

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:

10. Bitwise Operators & File Input / Output



0%



100% ^

Assigned zyLabs completion Screenshot:

10.7 LAB: Multiply using Bitwise Operator



0% v

Assignment

Program description:

This program will read a series of data rows from a file, and added the sum of the numbers from that row. The first number in the file will dictate how many rows of data you would like it to read

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.

File numbers.txt, int. Only require a file containing various integers. The first number will be the total number of rows to read

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

File output.txt, it. This program will output a series of numbers based on calculations from the input data. This is printed in to the file as each calculation is completed

- 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.

Int num1 + Int num2

- 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.

```
DECLARE File* inFile = NULL
DECLARE File* outFile = NULL
DECLARE Int fileData1[250]
DECLARE Int fileData2[250]
DECLARE Int arrSize
DECLARE Int i
DECLARE Int calcResult

DISPLAY "Welcome to my program!"
DISPLAY Data processing message

SET inFile = FileOpen numbers.txt arg Read
SET outFile = FileOpen output.txt arg Write

IF inFile == NULL or outFile == NULL
  - - DISPLAY "This file can not be opened"

ELSE
  - - WRITE "" to outFile
  - - READ inFile; SET arrSize = first int in inFile
  - -
  - - FOR i = 0 TO i < arrSize
    - - - SET fileData1[i] = inFile next Int; SET fileData2[i] = inFile next Int
    - -
    - - FOR i = 0 TO i < arrSize
      - - - SET calcResult = fileData1[i] + fileData2[i]
      - - - WRITE calcResult TO outFile
      - -
      - - FOR i = 0 TO i < arrSize
        - - - SET calcResult = fileData1[i] + fileData2[i]
        - - - DISPLAY calcResult
        - -
        - - DISPLAY "Thank you for using my program!"

FILE Close inFile
```

FILE Close outFile

RETURN 0

- 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:

```
int i;
int calcResult;

printf("Welcome to my Sum of Numbers Program!\n");
printf("\nReading from text file...\n");

// Check inFile exist
inFile = fopen("numbers.txt", "r");
// Check outFile exist
outFile = fopen("output.txt", "w");

if (inFile == NULL || outFile == NULL) {
    printf("\nThis file could not be opened.\n");
}
else {
    // Clear current data in output
    fprintf(outFile, "");

    // Get array size
    fscanf(inFile, "%d", &arrSize);

    // Construct data arrays
    for (i = 0; i < arrSize; i++) {
        fscanf(inFile, "%d %d", &fileData1[i], &fileData2[i]);
    }

    // Process data calculations; output to file in series
    for (i = 0; i < arrSize; i++) {
        calcResult = fileData1[i] + fileData2[i];
        fprintf(outFile, "%d ", calcResult);
    }

    // Repeat loop for user display output
    printf("\nHere is your output also written to output.txt\n\n");
    for (i = 0; i < arrSize; i++) {
```

Sample Program Run 2:

```
1 // *** Remove before submission ***
2 #define _CRT_SECURE_NO_WARNINGS
3 #define _CRT_SECURE_NO_DEPRECATED
4 #define _CRT_NONSTDC_NO_DEPRECATED
5 // *** Remove before submission ***
6
7 // NOTE: This template is to be used for partner practice ONLY! You must
8 // use the required Algorithmic Design Document for all Assignments.
9 //=====
10 # Author: Kyle Noyes
11 # Assignment: 10
12 # Date: March 17th, 2024
13 # Description: This program will read a file of numbers and sum the numbers
14 # Input: numbers.txt
15 # Output: output.txt, various state messages for user
16 //=====
17
18 #include <stdio.h>
19 #include <stdlib.h>
20
21
22 int main(void) {
23     FILE* inFile = NULL;
24     FILE* outFile = NULL;
25     int fileData1[250];
26     int fileData2[250];
27     int arrSize;
28     int i;
29     int calcResult;
30
31     printf("Welcome to my Sum of Numbers Program!\n");
32     printf("\nReading from text file...\n");
33
34     // Check inFile exist
35     inFile = fopen("numbers.txt", "r");
```

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> : <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	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 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	<pre> FUNCTION Integer add(Integer num1, Integer num2) DECLARE Integer sum SET sum = num1 + num2 RETURN sum END FUNCTION </pre>
Call a function	CALL <i>function_name</i>	CALL add(2, 3)
Return data from a function	RETURN <i>value</i>	RETURN 2 + 3