Programming Basics – Part 1

CS1A

- Creating Algorithms
- HIPO Charts
- Pseudocode
- Variables
- Flowcharts
- DeskChecks

Solving problems using the Computer as a Tool

- Much like the microscope does not define biology or the test tube does not define chemistry, the computer doesn't define Computer Science.
- The computer is a tool by which Computer Scientists accomplish their goals – to solve problems.

Computer Science

- Is NOT just about coding or hardware or software!
- Computer Science is about PROBLEM SOLVING
- Computer Science is about ***developing algorithms*** to solve complex problems

Control / Logic Structures

All modern programming languages are based on 3 basic control structures

- Sequence
  - Instructions are executed one after another in the order they appear in the program
  - Until another control structure takes precedence
- Selection
  - Based on some condition, either one part of the program is executed or another part is executed
  - The program chooses which part to execute based on the condition
- Repetition
  - Part of the code is executed over and over (repeated)
  - This can be for a set number of times or until a condition is met

Control / Logic Structures

- Tools we use to create Algorithms

WHAT IS AN ALGORITHM?

- An algorithm is a step-by- step definition of a process.
  - It should also be a well-developed, organized approach to solving a complex problem.
- Computer Scientists ask themselves ***four critical questions*** when they evaluate algorithms …

Algorithm Questions

1. Does the algorithm solve the stated problem?
2. Is the algorithm well-defined?
3. Does the algorithm produce an output?
4. Is it efficient?

Developing an Algorithm

1. Identify the Input
2. Identify the Output
3. Identify the Processes
4. Develop a HIPO Chart
5. Develop Pseudo-code or a Flowchart
6. Test our algorithm with a desk check

1. Identify the Inputs

- What data do I need to generate the output?
- How will I get the data?
  - From the user?
- What is the format of the data?

2. Identify the Output

- What output do I need to return to the user?
- How should it be displayed?
- How can I display the data to produce meaningful results?
  - Data vs. Information

3. Identify the Processes

- What do we need to do to produce the output with the given input

4. Develop a HIPO CHART….

A Simple Sequence Problem

Problem Statement:

- Develop an algorithm that calculates the sum and average of two numbers entered by the user

1. Create a HIPO chart
2. Write the Pseudo-code
3. Create a Flowchart
4. Perform a desk check to test our algorithm

All of these techniques help us develop our algorithm without dealing with the complexities of a programming language

Creating our HIPO Chart

Hierarchical and Input, Processing Output Chart

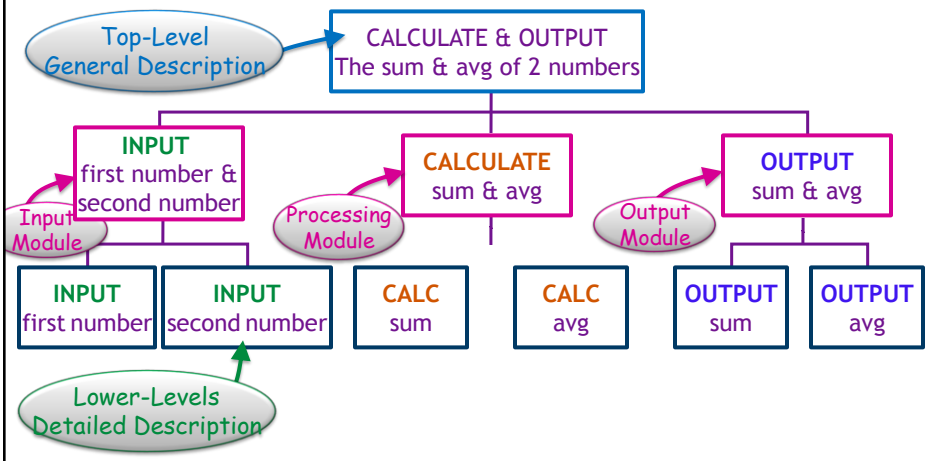
- Shows the overall structure of the program
- Shows the relationship between different components/modules in the system
- Top-level is general
- 2nd level breaks it down into
  → Input, Processing & Output
- Each successive level is more detailed

First we need to ask a few questions:

- What do we need as input?
- What do we need to output?
- What processing do we need to do to go from the input to the output

- How can we break out problem into more manageable sub problems

HIPO CHART



Top-Level : High-level or general description

2nd-Level : Breaks it down into Input, Processing and Output Modules

Bottom-Level: Detailed Solution → for now detail to 1 instruction

Exercise #1

Design the algorithm for a program that calculates the total of a retail sale.

The program should ask the user for the following:

- the retail price of the item being purchased and
- the sales tax rate.

Once the information has been entered the program should calculate and display the following:

- The sales tax for the purchase and
- the total sale.

