Flow control

1. Content

2. Objective

At the end of these exercises, you should be able to:

- Use the 3 loop statements :
  - for
  - while
  - do while

We advise you to **make your own summary of topics** that are new to you.

3. Exercises

Your name, first name and group should be mentioned at the top of each cpp file.

Make the labs in the same order as this assignment

Create a new project with name **IterationsBasics** in your **1DAExx_05_name_firstname** folder.

3.1.1. For – while – do while statements

a. Even numbers

We have 3 iteration statements (for, while and do-while) at our disposal, one being more convenient than the others for a given task.

Print the **even** numbers from **2 until 24** using the 3 loop statements.

```
-- Even numbers --
2 4 6 8 10 12 14 16 18 20 22 24
2 4 6 8 10 12 14 16 18 20 22 24
2 4 6 8 10 12 14 16 18 20 22 24
```

b. Exam scores

An exam score is translated into a letter score as indicated in the table below:

Letter	Score
A+	[18, 20]
A	[16,17]
B	15
C	[13,14]
D	[11,12]
E	10
FX	[8,9]

F	[0,7]

In this exercise, the user enters scores and when -1 is entered, the results are printed:

- Every letter followed by how many times that score has been entered.
- The total score
- A list of all the entered scores.

```
-- Exam scores --
Score [0,20] ? 6
Score [0,20] ? 8
Score [0,20] ? 21
Wrong score!
Score [0,20] ? 10
Score [0,20] ? 12
Score [0,20] ? 14
Score [0,20] ? 18
Score [0,20] ? 20
Score [0,20] ? -1
A+: 2
A: 0
B: 0
C: 1
D: 1
E: 1
FX: 1
F: 1
Total score: 88
Entered scores: 6, 8, 10, 12, 14, 18, 20
```

How to proceed?

Make a loop that requests the input of an exam score. The loop continues until -1 is entered.

If the exam score is invalid - not in the range [0, 20] - then print an error message.

If the entered score is valid then the corresponding letter score and the total number of scores are both incremented.

c. Random guess

In this exercise you'll count how many times the rand() function needs to be called before resulting in a requested value.

- Make the user enter a number.
- Using a loop, generate random numbers until getting a number equals the entered number.
- Display how many guesses (rand calls) were needed.

```
-- Guess number --
Number to guess ? 20
Number found after 5556 guesses
```

Always test your code for all cases, e.g. enter a negative number. What do you notice?

Adapt the code, don't start the statistics when an invalid number – not in [0, RAND_MAX] - is entered. RAND_MAX has the value 0x7FFF or 32767. When this is the case, give the user an appropriate message and ask again to ENTER a number. Repeat this until a correct value is entered.

```
-- Guess number --
Number to guess ? -6
This is a wrong number, number to guess ? -20
This is a wrong number, number to guess ? 21
Number found after 1881 guesses
```

d. Trigonometry

Print a table having 4 columns: degrees, radians, cosine and sine and fill them with the values for angles in the interval [0, 180] degrees using steps of 10 degrees.

The columns have a width of 10, the values a precision of 2 decimals and no exponents are used. The input/output manipulators help you to achieve this format.

For more information see Input/output manipulators

```
-- Trigonometry --
   Degrees    Radians        Cos        Sin
         0       0.00       1.00       0.00
        10       0.17       0.98       0.17
        20       0.35       0.94       0.34
        30       0.52       0.87       0.50
        40       0.70       0.77       0.64
        50       0.87       0.64       0.77
        60       1.05       0.50       0.87
        70       1.22       0.34       0.94
        80       1.40       0.17       0.98
        90       1.57       0.00       1.00
       100       1.75      -0.17       0.98
       110       1.92      -0.34       0.94
       120       2.09      -0.50       0.87
       130       2.27      -0.64       0.77
       140       2.44      -0.77       0.64
       150       2.62      -0.87       0.50
       160       2.79      -0.94       0.34
       170       2.97      -0.98       0.17
       180       3.14      -1.00       0.00
```

e. ASCII table

Print all the letters of the alphabet and its corresponding ASCII value in 4 columns, without using hardcoded ASCII values however using a loop of course.

For more information see ASCII table

```
-- ASCII table --
 a      97   n     110
 b      98   o     111
 c      99   p     112
 d     100   q     113
 e     101   r     114
 f     102   s     115
 g     103   t     116
 h     104   u     117
 i     105   v     118
 j     106   w     119
 k     107   x     120
 l     108   y     121
 m     109   z     122
```

3.2. ErrorSolving

Create a new project with name **ErrorSolving** in your **1DAExx_05_name_firstname** folder.

Overwrite the generated file **ErrorSolving.cpp** by the given one.

3.2.1. Compiler errors and warnings

None

3.2.2. Linker errors

None

3.2.3. Runtime errors

This code builds without problems, but the application doesn't behave as it should.

