SD11 Q1 Assignments – Week 2

Version: September 2018

Table of Contents

Assignments must be finished before the week 3 practicum!

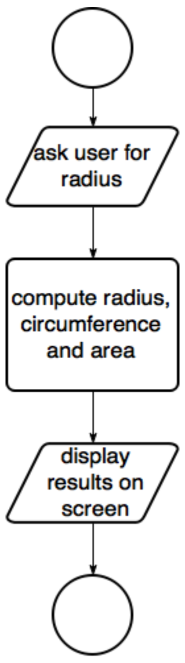
Assignment 2.1: Circle calculations (modified exercise 4.24 from the book)

Write a program that asks the user to enter the radius of a circle as an integer value. Your program should display the circle's diameter, circumference and area using the floating-point value 3.14159 for π.
Hint: Remember to use a named constant for π (of type **`double`**).

Given the radius (*r*) that the user enters, use the following formulas:

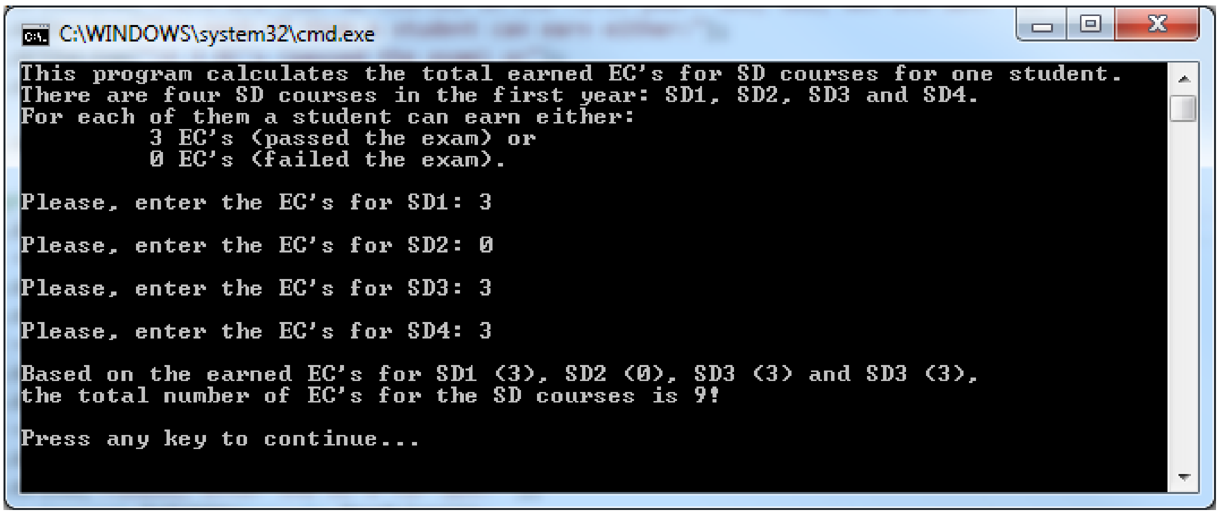
- diameter = 2**r*
- circumference = 2**r**π
- area = π**r***r*

In the flowchart (see picture on the right) you see the flow of the implementation. Make sure this flow is clearly visible in your code.


ask user for radius
compute radius, circumference and area
display results on screen

Assignment 2.2: Total earned EC for all SD courses

Write a C++ program to calculate the total earned EC's for SD courses for one Fontys Engineering student. In total, there are four SD courses in the first year: SD11, SD12, SD21 and SD22. For each of them a student can earn either 1.5 EC (passed) or 0 EC (failed). Ask the user to enter the number of earned EC's (0 or 1.5) for each of the four SD courses, and calculate and show the total of earned EC's. Draw a flowchart for this application similarly to the one provided in Assignment 2.1. The output of the program could look like the example shown below:



```
This program calculates the total earned EC's for SD courses for one student.
There are four SD courses in the first year: SD1, SD2, SD3 and SD4.
For each of them a student can earn either:
         3 EC's (passed the exam) or
         0 EC's (failed the exam).

Please, enter the EC's for SD1: 3

Please, enter the EC's for SD2: 0

Please, enter the EC's for SD3: 3

Please, enter the EC's for SD4: 3

Based on the earned EC's for SD1 (3), SD2 (0), SD3 (3) and SD3 (3),
the total number of EC's for the SD courses is 9!

Press any key to continue...
```