Draw the HIPO chart for this problem.

Some things to think about

- What is our input?
- What are our output?
- What do we need to calculate (processing)?

HIPO CHART

Writing Pseudo-code

- Terse English description of a algorithm
  - Can be easily converted into any programming language
  - Each sentence starts with a command
  - Uses keywords such as:
  - PROMPT, READ, INPUT, ASSIGN, CALCULATE, OUTPUT

Example:

Calculate sum and average of two numbers:

Should Be Capitalized

BEGIN PROGRAM

PROMPT the user for the first input
READ the first input
PROMPT the user for the second input
READ the second input

} these can be combined into just **INPUT**→

} It is implied that the user input will be read after a prompt

CALCULATE the sum
CALCULATE the average
OUTPUT the sum
OUTPUT the average

END PROGRAM

Exercise #1

Design the algorithm for a program that calculates the total of a retail sale.

The program should ask the user for the following:

- the retail price of the item being purchased and
- the sales tax rate.

Once the information has been entered the program should calculate and display the following:

- The sales tax for the purchase and
- the total sale.

Write the Pseudo-code for this problem.

Pseudocode

Calculate the sales tax and total purchase price:

Before we move on…

- We need to think about the values that we want the computer to store as the program executes
- For our example program…
  - We need to store into memory the two numbers we are using as input (num1 & num2)
  - We also will want to store the sum and the avg of these numbers
  - … those memory locations are called variables

Variable:
A place in memory we use to store data. Variables store data that can change during program execution

Variables

- We don't know exactly where the data is stored
  - But we don't need to…
  - We reference the memory locations with names that we determine
    - The OS keeps track of the actual address (like file names)
  - Naming memory locations is called symbolic addressing
- Variables are essential to programming because we want to be able to use data that we store to do calculations

Variable Naming Rules

We can't just name them anything…

- Required rules → what the compiler cares about
  - Can have letters, numbers, and underscore (_)
  - Variables must begin with a letter
  - They are case-sensitive
    - 'A' ≠ 'a'
  - Can't have spaces
  - Can't have special characters

Can't be keywords

Keywords
→ Words that have special meaning in C++

Variables – Naming Rules

- Stylistic rules → what we care about
  - Use meaningful names
  - For 1 word → Keep it lowercase
  - For 2 or more words → 1st word is lowercase
  - Capitalize the 1st char of each subsequent word

Variables - Examples

GOOD
sum
total
countGrades
payRate

BAD
SUM
Total grades
3num
%payrate
pay rate
Payrate
PAYRATE

Assignment Statements

An assignment statement is one way that put data into the memory location referenced by a variable

Syntax

variable = *expression*;

```
sum = num1 + num2;
```

Assignment Operator

This assigns the value of num1 + num2
to the memory location referenced by sum

Assignment Statement Examples

GOOD

```
ageOne = 15
ageTwo = 23
averageAge = (ageOne + ageTwo) / 2.0
answer = 'y'
sum = num1 + num2
```

BAD

```
10 + sum = sum
23 = sum + 5
sum + 5 = sum
```

Types of Calculations

We can use basic math calculations in programming

Name	Symbol	Example
addition	+	sum = 4 + 7
subtraction	-	difference = 18.55 - 14.21
multiplication	*	product = 5 * 3.5
division	/	quotient = 14 / 3
modulo ("mod")	%	remainder = 10 % 6

Back to flowcharts

Flowcharts

- A detailed picture (or description) of an algorithm
- Represents the flow of the program
  - The sequence in which instructions are executed
- Drawn from Top to Bottom → shows the exact order that the instructions will execute
  - The top symbol represents the first instruction executed
  - The bottom symbol represents the last instruction executed
- Uses special symbols to represent different program statements

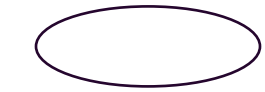
Using Variables in a Flowchart

- Names of the variables in a flowchart must match exactly the actual names of variables use in a program
- Thus… we must use appropriate names
- For the sum & average problem we will name ou variables as follows:
  - num1, num2, sum and avg

Note:
C++ is CASE SENSITIVE!
In other words it does not consider 'A' to be the same as 'a'

Flowcharting Symbols

We will start with the 3 flowcharting symbols



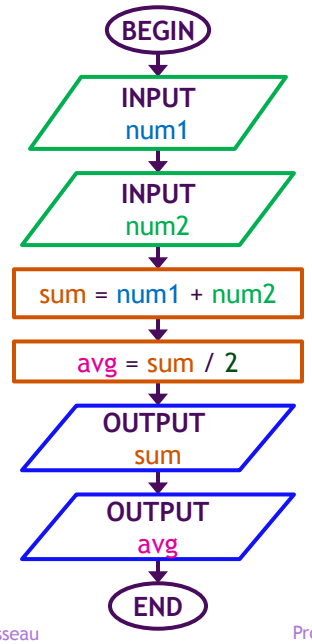
Begin/End of Block

Process

Input/Output

I will introduce more symbols as we go along

Sum & Average Flowchart



- **Note**: The flowchart begins with the process outlined on the bottom left of the HIPO chart
- i.e. the lowest level where the specific steps have been outlined

Exercise

Design the algorithm for a program that calculates the total of a retail sale.