The code is split in 3 functions each one having a runtime problem. The problems are described here:

(1) **Problem in the "ShowMultiplyTable" function**

This function should print the multiplication table of 5, but only 1 multiplication is shown.

```
5 x 0 = 0
```

Expected result

```
5 x 0 = 0
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
```

(2) **Problem in the "GuessNumber" function**

This function asks the user to guess a number within the interval [1,10], but the first guess is always considered to be the right one.

```
Number in [1, 10] ? 2
You entered the right number 5 after 1 guesses!
```

Expected result

```
Number in [1, 10] ? 1
Number in [1, 10] ? 3
Number in [1, 10] ? 4
You entered the right number 4 after 3 guesses!
```

(3) **Problem in the "GetNumber" function**

The **GetNumber** function asks to enter a choice [5,10], when a wrong number is entered it should print an error message and ask the question again.

But the error message is also shown at the start, so when the user even didn't enter a number

```
Sorry, but this is not a correct number
Enter a number in the interval [5, 10] 4
Sorry, but this is not a correct number
Enter a number in the interval [5, 10] 5
```

Expected result

```
Enter a number in the interval [5, 10] 4
Sorry, but this is not a correct number
Enter a number in the interval [5, 10] 5
```

3.3. Rectangles

Create a new project with name **Rectangles** in your **1DAExx_05_name_firstname** folder.

Delete the generated **Rectangles.cpp** file.

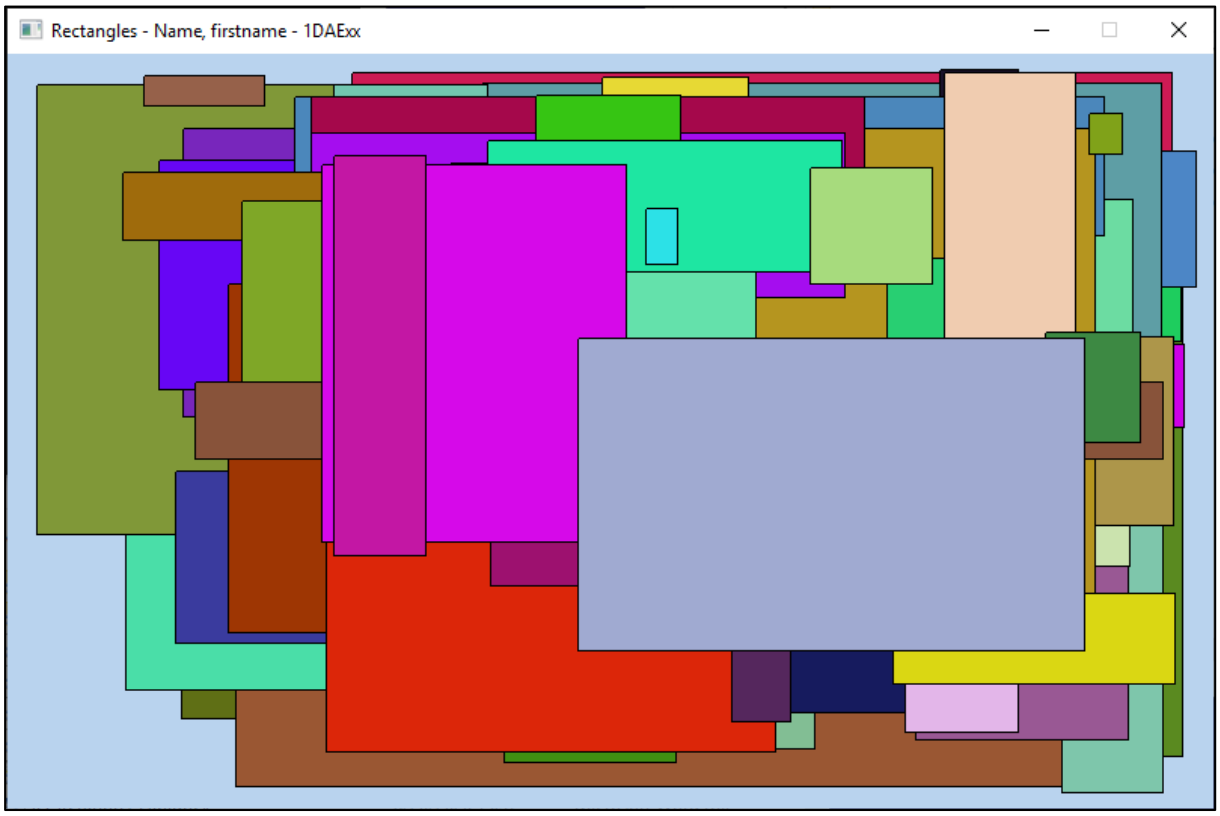
Change the window size into 800 x 500.

Adapt the title.

Draw 100 colored rectangles with a black border.

Each rectangle's color, position, width and height has a random value, each rectangle is completely inside the window boundaries, leaves at least a 10 pixel empty space and has a minimum height and width of 10 pixels.

