UNIVERSITEIT·STELLENBOSCH·UNIVERSITY
jou kennisvennoot • your knowledge partner

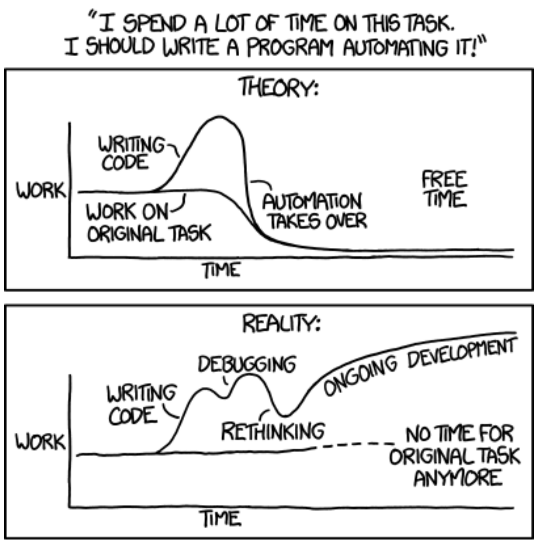
Computer Programming 143

Practical 3

2019

Aim of Practical 3:

- Implement a `switch-case` statement
- Use `do..while` loops in addition to previously applied loops
- Use complex boolean expressions
- Use functions from the math.h library
- Use the `double` declaration (instead of `float`) for all floating point variables.



Instructions

1. Attendance is **compulsory** for all the practical sessions of your assigned group. See the study guide for more details.
2. The last section (usually the last 30 minutes) of the practical will be used for a test.
3. If more than two tests have been missed for what ever reason, you will receive an **incomplete** for the subject. See the study guide for more details.
4. You must do all assignments **on your own**. Students are encouraged to help each other **understand** the problems and solutions, but each should write his/her own code. By simply copying someone else's code or solutions, you will not build an understanding of the work.
5. You are responsible for your own progress. Ensure that you understand the practical work. Check your work against the memorandum that will be posted on Wednesday afternoons on learn.sun.ac.za.
6. Use H:\CP143 as your Code::Blocks workspace folder for all projects. But it is highly suggested that you also use a **flash drive to backup** all your work.
7. Create a new project **for each assignment**. See *Creating a Code::Blocks Project* in Practical 0 for instructions on how to do this.
8. Include a comment block at the top of each source file according to the format given. It must include the correct filename and date, your name and student number, the copying declaration, and the title of the source file.
9. **Indent your code correctly.** Making your code readable is not beautification, it is a time and life saving habit. Adhere to the standards (refer to the documents on SUNLearn).
10. Comment your code sufficiently well. It is required for you and others to understand what you have done.

Question A

Goal: *Learn how to use* ***`switch-case`*** *statements.*

Getting started

1. Create a project named `Assignment3A`. Make sure that this is the active project in the workspace before compiling the program. This can be done by only having one project in the workspace or right click on the project and click "Activate Project."

2. Include the **standard comment block** above your main function. Also, comment your whole program appropriately.

3. Read the complete question before you start programming.

Program description

1. This question is very similar to Question C of Practical 2 (Assignment2C), where you wrote pseudocode for the program. The functionality of the program here is slightly different, but it has the same menu functionality. Instead of the commonly used **`if-else`** statements, you should use **`case`** statements to implement the menu here.

2. The description of the program is as follows: display a menu that will give the user three choices of operations to perform on three integer numbers. Read the choice. Print a message asking for the first number and read it in. Do the same for the second and third numbers. Execute the chosen operation on the three numbers. Do NOT include error checking, such as checking for a non-zero value. Print the result of the calculation. Use the sample output below as a guideline. Use **`case`** statements to implement the menu functionality.

3. The program should be able to execute the following three actions:

   (a) Calculate the mean of the three integers.

   (b) Determine which of the three integers is largest (maximum).

   (c) Determine which of the three integers is smallest (minimum).

   To calculate the maximum, you could use something like:

   ```
   max = num1;
   if (num2 > max)
       max = num2;
   if (num3 > max)
       max = num3;
   ```

4. Sample output:
   Sample 1:

```
Select an option:
1. Calculate the mean of the three integers
2. Calculate the maximum of the three integers
3. Calculate the minimum of the three integers
Enter operation number: 2
Enter first integer: -12
Enter second integer: 92
Enter third integer: 1

max(-12, 92, 1) = 92
```

Sample 2:

```
Select an option:
1. Calculate the mean of the three integers
2. Calculate the maximum of the three integers
3. Calculate the minimum of the three integers
Enter operation number: 1
Enter first integer: -1
Enter second integer: 2
Enter third integer: 1

mean(-1, 2, 1) = 0.66667
```

5. Ensure that your code is indented correctly and that the { } braces are on the correct lines.

6. Ensure that you copy the **Assignment3A** project folder to a flash drive as a backup.

Question B

Goal: *Use a* `do..while` *loop to calculate a sequence of values*

Getting started

1. Create a project named `Assignment3B`. Make sure that this is the active project in the workspace before compiling the program. This can be done by only having one project in the workspace or right click on the project and click "Activate Project."
2. Include the **standard comment block** above your main function. Also, comment your whole program appropriately.
3. Read the complete question before you start programming.

