

Assignment xx Algorithmic Design Document

Make a copy before you begin (File -> Make a copy). Add the Assignment # above and complete the sections below BEFORE you begin to code and submit with your Assignment to D2L (File -> Download -> PDF). The sections will expand as you type.

zyBooks

Add your zyBooks screenshots for the % and assigned zyLabs completions below. Required percentages: all assigned zyLabs, Challenge Activity with at least 70%, and Participation Activity with at least 80%.

zyLabs, Challenge, and Participation % Screenshot:

8. Structs

100% 100% 100% ▼

Assigned zyLabs completion Screenshot:

8.6 Separate files for structs

100% 100% ▼

8.7 LAB: Vending machine

100% ▼

Assignment

Program description:

This program will allow you to enter two numbers, and show you various calculated values from them.

Before you begin coding, **you must first plan out the logic** and think about what data you will use to test your program for correctness. All programmers plan before coding - this saves a lot of time and frustration! Use the steps below to identify the inputs and outputs, calculations, and steps needed to solve the problem.

Algorithmic design:

- Identify all of the user input. What are the data types of the inputs? Define the input variables.

Input a double or int for num 1,
Input a double or inf for num2

b. Describe the program output. What is displayed to the user? What are the data types of the output? Define the output variables.

This program will display the value of the calculations after running various arithmetic operations on them. We show:

Initial value;
Value after adding num1;
Value after mult. x 3;
Value after subtracting num2;
Value after dividing by 2;
Value after clearing back to 0.0;

c. What calculations do you need to do to transform inputs into outputs? List all formulas needed, if applicable. If there are no calculations needed, state there are no calculations for this algorithm.

We run the following arithmetic:

Value + num1;
Value * 3;
Value – num2;
Value / 2;
Value = 0.0

d. Design the logic of your program using pseudocode or flowcharts. See pseudocode syntax at the bottom of this document. Here is where you would use conditionals, loops, functions or array constructs (if applicable) and list the steps in transforming inputs into outputs. Walk through your logic steps with the test data from the assignment document.

---- Main.c ----

INCLUDE Calculator.h

DECLARE calc = InitCalculation(calc) //Initializes as 0

DECLARE Double num1

DECLARE Double num2

DISPALY "Welcome to my program!"

DISPLAY "Enter 2 nums"

INPUT num1

INPUT num2

DISPLAY "The starting value" + CALL GetValue(calc)

SET calc = CALL Add(num1, Calc)

DISPLAY "The new value is" + CALL GetValue(calc)

```
SET calc = CALL Multiply(3, num1)
DISPLAY "The new value is" + CALL GetValue(calc)
```

```
SET calc = CALL Subtract(num2, Calc)
DISPLAY "The new value is" + CALL GetValue(calc)
```

```
SET calc = CALL Divide(2, Calc)
DISPLAY "The new value is" + CALL GetValue(calc)
```

```
SET calc = CALL Clear(Calc)
DISPLAY "The new value is" + CALL GetValue(calc)
```

```
DISPLAY "Thanks for using this :)"
```

```
RETURN 0
```

```
---- Calculator.c ----
```

```
INCLUDE Calculator.h
```

```
FUNCTION Calculator InitCalculator(Calculator, aCalc)
- - SET aCalc.value = 0.0;
```

```
- - RETURN aCalc
```

```
FUNCTION Calculator Add(Double val, Calculator aCalc)
- - SET aCalc.vale = aCalc.value + val
```

```
- - RETURN aCalc
```

```
FUNCTION Calculator Subtract(Double val, Calculator aCalc)
- - SET aCalc.vale = aCalc.value - val
```

```
- - RETURN aCalc
```

```
FUNCTION Calculator Multiply(Double val, Calculator aCalc)
- - SET aCalc.vale = aCalc.value * val
```

```
- - RETURN aCalc
```

```
FUNCTION Calculator Divide(Double val, Calculator aCalc)
- - SET aCalc.vale = aCalc.value / val
```

```
- - RETURN aCalc
```

```
FUNCTION Calculator Clear(Calculator aCalc)
- - SET aCalc.vale = 0.0
```

```
- - RETURN aCalc
```

```
FUNCTION Calculator GetValue(Calculator aCalc)
```

```
- - RETURN aCalc
```