To slow down the animation, use Sleep(1); (PS: Sleep is not ISO C++)



3.4. Curves

Create a new project with name **Curves** in your **1DAExx_05_name_firstname** folder.

Delete the generated **Curves.cpp** file.

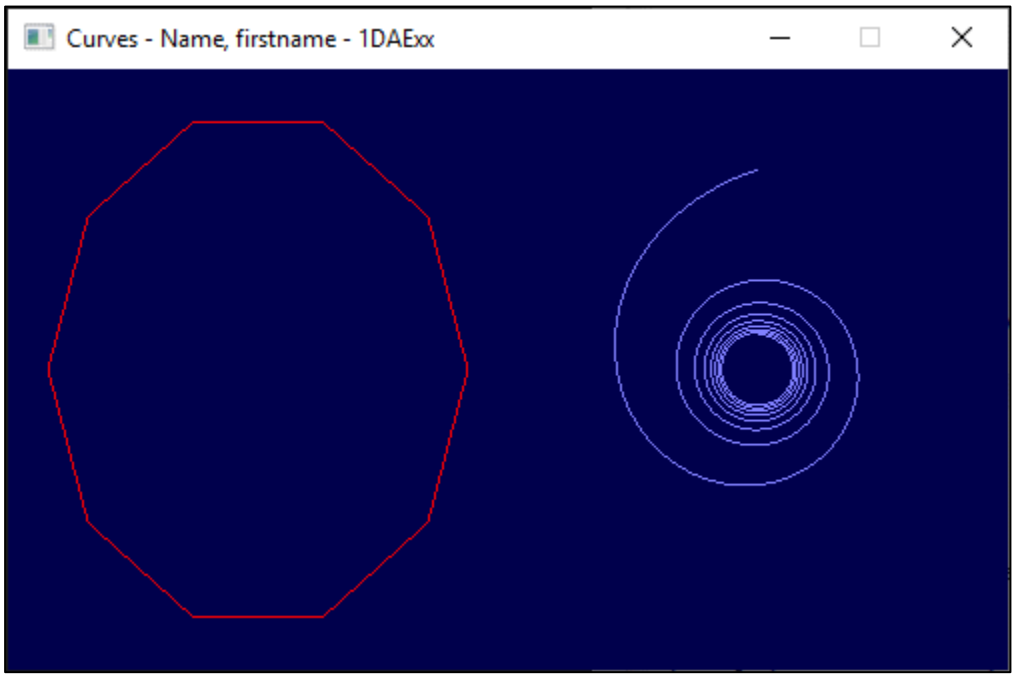
Change the window size to 500 x 300.

Adapt the title.

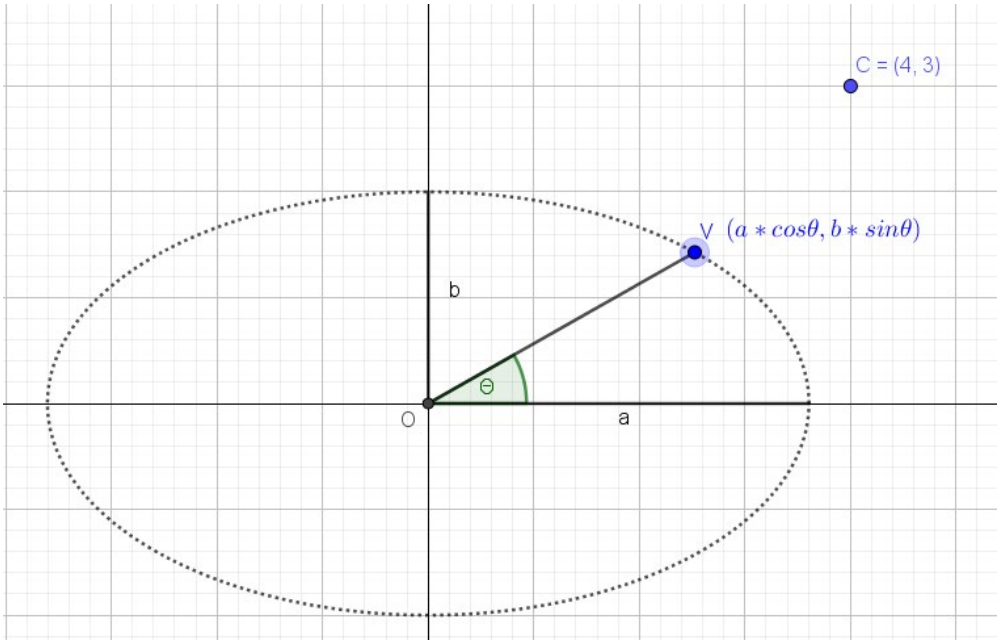
In this project you draw:

- A red ten corner shape
- A crosier curve using a loop in which you vary the angle of the polar coordinates of the coordinates located on these curves.

