Programming Lab

2023

CS1PC20 Sample: Programming in C/C++

This tutorial is an example from the Computer Science module Programming in C/C++, which is a first year module. It demonstrates how the lab practicals are organized:

- *Lectures deliver theoretical knowledge.*
- *Lab practicals apply the theory in practice and provide training for skills such as programming.*
- *In the Programming module, lab practicals use tutorials consisting of individual work and group work.*

If you were a first year student taking the Programming module, by this point, you would already have been given additional exercises to help you practise the basics before attempting these tasks. For this session, though, we have decided to jump straight into the more interesting part of the practical.

If you have any questions, we encourage you to attract the attention of the lecturer or the assistants, for example, by raising your hand.

Learning Objectives

By the end of the tutorial, you will be able to:

- Use Visual Studio to execute and modify a program.
- Describe the behaviour of the structured programming theorem constructs (sequence, selection, iteration).

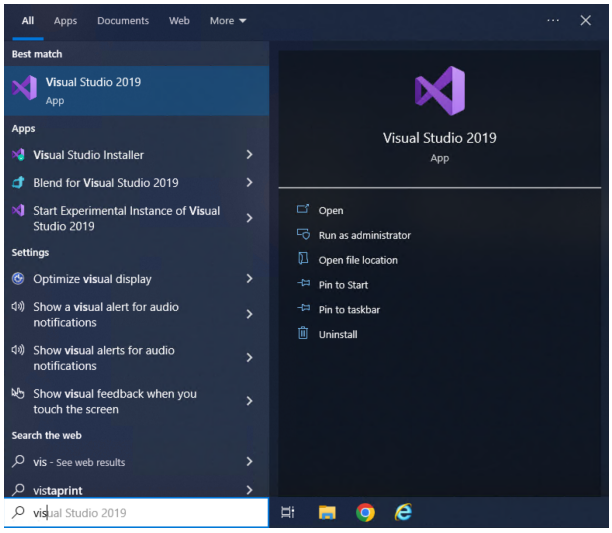
Contents

1. Tutorial (20 Minutes): Compiling a Program using Visual Studio

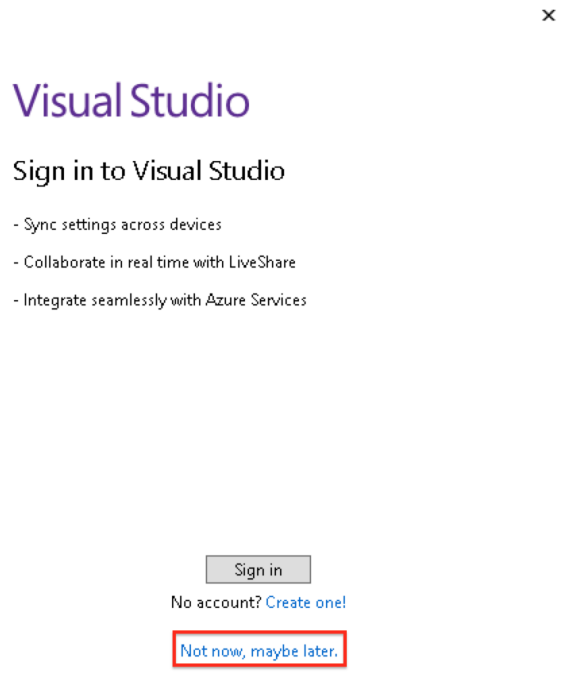
As part of this tutorial, we will use Visual Studio to download, modify and build an existing program.

Steps

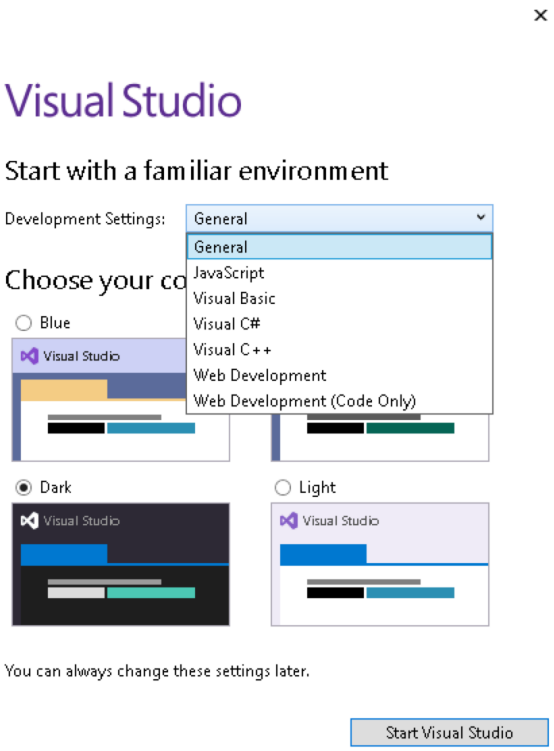
1. In the Start menu, search for and start **Microsoft Visual Studio**.



