debug the C# program

Let's start with an example

- Design and implement a program that calculates gross weekly pay for an employee. The user will enter hours worked and pay rate. An employee will be paid time and one-half for any overtime hours worked. More than 40 hours per week is considered overtime.
- The first step is to understand the problem. Questions?
- The second step is to use an IPO chart to describe WHAT needs to be done, in a very general way, to solve the problem. See the next slide.

IPO Chart

Input
- hoursWorked
- payRate

Processing
- get the hoursWorked
- get the payRate
- calculate the grossPay
- display the grossPay

Output
- grossPay

Example

- Example

  20 hoursWorked * 25 payRate = 500 grossPay

  50 hoursWorked * 20 payRate = 40 * 20 + 10 * 20 * 1.5 = 1100 grosspay
- Notice that this calculation is different under different circumstances.
- That requires a programming “construct” or building block that allows a program to branch or make choices.

Control Structures

- All programming languages have 3 control structures. These are sometimes referred to as structured programming constructs. They are
  - Sequence – in the absence of any other construct, programming statements are executed in order. All of your programs thus far have used sequence.
  - Selection – allows programs to branch or make choices based on a condition. That's the focus of our work in this topic.
  - Repetition – allows programs to execute blocks of code 1 or more times based on a condition. That's the next topic.

Algorithms and Pseudocode

Processing

- get the hoursWorked
- get the payRate
- calculate the grossPay
- display the grossPay

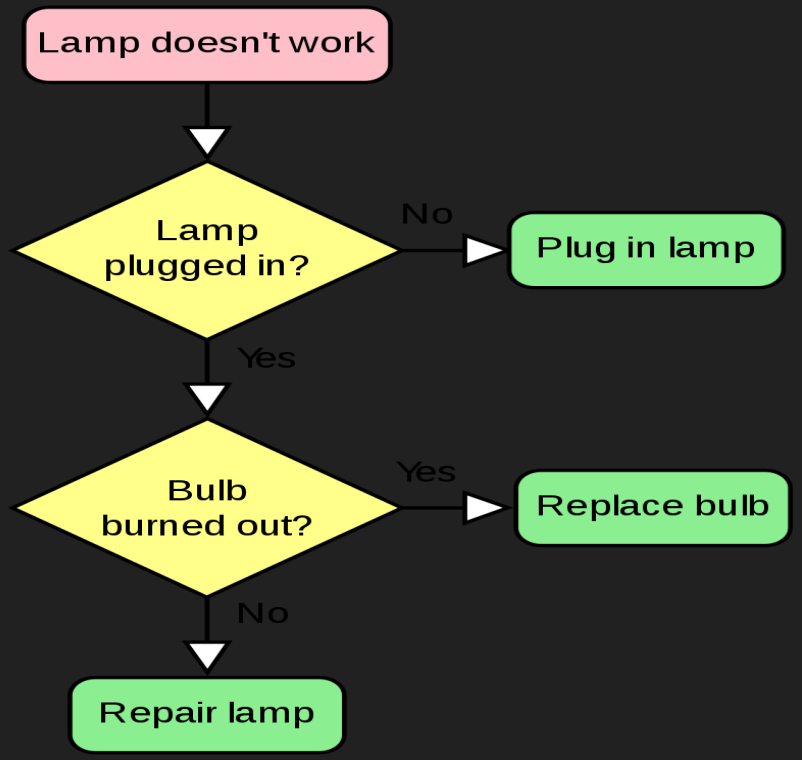
Now that the calculate step is more complicated, we need a tool that can help us describe not just what has to happen but EXACTLY HOW it happens. That's where the algorithm comes in.

Algorithms and Pseudocode

- An algorithm is a step-by-step set of instructions that describes exactly how to complete a specific task.
  - You use an algorithm whenever you follow a recipe, use google maps or a gps, do long division and even tie your shoes.
  - A computer program is just an algorithm, it tells the computer the steps that are required to perform at task, translated into a programming language.
  - Writing algorithms is difficult because human beings do all kinds of things without really thinking about what they're doing. Programming is easy once you've got the algorithm!

Algorithms and Pseudocode

- Programmers express algorithms in one of 2 ways
  - A flowchart is a graphical illustration of an algorithm
  - Pseudocode is an English language outline-like tool for expressing an algorithm. See next slide.