More information about this exercise follows. In the end you should have a window that looks like this.



3.4.1. Ellipse



We will NOT use the DrawEllipse function in utils. Instead we will draw 10 lines that connect points. We just need to know the coordinates of these points that describe the circumference of that ellipse. The coordinates of these points can be expressed as a function of the angle θ they form with the positive x- axis (see vertex V in the image above). Varying that angle from 0 to 2π using small angle steps and using a loop, we can calculate the cartesian coordinates (x, y) of all these vertices.

However, this picture shows an ellipse with center that coincides with the origin. When you want the ellipse to have a center C that doesn't coincide with the origin, you just have to add the coordinates of that center to the x and y of each vertex.

Vx = Cx + a * cos θ

Vy = Cy + b * sin θ

The C++ math function to calculate the sin and cos root of a float is: sinf and cosf.

Use this information to draw a red octagon.

3.4.2. Crosier

The crosier curve is defined with this polar equation

$r = a/\sqrt{\theta}$ With the opening a > 0

Using a loop varying the angle θ from [π/2, b * 2π + π/2[- with b being the desired number of windings - you can calculate the r value. And with this combination of r and θ values you can calculate the x and y value of this curve 's vertices (the same way as in the ellipse exercise)

Vx = Cx + r * cos θ

Vy = Cy + r * sin θ

The C++ math function to calculate the square root of a float is: sqrtf.

To obtain the curve like the one on the screenshot above we used these values

a	Window width / 4
b	8
Angle steps	0.01 radians

3.5. DiceStats

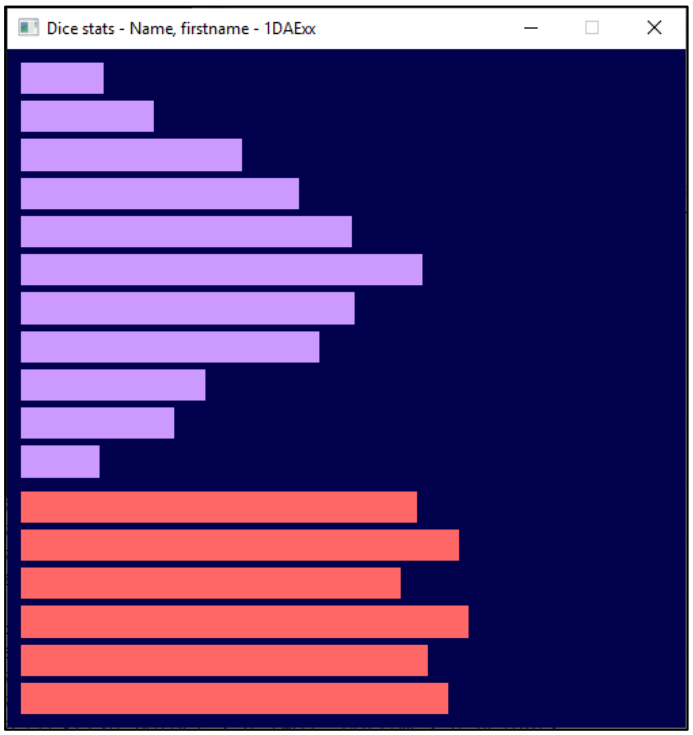
Create a new project with name **DiceStats** in your **1DAExx_05_name_firstname** folder.

Delete the generated **DiceStats.cpp** file.

Adapt the window title.

Change the size into 500 x 500.

In this exercise you visualize the statistics of numbers obtained by rolling the dice. These statistics are visualized using filled rectangles with a width corresponding to the number of times the corresponding number was rolled. At the end of this exercise, you should have a window that looks like this. More details follow this screenshot.



3.5.1. The statistics of throwing one die

Each Draw-frame you simulate rolling the die – use the rand function - and keep track of the number of times each number [1,6] was rolled (you 'll need a variable for each). Then visualize the number of times the values 1, 2, 3 ... 6 are thrown by drawing a filled red rectangle with a width that is equal to the number of times this number of pips was thrown.

3.5.2. The statistics of throwing 2 dice

Now simulate rolling two dice and keep track of each possible result.

Show on the window again a rectangular statistics for each possible result (2,3, … 12) using a blue color.

Rebuild and run. Can you explain the statistic results?

