CS 105 – Lab 6 – Arrays and File I/O

We can write more interesting programs if we are able to process more data. Today, you will work with an array, a data type designed to hold many cells of the same type, instead of just a single value. Later, you will also practice some file input & output, so that your program can read and write large amounts of data.

Insert your USB drive, and log into the computer. Start a couple of Terminal processes. It is good to have one terminal window for editing, and another for running programs. Use the `cd` command to navigate to the folder containing your USB drive. Please ask for help if you don't remember how to do this. With the `ls` command, you should see folders from earlier labs.

Create a new folder called lab6 with this command: `mkdir lab6`

Make sure that you do not have files stored on the computer's hard drive. To check, run the command `ls ~` and verify that you do not see any lab folders or Pascal programs. If you do, this means you may need to move them onto your USB drive. Please ask the instructor or lab aide if you are not sure.

Part 1 – An array

Create a new program called testarray.pas. It will feature an array of 5 integers. It will ask the user to enter 5 values, one at a time, and insert them into the cells of the array. Then, it will perform some simple operations on this data. The program will determine

- The largest number
- The smallest number
- The sum of all the elements
- The average of all the elements
- If any of the numbers is 0
- How many even numbers are in the array

```
(* testarray.pas – Analyze the contents of an array. *)
program testarray;

var
  a : array[1..5] of integer;
  i : integer;
  average : real;
  foundzero : boolean;
  (* You will need to declare more variables above. *)
```

```
begin
  (* input *)
  for i := 1 to 5 do
    begin
      write('Please enter value for cell ', i, ' -->  ');
      readln(a[i]);
    end;

  (* Task #1 - Print out the array, all on one line.
   * Insert a loop between these two statements:
   *)
  writeln('Your array contains:  ');

  writeln();

  (* Task #2 - A loop that finds the sum, max, min,
   * existence of zero, and counts how many are even.
   * Insert code inside the body of the loop.
   *)
  max := 0;
  min := 0;
  sum := 0;
  foundzero := false;
  counteven := 0;
  for i := 1 to length(a) do
    begin

    end;

  (* Output - You don't need to change this code.  *)
  writeln('The largest number is ', max);
  writeln('The smallest number is ', min);
  writeln('The sum is ', sum);

  average := sum / length(a);
  writeln('The average is ', average : 4 : 1);

  if foundzero then
    writeln('The array contains the number zero.')
  else
    writeln('There is no zero in the array.');

  writeln('I found ', counteven, ' even numbers.');
end.
```

When you are done editing your program, compile and run it. If anything is missing or incorrect, please correct that now before continuing. What mistake(s) did you find in the code provided above?

Run the program a few times, using several sets of input. Record the results in the following table. Choose your input so that the answers in the last 2 columns below differ, and have the opportunity to be zero or false. Also choose input to show that you fixed the bug in the original code I gave you. √

| 5 input values for array | Largest | Smallest | Average | 0 in array? | # evens |
|---|---|---|---|---|---|
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

Part 2 – File output

The reason why we want to do file output is that we often have too much output to fit on the screen. The idea is simple. Instead of printing to the screen, the program will create a new file and put all of its output into the file. Then, when the program is done, you can inspect the output later. By keeping the output in a file, we enjoy the benefit of retaining the output indefinitely without having to run the program again.

Let's create a new Pascal program called bigcount.pas:

```
(* bigcount.pas – Print a lot of numbers to a file. *)

var
  outfile : textfile;
  i : integer;
```

```
begin
  assign(outfile, 'numbers.txt');
  rewrite(outfile);

  for i := 1 to 100 do
    writeln(outfile, i);

  close(outfile);
end.
```

Compile and run the program. This might seem strange at first, but we see no output on the screen. Where did the output go? Use the ls command to see that new file called numbers.txt listed. To display the contents of this output file, enter the command `more numbers.txt`

1. Describe the contents of numbers.txt.

2. It is possible to use both interactive as well as file I/O at the same time. Some people like to see their file output also go to the screen so that they can observe the progress of the program. How would you modify your program so that it also prints to the screen the same data that is being written to the file?

3. Let's try a different experiment with file output. Copy your program bigcount.pas to a new source file called bigcount2.pas. Modify the program so that its output file is called numbers2.txt.

   Next, modify the output so that each line of output is not just a number, but an entire sentence such as:

   ```
   The next number is 1
   The next number is 2
   The next number is 3
   ```

   And so on. √

Part 3 – File input

It's also possible for a computer program to need to scan a lot of input. Interactive input is inconvenient, because the user would have to type in the input each time the program is run.

If you recently ran the program bigcount.pas, you will notice that it produced a file called numbers.txt. Let's use this file as input to a new program: one that reads numbers from the file and adds them up.

Edit a new Pascal program called add.pas. The source code is given below. But you need to complete two assignment statements. Some missing information can be found in the File I/O section in your Pascal tutorial.

```
(* add.pas – Read numbers from an input file, one per line,
 * and add them up.
 *)
program add;

var
  infile : textfile;
  sum, value : integer;
  line : string;

begin
  assign(infile, 'numbers.txt');
  reset(infile);

  sum := 0;
  while not eof(infile) do
    begin
      readln(infile, line);
      value := (* Need to convert the line to a number *)
      sum :=   (* Need to accumulate the sum *)
    end;

  writeln('The sum of the numbers in the file is ', sum);

  close(infile);
end.
```

1. How did you complete the two missing assignment statements? What other detail did you need to enter in your program that was not given in the code above?

2. Run the program. What is the output?

3. Modify bigcount.pas so that it will print the numbers 1 through 300 inclusive. Run `bigcount` so that it updates the file numbers.txt. Now, run the `add` program again. What is the output? Why is the output (not) what one might expect?

4. Explain what is necessary to fix the program so that it will print the correct output. Make the changes, and re-run your program. ✓