Program description

1. The Newton-Raphson Method can be used to obtain approximate numerical solutions for certain equations. Consider polynomials as an example. Second order polynomials (quadratic) are easy to handle, third and fourth order polynomials are much harder. From order 5 and higher there are no analytical solutions available and numerical approximations must be used.
2. Consider the quadratic function in **Figure 1** that will be used in this assignment. The Newton-Raphson Method works by taking a guess at where a specific root might be (point where the line crosses the y-axis) for a given function $f(x)$. This is the first approximation, x_1. A straight tangential line is then drawn at this point (at x_1). The point where the straight line crosses the y-axis, x_2, is the next (and hopefully more accurate) approximation of the function's root. Consider the quadratic function that will be used in this assignment:

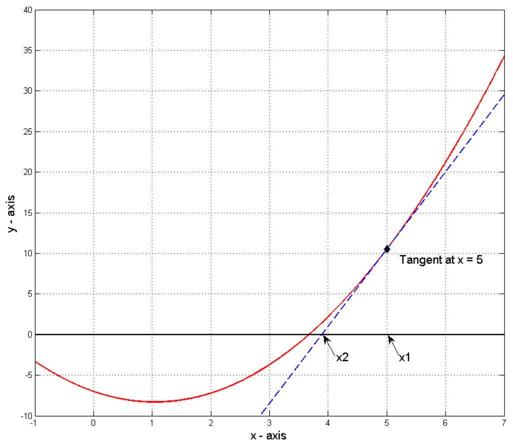


Figure 1: Parabola.

3. The first guess is $x = x_1 = 5$. A straight line (tangent) is drawn that crosses the y-axis at x_2, which becomes the new approximation for the function's root. x_2 can be used to calculate x_3, x_3 can be used to calculate x_4 etc.

4. The aim of the game is to find an x_n value for which $f(x_n) = 0$. The difference between $f(x_n)$ and zero is the error. The error should be reduced to below the error margin stipulated by the user, unless the maximum number of iterations is reached first.

5. You may use the following formula to obtain the next root approximation (given the current approximation x_n):

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

6. Your program must approximate the second (larger) root of the following quadratic function:

$$f(x) = 1.2x^2 - 2.5x - 7$$

7. Ask the user for an approximate x value to start off from. Check that it is larger than 4.

8. Ask the user for an error margin. This margin will be used in the **do..while** loop to decide if another iteration is needed.

9. Ask the user for a maximum number of iterations. If the error is not small enough by the time the maximum number of iterations have been completed the **do..while** loop must stop.

10. Your output must print the current value of x_n for each iteration, as well as the number of iterations (n).

11. Sample output of the program:
    Sample 1:

```
Enter a start value (greater than 4):5
Enter an error margin:0.001
Enter the maximum iterations:8

iteration number: 1, value of x1: 5.00, error of: 10.50
iteration number: 2, value of x2: 3.89, error of: 1.47
iteration number: 3, value of x3: 3.68, error of: 0.05
iteration number: 4, value of x4: 3.67, error of: 0.00
```

Sample 2:

```
Enter a start value (greater than 4):12
Enter an error margin:0.001
Enter the maximum iterations:4

iteration number: 1, value of x1: 12.00, error of: 135.80
iteration number: 2, value of x2: 6.84, error of: 31.99
iteration number: 3, value of x3: 4.54, error of: 6.35
iteration number: 4, value of x4: 3.78, error of: 0.69
```

12. Ensure that your code is indented correctly and that the { } braces are on the correct lines. Use the prescribed textbook as guideline.

13. Ensure that you copy the **Assignment3B** project folder to a flash drive as a backup.

Question C

Goal: *Implement the given flow diagram.*

Getting started

1. Create a project named `Assignment3C`. Make sure that this is the active project in the workspace before compiling the program. This can be done by only having one project in the workspace or right click on the project and click "Activate Project."
2. Include the **standard comment block** above your main function. Also, comment your whole program appropriately.
3. Read the complete question before you start programming.