3.6. LoopedDrawings

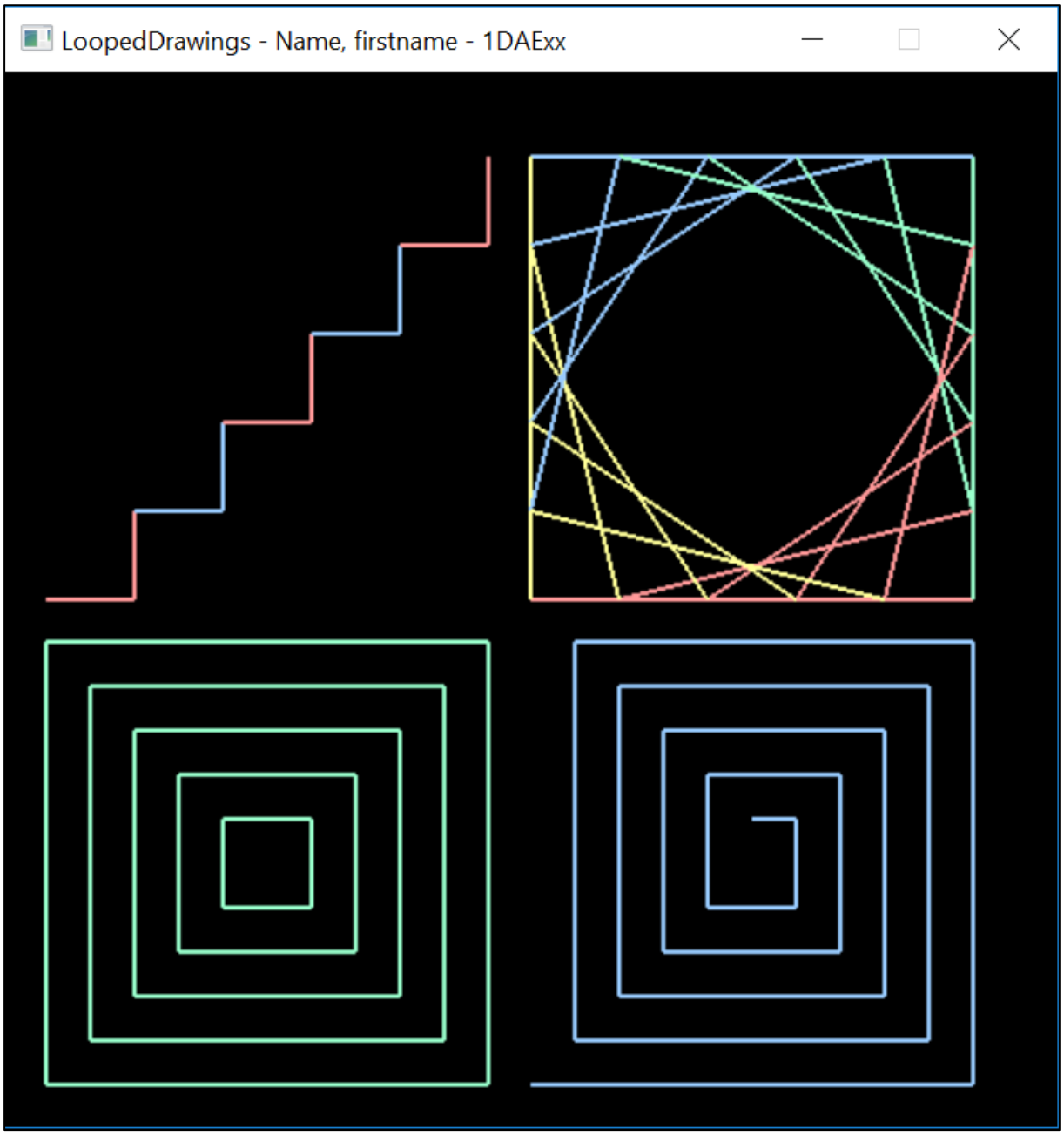
Create a new project with name **LoopedDrawings** in your **1DAExx_05_name_firstname** folder.

Delete the generated **LoopedDrawings.cpp** file.

Adapt the window title.

Change the size into 500 x 500.

After making the exercises, you should have a window that looks like this.



Using loop statements, and splitting up the code in functions, draw:

- 5 concentric squares.
- A stair having 5 steps with alternating color
- A spiral having 5 loops
- Four sets of 5 spinning lines, each set having another color.
    1. Lines starting at the **top** and ending at the **right** side
    2. Lines starting at the **right** side and ending at the **bottom**
    3. Lines starting at the **bottom** and ending at the **left** side
    4. Lines starting at the **left** side and ending at the **top**