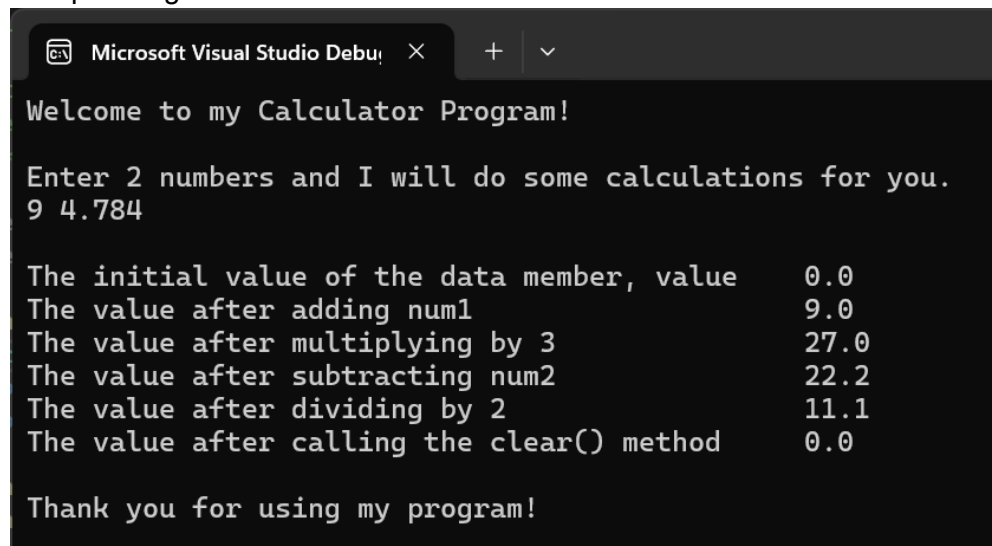
```
---- Calculator.h ----
```

```
DEFINE Struct Calculator_struct as Calculator
```

```
DECLARE Double value
```

- e. Include 2 Sample Program Runs for your program using your own set of data. This data set must be different from my Sample Runs in the Assignment document. This process is similar to Unit Testing and will help you test your program better.

Sample Program Run 1:



```
Microsoft Visual Studio Debug Console
Welcome to my Calculator Program!
Enter 2 numbers and I will do some calculations for you.
9 4.784

The initial value of the data member, value      0.0
The value after adding num1                       9.0
The value after multiplying by 3                  27.0
The value after subtracting num2                  22.2
The value after dividing by 2                     11.1
The value after calling the clear() method        0.0

Thank you for using my program!
```

Sample Program Run 2:

```
Microsoft Visual Studio Debug Console
Welcome to my Calculator Program!

Enter 2 numbers and I will do some calculations for you.
10.0 5.0

The initial value of the data member, value      0.0
The value after adding num1                      10.0
The value after multiplying by 3                  30.0
The value after subtracting num2                  25.0
The value after dividing by 2                     12.5
The value after calling the clear() method        0.0

Thank you for using my program!
```

Pseudocode Syntax

Think about each step in your algorithm as an action and use the verbs below:

| To do this: | Use this verb: | Example: |
|--|--|---|
| Create a variable | DECLARE | DECLARE integer num_dogs |
| Print to the console window | DISPLAY | DISPLAY "Hello!" |
| Read input from the user into a variable | INPUT | INPUT num_dogs |
| Update the contents of a variable | SET | SET num_dogs = num_dogs + 1 |
| Conditionals | | |
| Use a single alternative conditional | IF <i>condition</i> THEN <i>statement</i> <i>statement</i> END IF | IF num_dogs > 10 THEN DISPLAY "That is a lot of dogs!" END IF |
| Use a dual alternative conditional | IF <i>condition</i> THEN <i>statement</i> <i>statement</i> ELSE <i>statement</i> <i>statement</i> END IF | IF num_dogs > 10 THEN DISPLAY "You have more than 10 dogs!" ELSE DISPLAY "You have ten or fewer dogs!" END IF |
| Use a switch/case statement | SELECT <i>variable or expression</i> CASE <i>value_1</i> : <i>statement</i> <i>statement</i> CASE <i>value_2</i> : | SELECT num_dogs CASE 0: DISPLAY "No dogs!" CASE 1: DISPLAY "One dog.." CASE 2: DISPLAY "Two dogs.." CASE 3: DISPLAY "Three dogs.." DEFAULT: DISPLAY "Lots of |

| | | |
|--|---|--|
| | <i>statement</i> <i>statement</i> CASE <i>value_2</i> : <i>statement</i> <i>statement</i> DEFAULT: <i>statement</i> <i>statement</i> END SELECT | dogs!" END SELECT |
| Loops | | |
| Loop while a condition is true - the loop body will execute 0 or more times. | WHILE <i>condition</i> <i>statement</i> <i>statement</i> END WHILE | SET num_dogs = 1 WHILE num_dogs < 10 DISPLAY num_dogs, " dogs!" SET num_dogs = num_dogs + 1 END WHILE |
| Loop while a condition is true - the loop body will execute 1 or more times. | DO <i>statement</i> <i>statement</i> WHILE <i>condition</i> | SET num_dogs = 1 DO DISPLAY num_dogs, " dogs!" SET num_dogs = num_dogs + 1 WHILE num_dogs < 10 |
| Loop a specific number of times. | FOR <i>counter</i> = <i>start</i> TO <i>end</i> <i>statement</i> <i>statement</i> END FOR | FOR count = 1 TO 10 DISPLAY num_dogs, " dogs!" END FOR |
| Functions | | |
| Create a function | FUNCTION <i>return_type</i> <i>name (parameters)</i> <i>statement</i> <i>statement</i> END FUNCTION | FUNCTION Integer add(Integer num1, Integer num2) DECLARE Integer sum SET sum = num1 + num2 RETURN sum END FUNCTION |
| Call a function | CALL <i>function_name</i> | CALL add(2, 3) |
| Return data from a function | RETURN <i>value</i> | RETURN 2 + 3 |