The program should ask the user for the following:

- the retail price of the item being purchased and
- the sales tax rate.

Once the information has been entered the program should calculate and display the following:

- The sales tax for the purchase and
- the total sale.

Draw the flowchart for this algorithm.

Some things to think about

- What is our input?
  - retail price of the item being purchased
  - sales tax rate.
- What is our output?
  - Sales tax
  - Total sale
- What do we need to calculate?
  - Sales tax = retail price * sales tax rate
  - Total sale = retail price + sales tax
- What should we name our variables?

SOMETHING THAT MAKES SENSE!

Flowchart Exercise

VARIABLE LIST

INPUT:

OUTPUT:

PROCESSING:

Exercise #2

Draw a flowchart to match the following pseudocode.

```
BEGIN PROGRAM
    ASSIGN num1 = 5
    ASSIGN num2 = 10
    CALCULATE num2 = num2 + 10
    CALCULATE num3 = num1 * 2
    CALCULATE num 2 = num2 - num1
    OUTPUT num1
    OUTPUT num2
    OUTPUT num3
END PROGRAM
```

- What is our input?
- What is our output?
- What are our variables?

Exercise #2

VARIABLE LIST

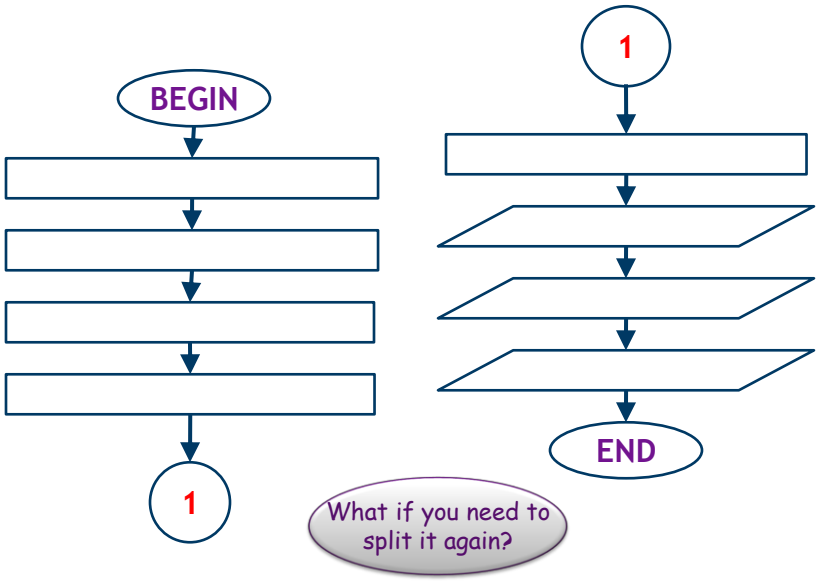
INPUT:

OUTPUT:

PROCESSING:

37

What if you run out of space?

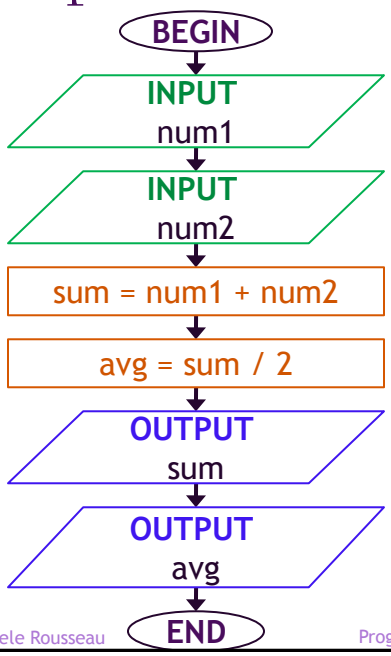


Desk Checks

- We perform Desk checks to test our algorithm
- The idea is to *mimic the computer* and track
  - the values that are stored in memory &
  - the output
- We do this by writing down all the variables and tracing through the program

Example Desk Check

Let's perform a desk check on the sum and avg algorithm Using the values 3 & 7



INPUT: 3 (num1)
7 (num2)

EXPECTED OUTPUT: 10 (sum)
5 (avg)

num1	num2	sum	avg
3	7	10	5

OUTPUT
10 (sum)
5 (avg)

Exercise Desk Check

- Trace the steps in your flowchart from the previous exercise and show the output produced by this program.