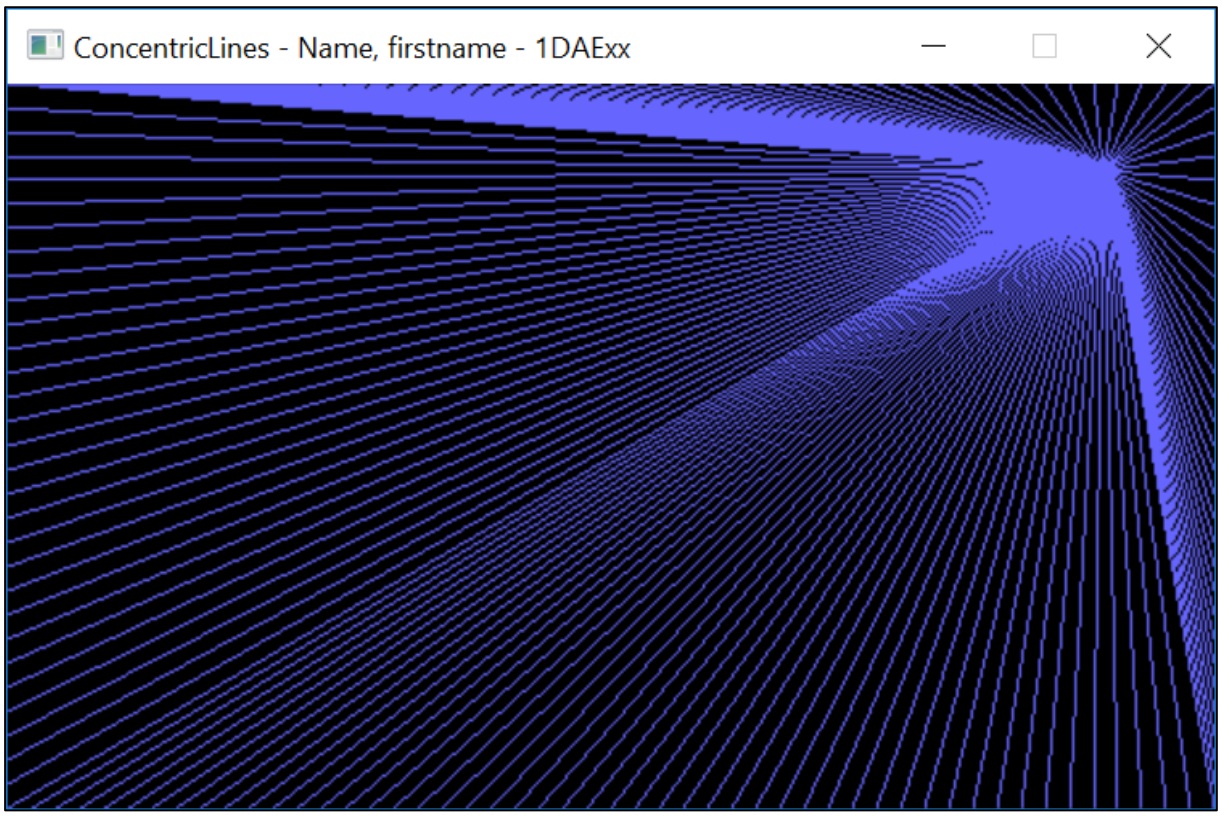
3.7. ConcentricLines

Create a new project with name **ConcentricLines** in your **1DAExx_05_name_firstname** folder.

Delete the generated **ConcentricLines.cpp** file.

Adapt the window title.

After making the exercises you should have a window that looks like this.



Track the mouse position in the mouse move event.

Define and call a **DrawLines** function that draws lines from the edges of the window to the mouse position. There are 9 pixels between each line on the edges.

Then switch between 3 colors each time a key is pressed.

- Define a **Color** enum type with 3 enumerators: **red**, **green** and **blue**. And define a global variable g_Color of that enum type.
- When the right arrow key is pressed, make g_Color switch to the next color, to that end define a function **SwitchColor**. Do not set the Draw Color in the SwitchColor function, its only purpose is to switch the color, not to set the draw color!
- In the DrawLines function check this variable to decide which color you should use to draw the lines.

