

## Quality

Lack of quality means **fewer points**..

The code must be easy to read and maintain and complies with the programming rules.

The project has to build without errors and without warnings.

Use functions where it's appropriate.

There may not be any code that has nothing to do with this assignment (no copy/paste of other exercises).

Names of variables, functions, enumerations, structs,.. must be meaningful and follow the agreed naming rules.

Choose local variables over global if possible.

**It is not allowed to use C++ features that have not yet been discussed in the lessons. For example: arrays, classes,...**

## General

The assignment consists of two parts:

1. A console exercise.
2. A graphic exercise.

You make both in the same framework solution.

## Technical requirements

It is important that you perform the assignment as requested so that we see if you have acquired certain programming skills. If your solution works, but if you have solved it in a different way, you will get fewer points. This includes, for example: The creation and use of the described functions, structs, enumerations, ...

## Given

- **A zip file with** a working example.

## Create the project

1. Create a visual studio project with the framework, using the files in 00\_General/FrameworkFunctions\_V0.2.3.zip. And described in 00\_General/HowToCreateProject.pdf. The name of the project is **Prog1DAExx\_Qyy\_Familyname\_Firstname**, replacing **xx** with your group number, **Family name** by your name and **First name** by your first name, **yy** by the Q exam number as it can be found in the header of this document.
2. Place your full name and group in the **title of the window**.

## Console part

## 1. Introduction

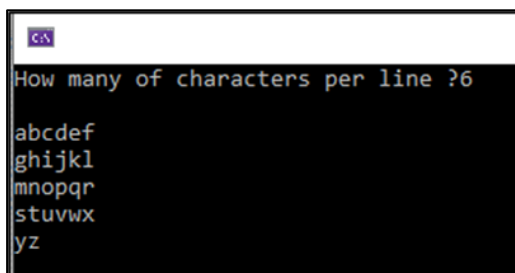
To be able to print in the console from the game.cpp file, you need to include this: `#include <iostream>`.

## 2. Assignment

**Create a PrintLetters** function. The function is called from the Start function in Game.cpp.

In the function there is code that results in this:

- Ask the user how many characters per line the user would like to see.
- Print all the letters of the alphabet (lower case) in the requested number of characters per line.
- Don't use `std::string` for this assignment.
- Don't print hard-coded text on the console e.g. `std::cout << "abcdef\n";`
- Use one or more iteration loops.



```
How many of characters per line ?6
abcdef
ghijkl
mnopqr
stuvwx
yz

```

Here you can see what the console should look like.

## Circle

### **Behaviour description:**

When clicking on the window, a randomly coloured circle starts growing, centred on the clicked position from 0 till a random maximum radius. That max. radius can be any whole number in the range [50,150], including 50 and 150.

When the maximum radius is reached, it shrinks till it disappears.

The moment the circle overlaps the window border while growing and before the maximum radius is reached, it turns red and prints "Ouch!" on the console. It also immediately starts shrinking till it disappears. In other words, the circle never goes out of the window.

After that nothing happens until there is another click in the window and it all re-starts.

Be careful to not draw/fill an ellipse with a negative radius, as this results in an infinite loop in the fill ellipse function.

Clicking the mouse while the circle is visible has no effect.

Each new circle has a new random colour and a new maximum radius.

The growing/shrinking speed of the radius can be modified with the arrow keys in the inclusive interval [0.2,5], in steps of 0.4 including the 0.2 and 5. Each new speed is printed in the console.

### **Implementation:**

The circle has three states: growing, shrinking, and waiting.

First create an enumeration class and variable to represent these states. If you are not able to work with enumerations, use an alternative, but you will lose points.

In the OnMouseUpEvent function:

- When the left mouse button is released in the window while no circle is visible, the state changes to growing and a new max radius and colour value is generated. Clicking mouse buttons while the circle is visible has no effect. The new position is stored. The maximum radius value and the new state are printed on the console. e.x and e.y contain the Windows coordinates of the mouse. Don't forget to flip the y-coordinate.