Assignment 2.3: Midpoints (Exercise 4.25 from the book)

In mathematics, the midpoint between the two points (x_1, y_1) and (x_2, y_2) is computed by the formula

$$\left(\frac{x_1+x_2}{2}, \frac{y_1+y_2}{2}\right)$$

Write a C++ program that asks for two mathematical points from the user and computes and computes and displays their midpoint.

A sample run of the program could produce:

```
Please enter the first point: (0,0)
Please enter the second point: (1,1)
The midpoint of (0,0) and (1,1) is (0.5,0.5)
```

The user literally enters ”(0,0)” and ”(1,1)” with the parentheses and commas as shown. To see how to do this, suppose you want to allow a user to enter the point (2.3,9), assigning the *x* component of the point to a variable named `x` and the *y* component to a variable named `y`. You can add the following code fragment to your program to achieve the desired effect:

```
double x, y;
char leftParen, comma, rightParen;
cin >> leftParen >> x >> comma >> y >> rightParen;
```

If the user literally types `(2.3,9)`, the `cin` statement will assign the `(` character to the variable `leftParen`. It will then assign 2.3 to the variable `x`. It assigns the `,` character to the variable named `comma`, the value 9 to the `y` variable, and the `)` character to the `rightParen` variable.

The `leftParen`, `comma`, and `rightParen` variables are just placeholders for the user’s input and are not used elsewhere within the program. In reality, the user can type in other characters in place of the parentheses and comma as long as the numbers are in the proper location relative to the characters; for example, the user could enter `*2.3:9#`, and the program would still interpret this as the point (2.3,9). We’ll learn ways to prevent this later in the course.

Assignment 2.4: Calories of food (Exercise 4.26 from the book)

The table on the right lists the Calorie contents of several types of food. Running or walking burns approx. 100 Calories per mile. Write a C++ program that requests three values from the user: the number of bean burritos, salads, and shakes consumed (in that order). The program should then display the number of miles that must be run or walked to burn off the Calories in the amount of food the user had. The program should run as follows (e.g. the user types in 3 2 1):

Food	Calories
Bean burrito	357
Salad w/dressing	185
Milkshake	388

```
How many bean burritos, bowls of salad, and milkshakes did you consume?

You ingested 1829 Calories
You will have to run 18.29 miles to expend that much energy
```

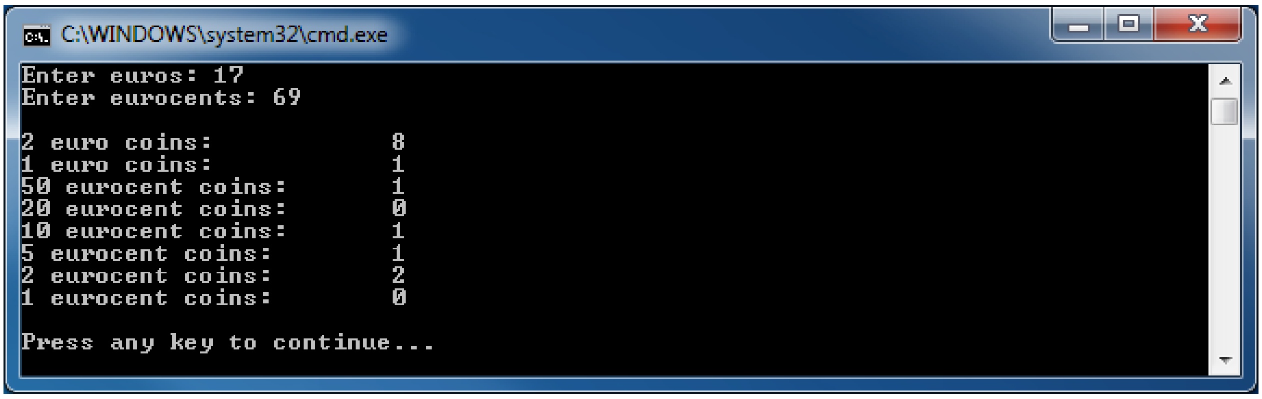
Observe that the result is a floating-point value, so you should use floating-point arithmetic to compute the answers for this problem.