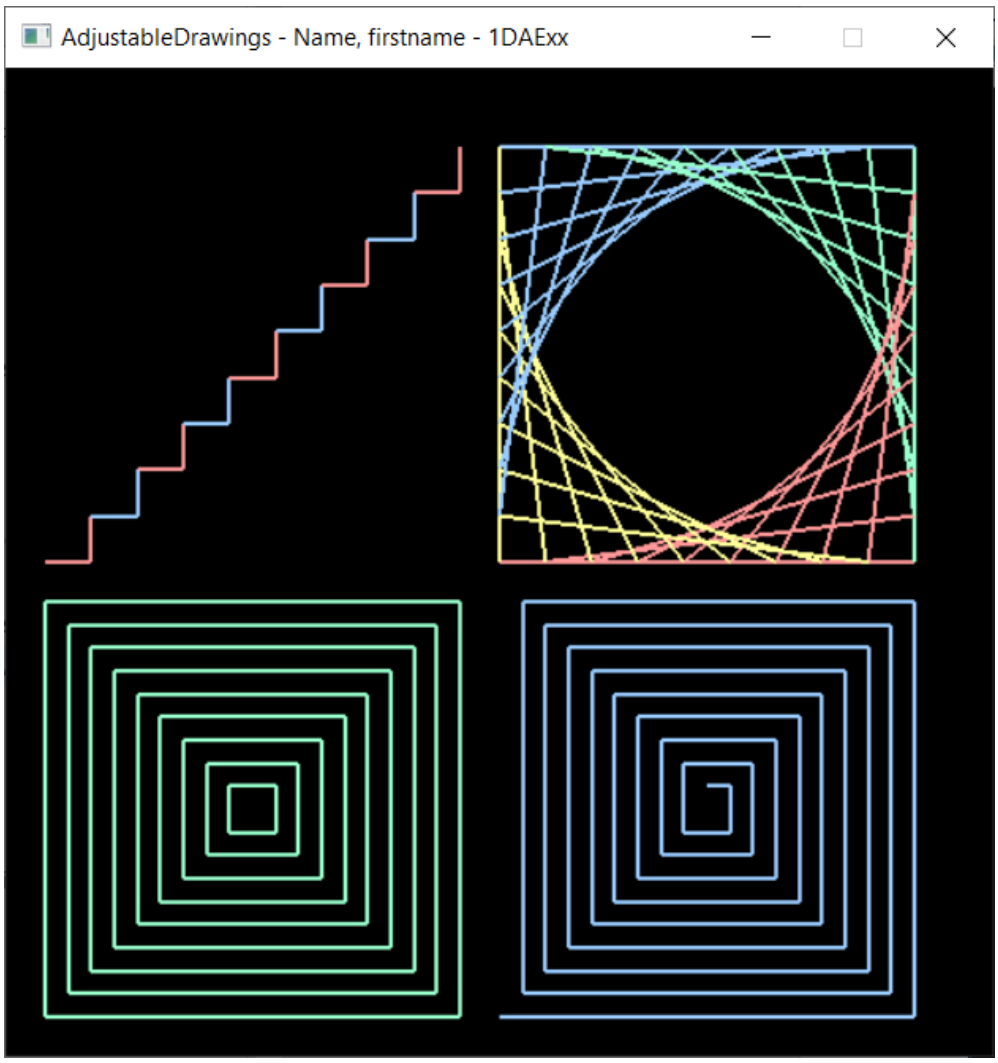
3.8. AdjustableDrawings

Create a new project with name **AdjustableDrawings** in your **1DAExx_05_name_firstname** folder.

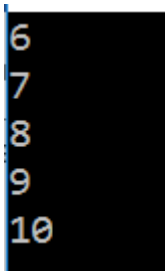
Delete the generated **AdjustableDrawings.cpp** file.

Adapt the title and change the window size to 500 x 500.

This project is similar to the previous LoopedDrawings, however now the number of loops is adjustable within the inclusive range [5, 20] by pressing the up and down arrow key.



Show the number of loops in the console, each time this values is changed.



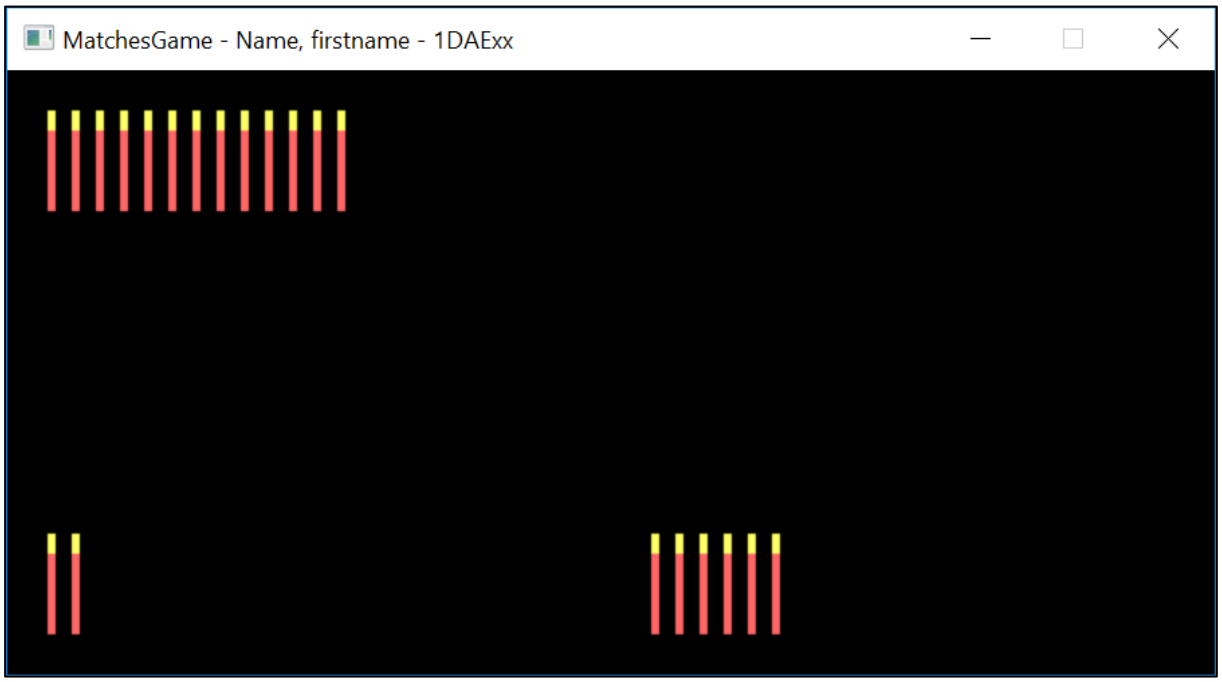
3.9. MatchesGame

Create a new project with name **MatchesGame** in your **1DAExx_05_name_firstname** folder.