2. Once Visual Studio is loaded, if you are asked whether you want to "Sign in to Visual Studio", select: "**Not now, maybe later**" at the bottom.

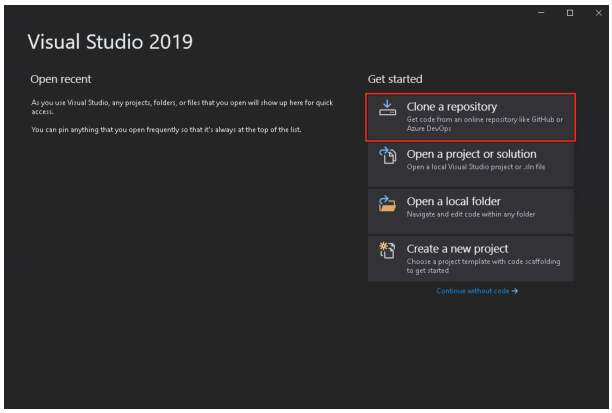


3. Choose **General** Development settings. Select the **Dark** theme to match the following images.

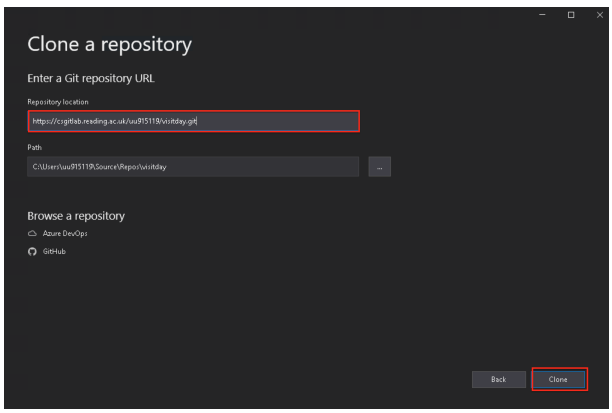


4. Wait a minute or two for the system to configure the settings.

5. On the opening splash screen, select "**Clone or check out code**".

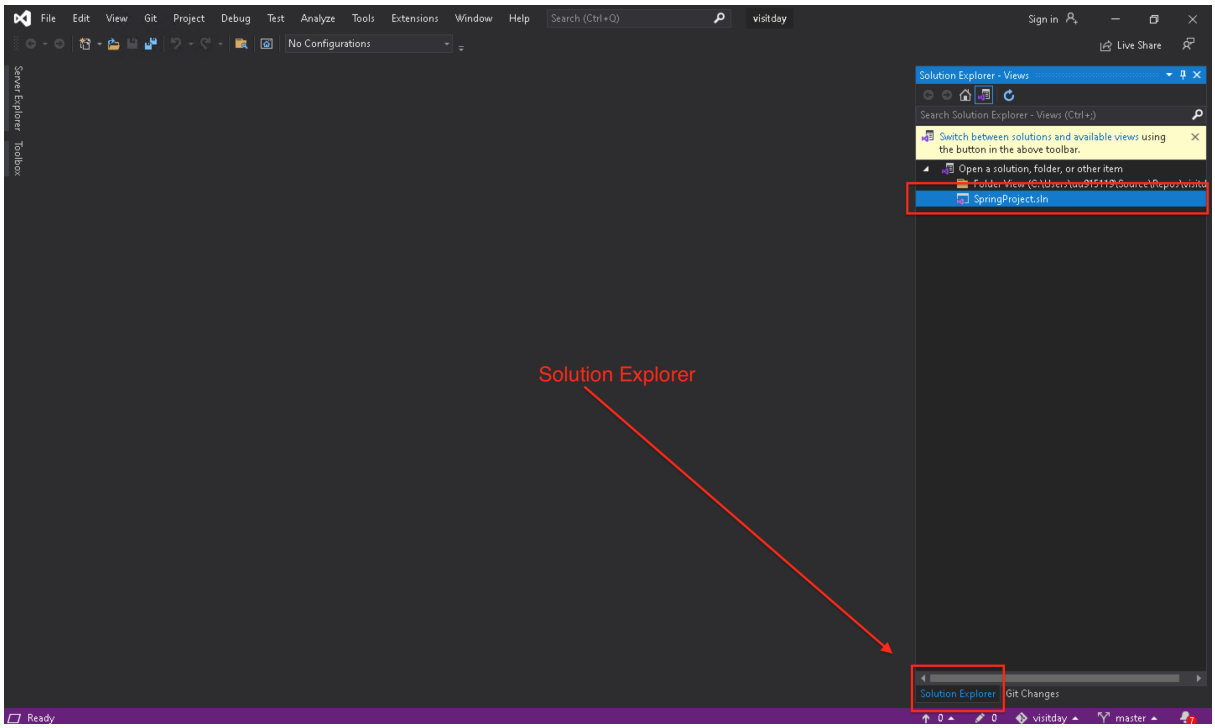