Assignment 2.5: Minimize the coins

Write a program that first asks the user for two (positive) integer values. The first one is the number of euros, the second one is the number of eurocents.

The program must determine a set of coins with a value equal to the value of the amount entered, with the restriction that the number of coins must be minimized!
First think about how you solve this problem and only then start implementing!

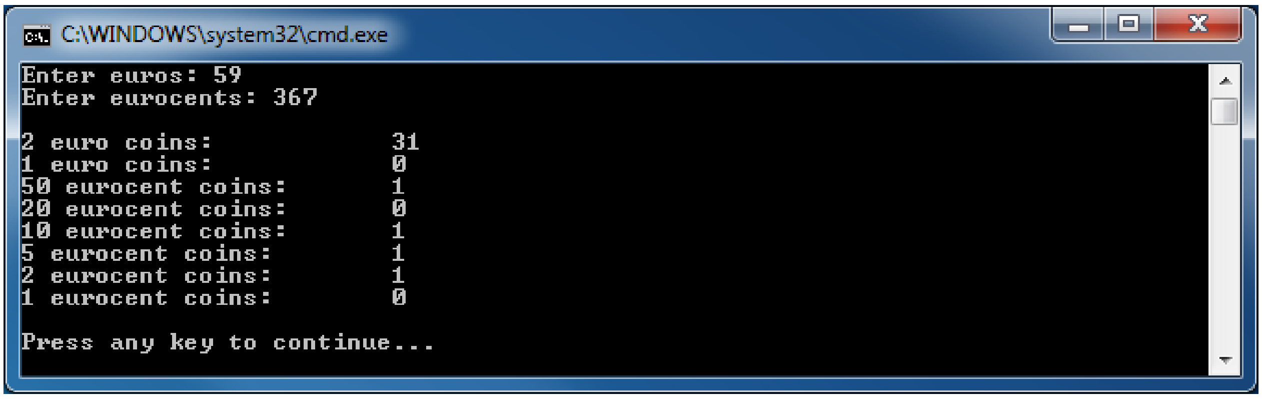
You must display the amount of coins at the end. Two possible results of your program are displayed below. What is the difference between these two examples?



```
Enter euros: 17
Enter eurocents: 69

2 euro coins:           8
1 euro coins:           1
50 eurocent coins:      1
20 eurocent coins:      0
10 eurocent coins:      1
5 eurocent coins:       1
2 eurocent coins:       2
1 eurocent coins:       0

Press any key to continue...
```



```
Enter euros: 59
Enter eurocents: 367

2 euro coins:           31
1 euro coins:           0
50 eurocent coins:      1
20 eurocent coins:      0
10 eurocent coins:      1
5 eurocent coins:       1
2 eurocent coins:       1
1 eurocent coins:       0

Press any key to continue...
```

Optional: could you change your program such that the total amount of money is read in a single line, similar to the method that we used in Assignment 2.3?

Assignment 2.6: Time (optional)

Design and write a program that reads two times in military format (0900, 1730) and prints the number of hours and minutes between the two times. Here is a sample run. User input is in color.

```
Please enter the first time: 0900
Please enter the second time: 1730
8 hours 30 minutes
```

Now try to also deal with the case where the first time is later than the second (i.e. the second time is on the next day)? For example:

```
Please enter the first time: 1730
Please enter the second time: 0900
15 hours 30 minutes
```