Delete the generated **MatchesGame.cpp** file.

Adapt the title. The window size is 600 x 300.

After making the game you should have a window that looks like this.



Make a player-alternated game that starts with a pile of 21 matches lying on the table. In turn, the players have to take 1, 2 or 3 matches from the pile. The player who removes the last match, loses the game.

You are one of the two players.

You have to push 1, 2 or 3 to enter how many matches you want to take from the pile at the top of the window. Draw your own pile at the bottom left side of the window.

The other player is your computer program, which - using the rand function - determines the number of matches to take from the pile. The programs pile is on the bottom right side.

At the end of the game, the program shows who won the game and a list of drawn matches on the console.

```
PC moves: 3 3 2 2 3 1
Your moves: 1 1 1 1 1 2
You win
```

For those who don't know how to start, some tips:

1. **The drawing part**. What do you need to know in the Draw function? The number of matches on the main pile, the user pile and the computer pile. So define 3 variables that hold this information. Think about the value each one should get at start. Then use these variables to draw the 3 series of matches. Test different situations by initializing the variables with some other values.
2. **The part in which the user draws matches** from the main pile.
   When does this happen ? when one of the keys 1,2 or 3 is pushed. So add this code in the event handling part.
   What should happen ? Change the number of matches in the main pile and and the user pile. Test this for different cases.
3. **The part in which the computer draws matches** from the main pile.
   When does this happen? After the user has drawn some matches and provided that there are still matches remaining on the main pile. So add this code after the code of previous point.
   What should happen ? A random number in [1,3] and not more than

available in the main pile is generated. Then change the number of matches in the main and computer pile accordingly. Test this for different cases.

4. **The part that determines whether the player wins or loses** the game.
   When ? When the number matches in the main pile matches becomes 0.
   Where ? After code that moves matches from the main pile.
   Does the player win or lose ? Player loses when taking the last match, wins when the computer takes the last match. Show a message in the console. Test this.
5. **The part that shows the drawn matches of the player and computer at the end.**
   How and when ? You need a string variable for each one and concatenate the number of matches drawn to it each time when the player/computer draws matches.

3.10. FreeGame

Create a new project with name **FreeGame** in your **1DAExx_05_name_firstname** folder.

Delete the generated **FreeGame.cpp** file.

Adapt the title.

Now that you know how to handle gamer events and how to control the flow of your code using selection and iteration statements, make a basic game of your choice using this knowledge.

If you have no inspiration, you could make one of the following games.

3.10.1. Animated matches game

Add a lot of animation and extras to the matches game e.g. one in which you see the matches move from the main pile to the gamers piles..

3.10.2. Pong game

Pong is a "tennis like" game, it is one of the earliest arcade video games. You decide about the visual aspects of the game.

These are the minimal requirements:

- The game should be playable by 2 human players.
- The paddles are controlled using keys.
- The game is resettable (scores become 0) by pressing a key.
- The scores are shown on the window (e.g. using a series of filled rectangles).
- When the I-key is pressed, info about the game (e.g. which keys to use) is displayed on the console.

Don't hesitate to add some extra's, e.g. multiple levels, play against computer…

4. Submission instructions

You have to upload the folder *1DAExx_05_name_firstname, however first clean up each project. Perform the steps below for each project in this folder:*

- In Solution Explorer: Select the solution, RMB, choose **Clean Solution** or delete the debug folder after closing the visual studio.
- Then **close** the project in Visual Studio.

- Delete the .vs folder.

Make sure that you answered the quiz.

Compress this *1DAExx_05_name_firstname* folder and upload it before the start of the first lab next week.

5. References

5.1. Input/output manipulators

5.1.1. Operator Precedence

http://en.cppreference.com/w/cpp/language/operator_precedence

5.1.2. Set field width

http://www.cplusplus.com/reference/iomanip/setw/

5.1.3. Set the decimal precision

http://www.cplusplus.com/reference/iomanip/setprecision/

5.1.4. Use fixed floating-point notation

http://www.cplusplus.com/reference/ios/fixed/

5.2. Enumeration

https://www.youtube.com/watch?v=Pxvvr5FAWxg

http://www.learncpp.com/cpp-tutorial/4-5a-enum-classes/

5.3. ASCII table

http://www.learncpp.com/cpp-tutorial/27-chars/

5.4. Algorithm to draw circles and ellipses

http://www.mathopenref.com/coordcirclealgorithm.html