Program description

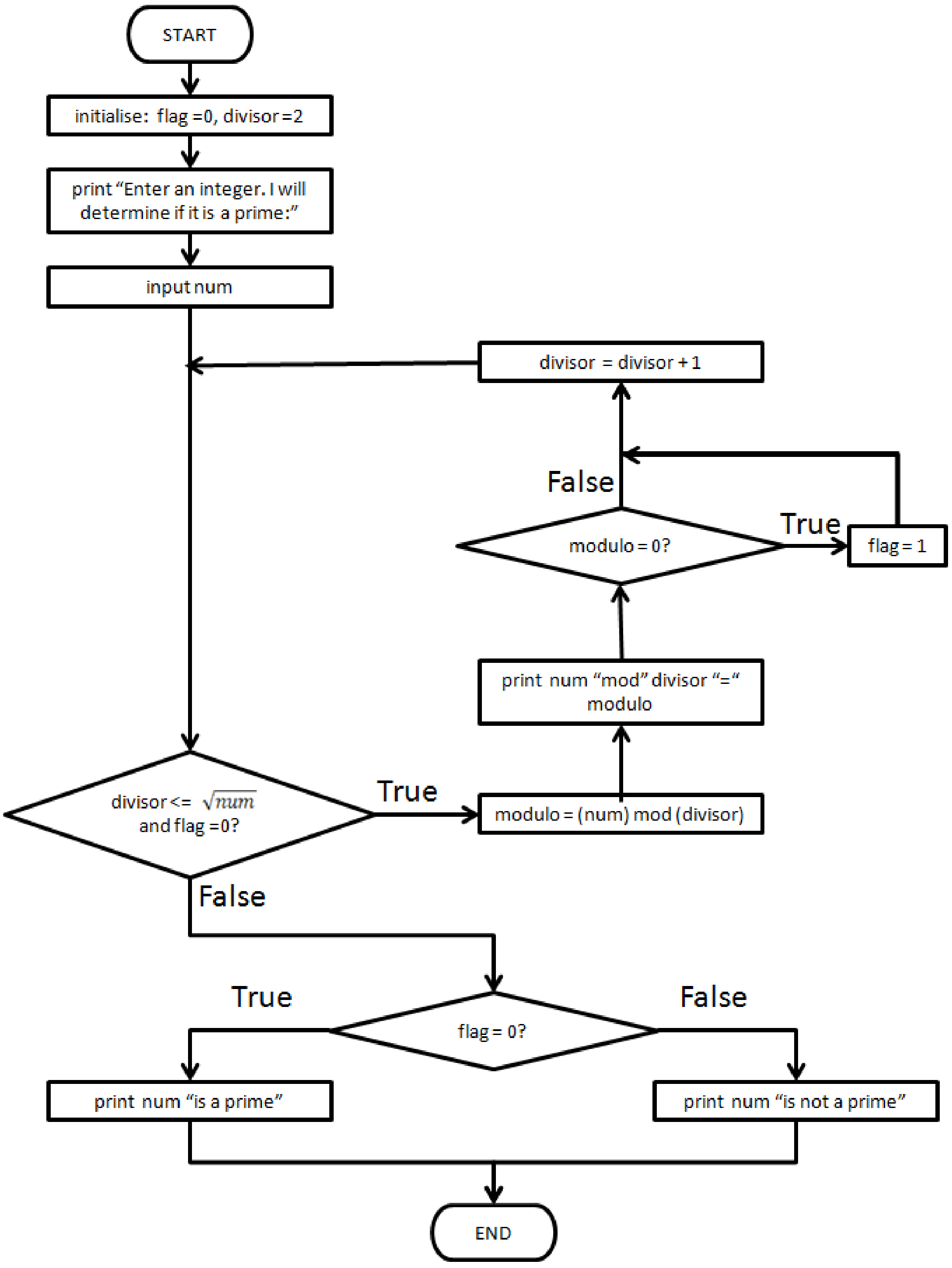
1. **Write a program that implements** the flow diagram given on page 10.
2. The **flow diagram** describes a program that asks **the user to enter** an integer, **which is then tested** to see if it is a **prime number** by determining if it is divisible by numbers smaller or equal to its square root (Hint: use the math.h library). A message stating whether it is prime or not is printed at the end.
3. Sample output of the program:
   Sample 1:
```
Enter an integer. I will determine if it is a prime: 2

2 is a prime
```
   Sample 2:
```
Enter an integer. I will determine if it is a prime: 5

5 mod 2 = 1
5 is a prime
```
   Sample 3:
```
Enter an integer. I will determine if it is a prime: 100

100 mod 2 = 0
100 is not a prime
```
   Sample 4:
```
Enter an integer. I will determine if it is a prime: 1345

1345 mod 2 = 1
1345 mod 3 = 1
1345 mod 4 = 1
1345 mod 5 = 0
1345 is not a prime
```

4. The program as it is written now has one big problem. Prime numbers are defined as positive integer numbers > 1, that are divisible only by 1 and itself. From this definition, it is clear that the program will give the wrong result for 0, 1 and any negative number. Add the functionality to the program so that it caters for these cases.

5. Ensure that your code is indented correctly and that the { } braces are on the correct lines. Use the prescribed textbook as guideline.

6. Ensure that you copy the **Assignment3C** project folder to a flash drive as a backup.

Note how easy it was to convert from the flow diagram into code, showing how one can go from a problem, implementing it as a flow diagram solution, and then converting into code in a simple implementation step. Be sure to figure out how this flow diagram works.


START
initialise: flag =0, divisor =2
print "Enter an integer. I will determine if it is a prime:"
input num
divisor = divisor + 1
False
modulo = 0?
True
flag = 1
print num "mod" divisor "=" modulo
divisor <= $\sqrt{num}$ and flag =0?
True
modulo = (num) mod (divisor)
False
True
flag = 0?
False
print num "is a prime"
print num "is not a prime"
END