Desk Check

num1 num2 num3

Output

Exercise #4: Desk Check

- Trace the steps in your flowchart from the sales tax and retail price problem.
- INPUT: $300.00 & 10%
- EXPECTED OUTPUT:

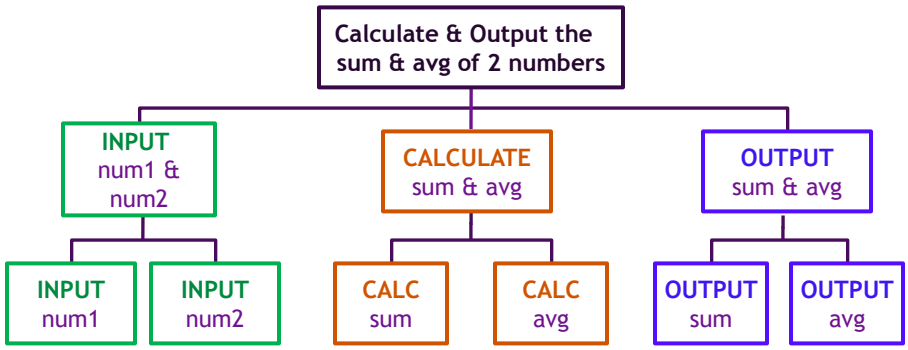
Exercise Desk Check

Let's perform a desk check on the sum and avg algorithm Using the values 300.00 & 0.10

EXAMPLE UPDATED

- The next 3 slides show the final versions of the topic example for the HIPO chart, pseudocode and flowchart
- The 4th slide provides a detailed explanation of how to conduct a desk check
- The 5th slide will show the final desk check
- Notice that all the diagrams and the pseudocode use proper variable names

HIPO CHART



INPUT	OUTPUT	PROCESSING
num1	sum	
num2	avg	

Pseudo-code

```
BEGIN PROGRAM
    INPUT num1
    INPUT num2
    CALCULATE  sum = num1 + num2
    CALCULATE  avg = sum / 2
    OUTPUT sum
    OUTPUT average
END PROGRAM
```

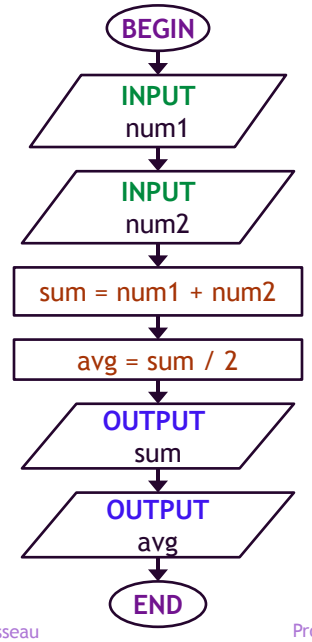
VARIABLE LIST

INPUT:
- num1
- num2

OUTPUT:
- sum
- avg

PROCESSING:
- *<no additional>*

Sum & Average Flowchart



VARIABLE LIST

INPUT:

num1

num2

OUTPUT:

sum

avg

PROCESSING:

<no additional>

Example Desk Check

For a desk check we are testing our algorithm

- we start by selecting test data - for this example we will test 2 sets of data. Now every time we execute a PROMPT in pseudocode or an INPUT in a flowchart we store one of our input values into the variable specified. We start read our input values in from left to right.
  - Test set #1: 3 & 7
  - Test set #2: 9 & 5
- Next, we want to do is figure out what the output should be and then walk through our algorithm (flowchart or pseudocode) one step at a time and see if it produces the same results.
  - Test set #1:

    INPUT VALUES: 3, 7

    EXPECTED OUTPUT: sum: 10, avg: 5
  - Test set #2:

    INPUT VALUES: 9, 5

    EXPECTED OUTPUT: sum: 14, avg: 7
- Now we step through our algorithm and trace what will be stored in each of our variables AND we track what will be output
- Finally, we confirm that our output from our desk check matches our EXPECTED OUTPUT

Example Desk Check

TEST CASE #1:

INPUT: 3 (sum)
7 (avg)

EXPECTED OUTPUT: 10 (num1)
5 (num2)

num1	num2	sum	avg
3	7	10	5

OUTPUT
10 (sum)
5 (avg)

TEST CASE #2:

INPUT: 9 (num1)
5 (num2)

EXPECTED OUTPUT: 14 (sum)
7 (avg)

num1	num2	sum	avg
9	5	14	7

OUTPUT
14 (sum)
7 (avg)