Algorithms and Pseudocode

- Here's an algorithm for the gross pay problem.

This selection structure provides the details for calculate grossPay in our IPO chart.

The if statement is the most common selection statement. It starts with if and ends with end if

```
display instructions
get hoursWorked
get payRate
if hoursWorked <= 40
        grossPay = hoursWorked * payRate
else
        grossPay = 40 * payRate + (hoursWorked - 40) * payRate * 1.5
end if
display grossPay
```

When the condition is true the top block executes. When it's false the bottom block executes

Testing an Algorithm

```
display instructions
get hoursWorked
get payRate
if hoursWorked <= 40
      grossPay = hoursWorked * payRate
else
      grossPay = 40 * payRate + (hoursWorked - 40) *
                  payRate * 1.5
end if
display grossPay
```

hoursWorked	payRate	grossPay
20	25	500

Testing an Algorithm

```
display instructions
get hoursWorked
get payRate
if hoursWorked <= 40
      grossPay = hoursWorked * payRate
else
      grossPay = 40 * payRate + (hoursWorked - 40) *
                  payRate * 1.5
end if
display grossPay
```

hoursWorked	payRate	grossPay
50	20	1100

Another Example

- Design and implement a program that asks the user to enter 3 quiz scores (out of 50), calculates and displays the average score and the average percentage. The program will also display a congratulatory message when the percentage is 90 or above.
- The first step is to understand the problem. Questions?
- The second step is to use an IPO chart to describe WHAT needs to be done, in a very general way, to solve the problem. See the next slide.

IPO Chart

Input
- score1, score2, score3

Processing
- get score1, score2, score3
- calculate averageScore
- calculate averagePercentage
- display averageScore, averagePercentage
- display message when averagePercentage over 90

Output
- averageScore
- averagePercentage
- message

Example

- Example

  45 + 46 + 47 / 3 = 46 averageScore

  46 / 50 * 100 = 92 averagePercentage

  Should show a message

  40 + 42 + 44 / 3 = 42 averageScore

  42 / 50 * 100 = 84 averagePercentage

  Should NOT show a message

Algorithms and Pseudocode

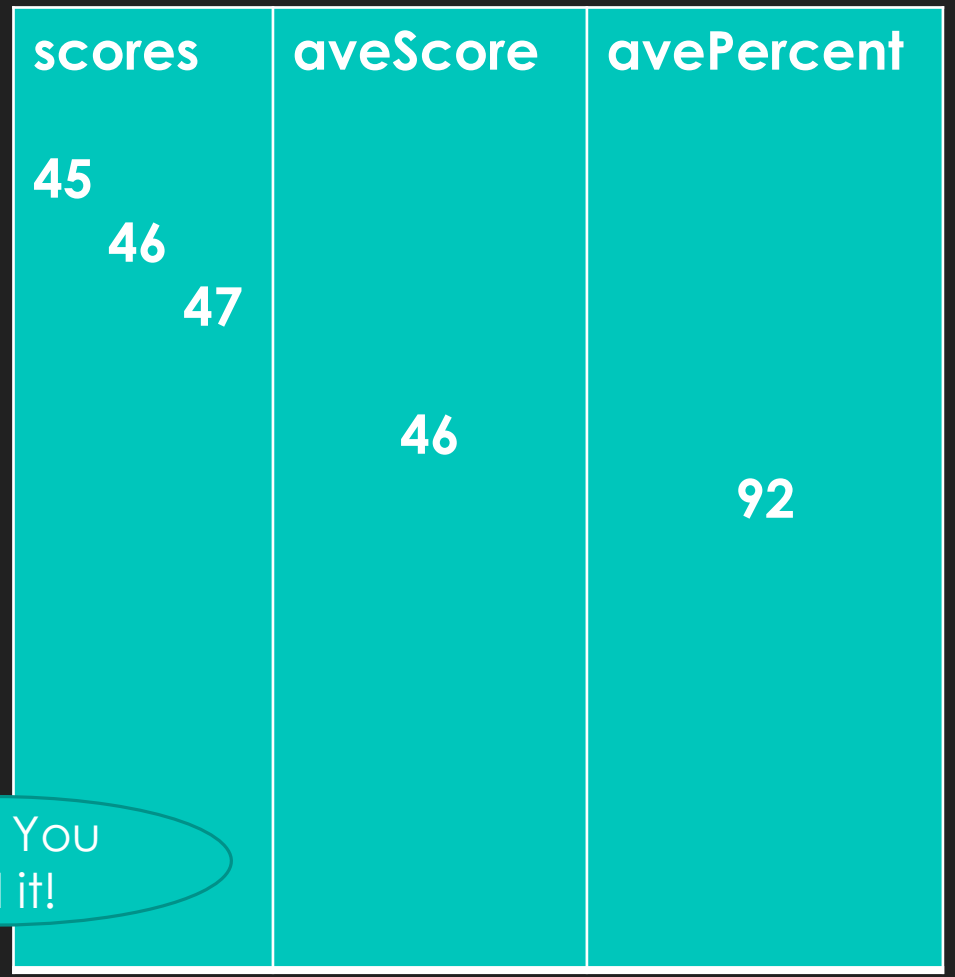
- Let's see if we can write an algorithm

```
display instructions
get score1
get score2
get score3
averageScore = (score1 + score2 + score3)/3
averagePercentage = averageScore / 50 * 100
display averageScore
display averagePercentage
if averagePercentage >= 90
        display congratulatory message
end if
```