6. Input the URL: `https://csgitlab.reading.ac.uk/uu915119/visitday.git` and confirm.



7. Next, we will look at the basics of Visual Studio to understand the Main Editor and Solution Explorer. For a detailed introduction, have a look at https://learn.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?view=vs-2019 in a web browser.

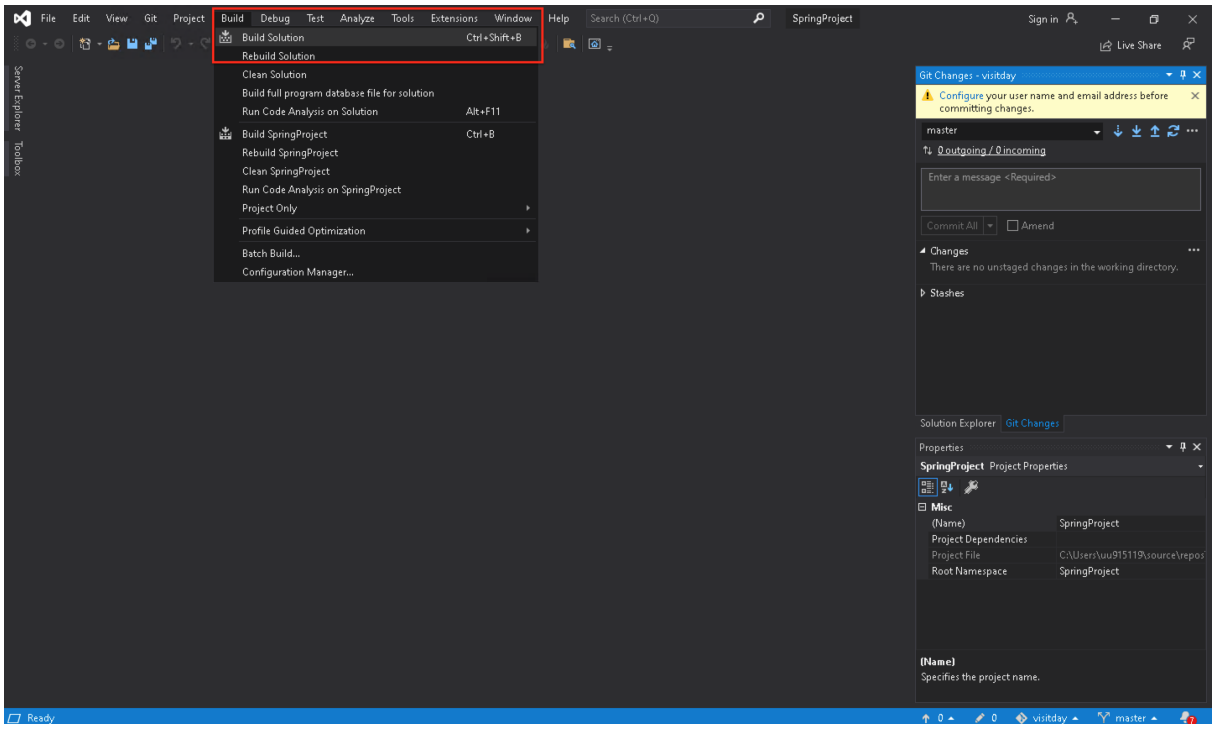
8. In the **Solution Explorer**, select the file `SpringProject.sln`; double-click on it to open the solution.

