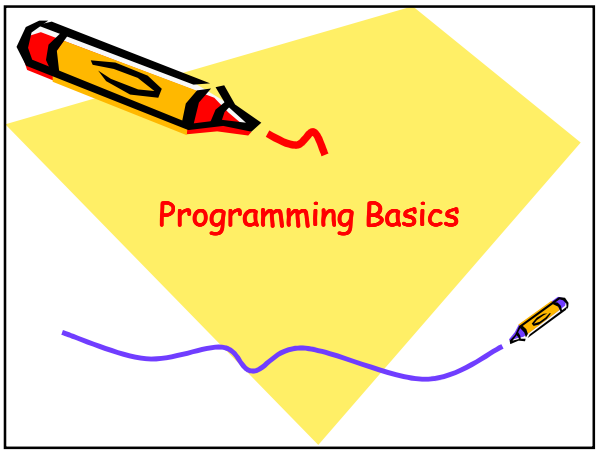


Programming

- Programming – the creation of an ordered set of instructions to solve a problem with a computer.
- The most valuable part of learning to program is learning how to think about arranging the sequence of instructions to solve the problem or carry out the task

Problem Solving

- The process of defining a problem, searching for relevant information and resources about the problem, and of discovering, designing, and evaluating the solutions for further opportunities. Includes:
  - Finding an Answer to a Question
  - Figuring out how to Perform a Task
  - Figure out how to Make Things Work

4 Steps of Problem Solving

- U – Understand the Problem
- D – Devise a Good Plan to Solve
- I – Implement the Plan
- E – Evaluate the Solution

Example: Solving Math Word Problem

- Read the Problem: Understand the description of problem or scenario, identifying the knowns and unkowns
- Decide how to go about solving the problem: Determine what steps need to be taken to reach the solution
- Solve the Problem: Write the solution
- Test the Answer: Make sure the answer is correct

Solving Computing Problems

- In general, when we solve a computing problem we are taking some **inputs**, **processing** (performing some actions on) the inputs, and then **outputting** the solution or results.
- This is the classic view of computer programming – computation as calculation

Step 1 - Understand the Problem

- What is the Problem to be solved? What is the unknown? What is the condition? What is the data? What is needed to solve the problem? What actions need to take place?
- Identify the **inputs** and **outputs**
- Identify the **processes** needed to produce the **outputs** from the given **inputs**
- Draw a figure. Introduce suitable notation.
- Isolate Principle parts of the problem.

Step 2 - Devise a Plan

- Find connections between the knowns and unknowns.
- Simplify: Break the problem into smaller sub-problems
- Design a solution
- Make a plan or list of actions to implement the solution
  - Algorithm / Flowchart / Psuedocode

Step 2 - Devise a Plan (cont.)

- Algorithm
- In programming, algorithm are the set of well defined instruction in sequence to solve a program.
  - To write a computer program, you have to tell the computer, step by step, exactly what you want it to do.
  - The computer then "executes" the program, following each step mechanically, to accomplish the end goal.
  - When you are telling the computer *what* to do, you also get to choose *how* it's going to do it. That's where **computer algorithms** come in

Examples

- **Write an algorithm to add two numbers entered by user.**
- Step 1: Start
- Step 2: Declare variables num1, num2 and sum.
- Step 3: Read values num1 and num2.
- Step 4: Add num1 and num2 and assign the result to sum. sum←num1+num2
- Step 5: Display sum
- Step 6: Stop

Step 2 - Devise a Plan (cont.)

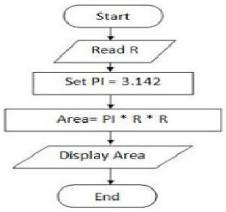
- Pseudocode
  - Written like program code but more "English Like" and doesn't have to conform to language syntax
  - It uses the structural conventions of a normal programming language, but is intended for human reading rather than machine reading.
  - **Pseudocode** typically omits details that are essential for machine understanding of the algorithm, such as variable declarations, system-specific **code** and some subroutines.

Example

```
1. START
2. READ R
3. SET PI=3.142
4. CALCULATE AREA OF CIRCLE
   4.1.    FORMULA:
           Area= PI * R * R
5. DISPLAY Area
6. END
```

Step 2 - Devise a Plan (cont.)

- Flowchart
  - Diagram that visually represents the steps to be performed to arrive at solution.



Step 3 - Implement the Plan

- Implement in a Programming Language
- Carry out the plan checking the preliminary results at each step.
- Code A Little Test A lot

Step 4 - Evaluate the Solution

- Run the Code
- Check results repeatedly and thoroughly
  - Use numerous test cases or data sets
  - Use highly varied test case, including expected as well as and unexpected cases
- Look for new solutions
  - Is there a better, easier, or more efficient solution
- Can other problems be solved using these techniques?

Summary

- U - Read the Problem Statement
  - Identify the inputs, outputs, and processes
- D - Decide how to Solve the Problem
  - Create an Algorithm / Flowchart / Psuedocode
- I - Program the Code
  - Implement in Programming Language
- E - Test the Solution
  - Run the Code using numerous, varied test cases

Practice

Using the first 2 steps, understand and devise a solution to the following problem:

- Determine the week's earnings for an employee from the hourly pay rate and hours worked for the week. Report the gross earnings (including overtime earnings) for the week.
- Definitions:
  - Overtime hours = hours over 40
  - Overtime pay rate = 1.5 * regular pay rate

Questions