Some if statements don't need to do anything when the condition is false. Notice that the else part is omitted.

Testing an Algorithm

```
display instructions
get score1
get score2
get score3
averageScore = (score1 + score2 + score3)/3
averagePercentage = averageScore / 50 * 100
display averageScore
display averagePercentage
if averagePercentage >= 90
      display congratulatory message
end if
```



Our Third Example

- Design and implement a program that asks the user to enter 3 quiz scores (out of 50), calculates and displays the average score and the average percentage as well as a letter grade. >= 90 is an A, between 80 and 90 is a B, between 70 and 80 is a C, below 70 is an F.
- The first step is to understand the problem. Questions?
- The second step is to use an IPO chart to describe WHAT needs to be done, in a very general way, to solve the problem. Do that in small groups.

Algorithms and Pseudocode

```
display instructions
get score1, score2, score3
averageScore = (score1 + score2 + score3)/3
averagePercentage = averageScore / 50 * 100
if averagePercentage >= 90
        letterGrade = A
else if averagePercentage >= 80
        letterGrade = B
else if averagePercentage >= 70
        letterGrade = C
else
        letterGrade = F
end if
display averageScore, averagePercentage, letterGrade
```

Some if statements have multiple branches (not just 1 or 2). Each branch after the first starts with else if and a condition.

This condition could have been written averagePercentage < 90 AND averagePercentage >= 80. And is a logical operator. Or and Not as well.

Your Turn

- Let's look at the 6 problems that are part of lab 3 together. For each problems you should
  - Create an IPO chart
  - Do one or more examples
  - Write an algorithm in pseudocode
  - Test the algorithm
- Next time I'll show you how to translate the algorithms into C# code

Translating an Algorithm into Code

- Each line in your pseudocode will be replaced with one or more statements of C# code.
- You already know how to
  - Declare variables
  - Get input
  - Assign values to variables
  - Display output
- The only new syntax is the if statement

General Syntax for if

```
if (condition)
{
    statements that execute when condition is true
}
else
{
    statements that execute when condition is false
}
```

{ } can be omitted if there's only one statement in the block

Sometimes the else will be omitted. Other times there will be one or more else if clauses in addition.

Conditions

- Include relational operators.
  - >
  - >=
  - <
  - <=
  - ==
  - !=
- Include logical operators.
  - &&
  - ||
  - !

More examples

- Let's look at the first 3 examples we used for practicing pseudocode in dotnetfiddle.net.
- Then we'll use dotnetfiddle.net to do the first 2 problems from the lab together.
  - I'll introduce the switch statement when we do the last problem in dotnetfiddle.
- Finally, there will be time to do the other 4 problems from the lab (you've already done the IPO charts and algorithms) in small groups. I'll help whenever you get stuck.

What's Next

- Repetition and more algorithms in pseudocode
- Syntax
  - while loop
  - for loop
  - do while loop
- Don't forget
  - Reading Quiz 3
  - Programming Quiz 3
  - Lab 3 – 6 problems