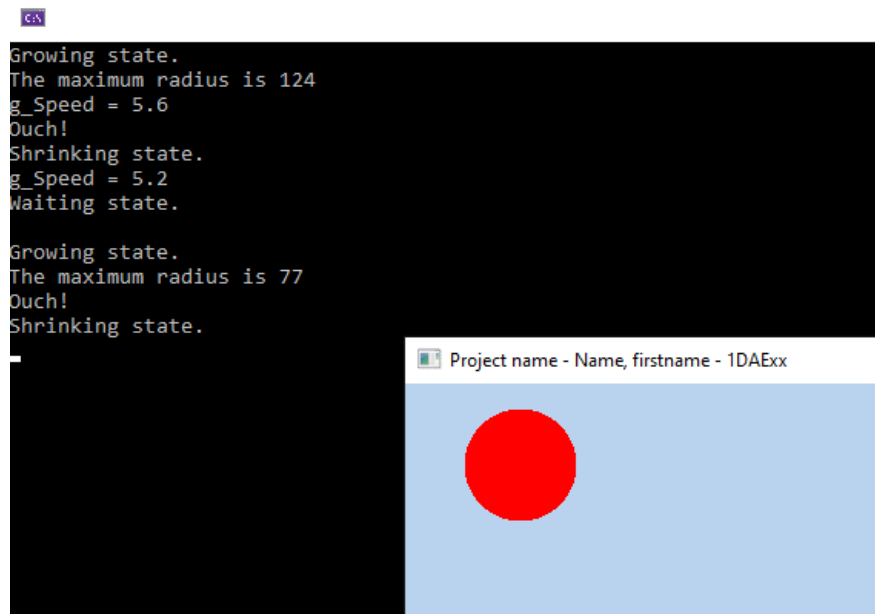
In the Update function:

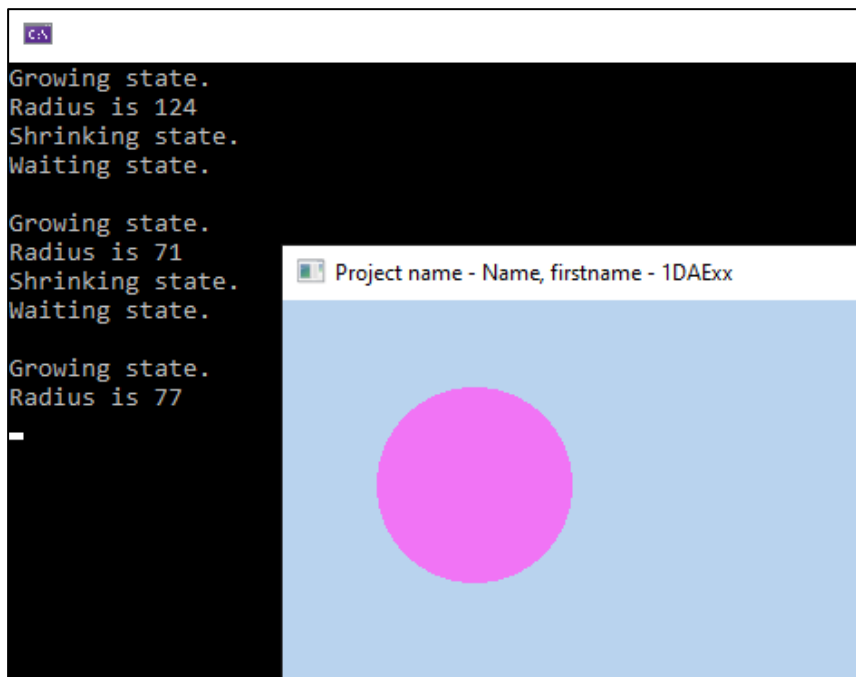
- Check the state and change the radius value accordingly.
- Check for collision with the window border and maximum radius value and handle the state/colour accordingly. When the circle overlaps the window border, "Ouch!" is printed to the console.

In the OnKeyUpEvent function:

- Check the up and down arrow keys and change the speed as described above.

Each time the state changes, a message showing the new state is printed to the console as seen on the screenshot.





## Submit

Close Visual Studio. Remove the .vs and the debug and/or the x64 folder. Convert the solution folder to a rar/zip file with the same name as the folder. Check if the zip file still contains all the necessary code files.

Submit this rar / zip file under the leho assignment of this lab exam. See **for yourself** if this has been done correctly. Incorrect exams – for example, another project, a project with missing code files – get 0 points.