CS 105 Lab 2

The purpose of our lab today is to explore Pascal operations and assignment statements, and then begin to type in complete programs. You may find these programs helpful in preparing for future labs and studying for exams.

*Please have your answers checked by the instructor or lab aide periodically during the lab. Logical stopping points for doing this would be right after you have completed questions 5, 10 and 13.*

The software that we will use to program in Pascal is called Free Pascal, which is abbreviated as fpc (Free Pascal Compiler) on your computer. We will open an editor window to type in full programs to save.

How to use Free Pascal on the Mac

A. Insert your USB drive into the back of the Mac.
B. Log in, using your usual Furman computer username and password.
C. Start the **Terminal** program. We will be doing today's work via the terminal. When the terminal appears on the screen, you may want to enlarge it.
D. We are going to have the file system navigate to your USB drive. Type in this command:
   `cd /Volumes/USB\ DISK`
   and then hit enter.
E. Now that the file system is looking at your USB drive, we are ready to create any files and folders we need. Let's create a folder called lab2. To do this, enter this command:
   `mkdir lab2`
   and then hit enter. From now on, just note that you need to hit enter after each terminal command.
F. A useful file-system command is `ls`, which means to list the contents. Enter the `ls` command, and verify that `lab2` is shown.
G. Now, we are going to navigate into your lab2 folder so we can do all of our work in there. Enter this command:
   `cd lab2`
   Now you are ready to do some Pascal!

---

1. First, let's look at how we can display numbers in Pascal. In the following table, you will see a Pascal program on the left. Create a new Pascal program and type in this code. Here is how to accomplish this:
   - The first step is to edit (type) the program. The name of the program will be format.pas. Therefore, enter the following command to the terminal:
     `nano format.pas`
     "Nano" is the name of a text editor.
   - Inside nano, type in the code you see in the table below. When you are finished, you need to save, and then exit nano. To save, use the keystroke sequence Control-O. To exit, hit Control-X.

- Now that you have exited nano, you are back at the terminal command prompt. The second step is to compile. Enter this command to compile your program:
  `fpc format.pas`
  "fpc" is the name of the compiler, Free Pascal Compiler. If the compiler gives you any error messages, it will tell you what (approximate) line number the error occurs on. Go back to nano and edit the program again, and try to compile again. After the program successfully compiles, you can enter the "ls" command again to see that the compiler has created an executable file.
- After having edited and compiled the program, you are ready to run the program! To run a program called format, type in this command:
  `./format`
  Anytime you want to run one of your programs, it will be necessary to precede its name with "./".

When the program runs, write the output you see in the right side of the table below.

| Program | Output |
|---|---|
| `program format;`<br>`begin`<br>`  writeln(16);`<br>`  writeln(16 : 5);`<br>`  writeln(16 : 10);`<br>`  writeln(12.34);`<br>`  writeln(12.34 : 5 : 1);`<br>`  writeln(12.34 : 10 : 4);`<br>`end.` | |

We use the colons in a writeln statement to control how the numbers are printed.

a. Based on what you see in the output, how does the formatting affect how the 16 is displayed?

b. How does the formatting affect how the number 12.34 is displayed? Why do we need to specify 2 format parameters?

2. Let's look at how the mod operator works. Create a new Pascal program called expr1.pas. Use the edit/compile/run procedure described earlier in step #1. When your program runs, write down the output you see in this table.

| Program | Output |
|---|---|
| `program expr1;`<br>`begin`<br>`  writeln(16 mod 9);`<br>`  writeln(25 mod 9);`<br>`  writeln(2020 mod 4);`<br>`  writeln(2021 mod 4);`<br>`  writeln(2022 mod 4);`<br>`  writeln(2023 mod 4);`<br>`  writeln(2024 mod 4);`<br>`end.` | |

a. Are any of the results equal? Show why they should be.

b. Explain in your own words what "mod" means.

c. If x and y are integer values, sometimes x mod y equals zero. What would this say about the relationship between these two numbers?

d. What is the maximum value of x mod y ? (Hint: Express your answer in terms of y.)

e. How can the mod operator be used to tell if a number representing a year is a leap year?

3. Let's look at the order of operations. Type in the following program (expr2.pas), compile, and run it. Write down the output you see. If any result is a real number, it is sufficient to write it to the nearest thousandth.

| Program | Output |
|---|---|
| `program expr2;`<br>`begin`<br>`  writeln(3 + 4 * 2);`<br>`  writeln(10 – 3 – 3);`<br>`  writeln(48 / 6 / 3);`<br>`end.` | |

   a. When evaluating an expression containing two subtractions in a row, or two divisions in a row, do we evaluate from left to right, or right to left? How can you tell?

   b. When an expression features + and *, which operator is performed first? How can you tell?

4. Let's look at / and div. The following program is called expr3.pas. Edit, compile, and run this program as you have with the previous programs. When running the program, if you see a real number as an answer, modify the source code so that it will print just 3 decimal places instead of scientific notation. Compile and run the program again, and write the results in the right side of the table.

| Program | Output |
|---|---|
| `program expr3;`<br>`begin`<br>`  writeln(15 / 7);`<br>`  writeln(15 / 8);`<br>`  writeln(15 div 7);`<br>`  writeln(15 div 8);`<br>`end.` | |

In your own words, what is the difference between using / and div ?

5. Let's look at type conversions. Type in a new program called expr4.pas. Compile and run this program, and write the output that you see.

| Program | Output |
|---|---|
| `program expr4;`<br>`begin`<br>`  writeln(2 + 1 / 2);`<br>`  writeln(7.0 - 1);`<br>`  writeln(7 - 1);`<br>`  writeln(round(4.8));`<br>`  writeln(trunc(4.8));`<br>`end.` | |

a. Based on the output, which expressions is/are an integer value?

b. If an expression contains an int and a real, what is the type of our answer?

c. Give an example of an expression whose operands are all int, but the final answer is a real.