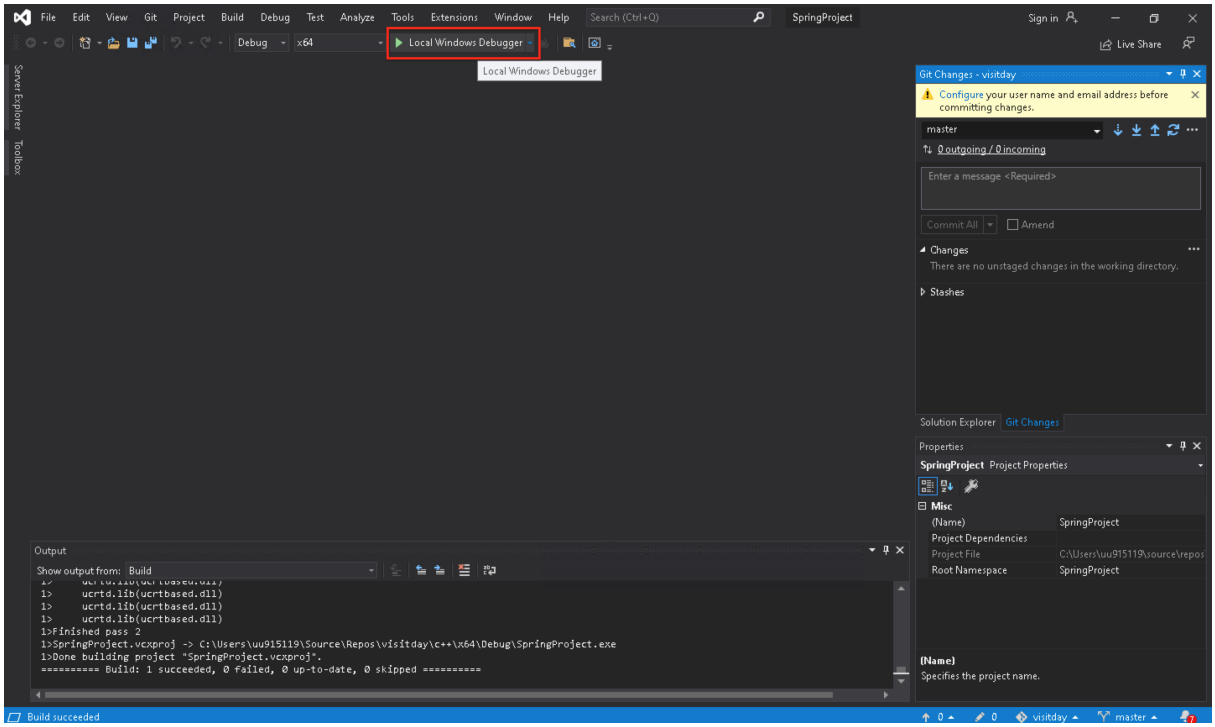


9. Compile[1] and run the program by selecting the **Build Configuration** and running the **Local Windows Debugger** at the top of the interface. The key `F5` also triggers compilation and execution.