6. Let's take a brief look at integer and real variables. This next program is called expr5.pas. Type in and compile this program.

| Program | Output |
|---|---|
| `program expr5;`<br>`var`<br>`  a : integer;`<br>`  x : real;`<br>`begin`<br>`  a := 4;`<br>`  x := a;`<br>`  writeln(x);`<br><br>`  x := 7.9;`<br>`  a := x;`<br>`  writeln(a);`<br>`end.` | |

In this program we attempt to assign an integer value into a real variable, and then a real value into an integer variable. One of these two operations is illegal. Which is it? Why do you think Pascal forbids this?

Comment out the illegal statement, as well as the subsequent call to the writeln procedure. Compile the program again. This time there should be no error messages. Run the program, and write the output you see in the box above. Why do you think the output appears the way it does?

7. Now, let's consider some more assignment statements. Type in this program, which is called expr6.pas. Compile it and run. Write down the output you see.

| Program | Output |
|---|---|
| `program expr6;`<br>`var`<br>`  a, b, x, y : integer;`<br>`  ans1, ans2 : real;`<br>`begin`<br>`  x := 5;`<br>`  y := 3;`<br>`  writeln(x:5, y:5);`<br><br>`  x := y;`<br>`  writeln(x:5, y:5);`<br><br>`  a := 7;`<br>`  b := a + 1;`<br>`  writeln(a:5, b:5);`<br><br>`  ans1 := a + b / x + y;`<br>`  ans2 := (a + b) / (x + y);`<br>`  writeln(ans1:5:2, ans2:5:2);`<br>`end.` | |

a. What happens when the above statement `x := y` is performed?

b. Why are the values of `ans1` and `ans2` different? Write down the conventional mathematical way of writing those assignment statements. In other words, using a horizontal fraction bar instead of using the / symbol.

c. Show how we would type an expression equal to: $\frac{\sqrt{2}+1}{\sqrt{2}-1}$. Use the built-in sqrt function. Verify with Pascal that your answer is about 5.828.

8. The plus operator can be applied to numbers or strings. Let's see it in action. Type in the following program expr7.pas. Compile and run this program, and write down the output you see.

| Program | Output |
|---|---|
| `program expr7;`<br>`begin`<br>`  writeln(8 + 4);`<br>`  writeln('8' + '4');`<br>`  writeln('numb' + 'er');`<br>`end.` | |

Explain how + works when we are working with strings.

9. Besides the error that you encountered in expr5.pas, can you think of another example of an illegal operation in Pascal? Try it out!

10. So, let me ask you: what is the most surprising or unexpected thing you have seen with the Pascal operators?

Next, let's compose more interesting programs. Here is one called add2.pas. Use nano to type in the following code.

```
(* add2.pas
 * Here is a simple program that will add 2 numbers.
 * Note that the compiler ignores comments, so this is
 * the perfect place to explain what the program does.
 *)
program add2;

var
  first, second, sum : string;

begin
  write('Please enter first number:  ');
  readln(first);
  write('Please enter second number:  ');
  readln(second);

  sum := first + second;

  writeln('Here''s the sum of your numbers:  ', sum);
end.
```

Now, we are ready to compile and run the program. Even with an easy program like this, it is always necessary to test our program to make sure we did not make a simple mistake somewhere.

11. Uh-oh! Something is wrong. Does the problem manifest itself by an error message, or by something awry in the output itself?

Well, we often learn by making mistakes! The problem we have in add2.pas is that the variables were declared of type string. Instead they should be integer. With this in mind, make the appropriate changes to the add2.pas program. Re-run the program to verify that it works correctly.

12. Type in this Pascal program. Save it as payroll.pas inside your lab2 folder.

```
(* payroll.pas – Calculate someone's weekly pay. *)
program payroll;

var
  pay, rate, HoursWorked : real;

begin
  rate := 5.75;
  HoursWorked := 15;

  pay := rate * HoursWorked;

  writeln('Your pay is $ ', pay);
end.
```

Now, run the program to make sure that it gives the correct output. Note that Pascal by default will print real numbers in scientific notation. How should we specify that the pay needs to be printed to only 2 decimal places? Make the necessary change in your program, and verify that the output looks presentable to the user. In the space below, describe what change you made to the writeln() procedure call.

Why are the variables declared to be real? Under what circumstances would integer variables make sense for this program?

Let's modify the program so that it asks the user for the hourly rate and the number of hours worked. Write down your input statements here:

13. Finally, let's play with a right triangle. We'll ask the user for the lengths of the two legs, and then print out the triangle's hypotenuse and area. Type in the following program, and save it as triangle.pas. Note that you need to complete some missing pieces.

```
(* triangle.pas
 * Determine the hypotenuse and area of a right triangle,
 * given the two legs.
 *)
program triangle;

var
  a, b, hypoteneuse, area : real;

begin
  writeln('Welcome to the right triangle program.');
  writeln();
  write('What is the length of side a?  ');
  readln(a);
  write('What is the length of side b?  ');
  readln(b);


  hypotenuse :=  (* you need to enter a formula here *)
  area :=        (* you need to enter a formula here *)

  writeln('The hypotenuse has length:  ', hypotenuse);
  writeln('The area is:  ', area);
end.
```

Save and run your program. Use 8 and 15 as input. What are the answers? _____

That's it! You are done with the lab.