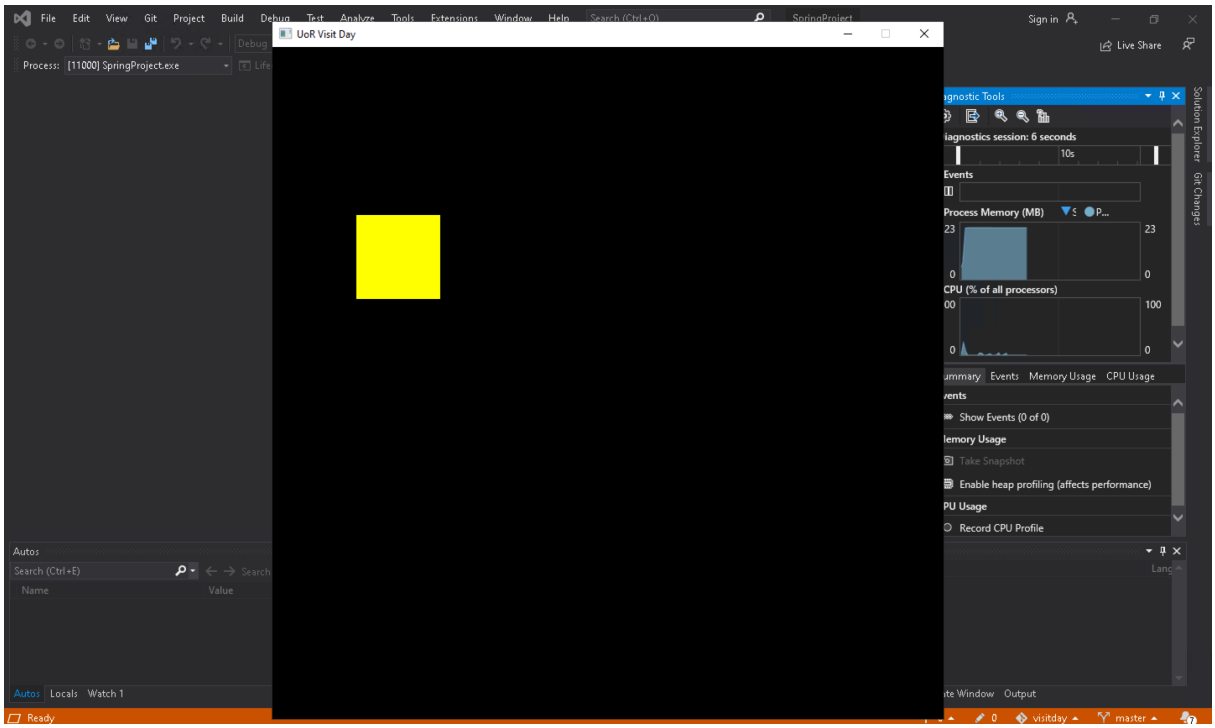
   This action will open an output interface at the bottom, in which the output of compilation is shown. Once the build is complete, a console window should open containing the expected output, and Visual Studio will switch into the **Debugger** interface. This process might take some time on the first run, so do not be surprised if it takes a while to load and start the first time you compile and run. Note: If the code contains an error and is not well formed, Visual Studio will not run the code.

[1] A compilation process generates a executable program from source code.

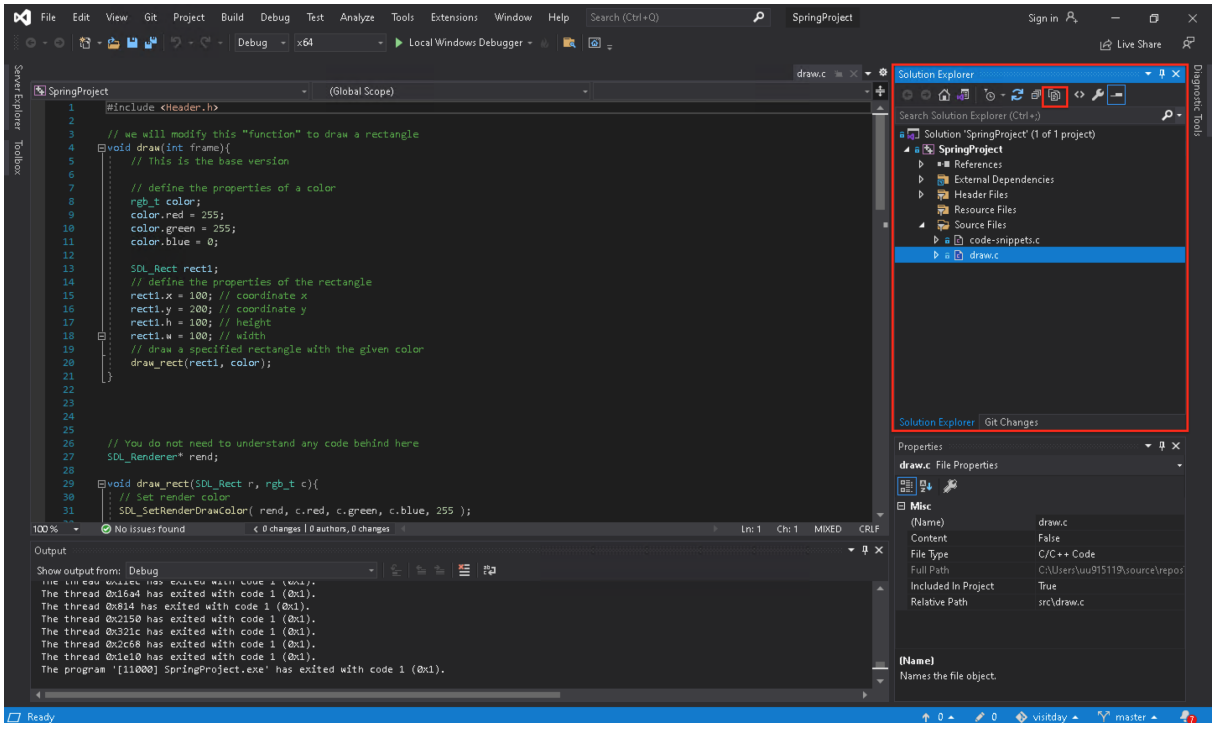




10. You should see a new window opening with a colored rectangle. This is our application. It is not very interesting yet, so just close the window of the started program.



11. We will now start to experiment by modifying the source code in the main window. First, open the file "`draw.c`" in the Solution Explorer.



12. The file contains comments starting with the characters "`//`" that describe the behavior of the program. If you scroll down a few lines you find this code:

```
3 // we will modify this "function" to draw a rectangle
4 void draw(int frame){
5   // This is the base version
6
7   // define the properties of a color
```

```
8   rgb_t color;
9   color.red = 255;
10  color.green = 255;
11  color.blue = 0;
12
13  SDL_Rect rect1;
14  // define the properties of the rectangle
15  rect1.x = 100; // coordinate x
16  rect1.y = 200; // coordinate y
17  rect1.h = 100; // height
18  rect1.w = 100; // width
19  // draw a specified rectangle with the given color
20  draw_rect(rect1, color);
21 }
```

Listing 1: `draw.c`

This code uses our concept of "**Sequence**" to run several instructions, one after the other, sequentially. It first defines a color (Lines 8-11) using the RGB color mode (0 means no color in the respective channel and 255 means maximum brightness). Then, it defines the dimensions of a rectangle (Lines 13-18). Finally, it draws the rectangle by calling the "function" `draw_rect()` in Line 20.

13. **Experiment** by changing some values in the code as you like and rerun the code to see the impact. Maybe change one of the fixed numbers to depend on the `frame` variable (an integer value) that is passed into this `draw()` function, too?

Further Reading

- Welcome to the Visual Studio IDE: https://learn.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?view=vs-2019
- Tour of Visual Studio: https://learn.microsoft.com/en-us/visualstudio/ide/quickstart-ide-orientation?view=vs-2019

2. Group Work (40 Minutes): Control Structures for Structured Programming

As part of this task, we will explore basic control structures in C that implement **selection** and **iteration**. We have already seen **sequence** in action because the program is executed step by step.

We will modify our little `draw()` function in the same project that we used before, exploring several options. There are several different ways you can achieve each goal.

Please feel free to explore and play with the code. Some steps are marked as **advanced**; only perform them if you feel comfortable. The hints section underneath provides some support.

Steps

1. Team up with one of your neighbours, and sit in front of one computer.
2. Work through the following tasks: (time: 30 minutes)

   (a) First, modify the code so that the graphics change over time. To do this, we will use the variable `frame` that contains how often the image has been re-drawn.

   After some initial experimentation, think about how you could make the picture displayed alternate between two states every other frame. To do this, you can use the C construct for **selection**, which is illustrated in the following code:

```
if ( condition ){ // if condition != 0, then
  // run this code
} else {
   // run this other code
}
```

Listing 2: C Selection Construct

So, you could create, for instance, code like this:

```
if ( 5 < 10 ){ rect1.x = 100;
} else {
rect1.x = 200;
}
```

Listing 3: Example Selection

Which means that if 5 is smaller than 10, which it is, then set the x coordinate of our rectangle to 100. The condition here is fixed but normally we use variables in the condition; here, you want to do something with the variable `frame`. Modify the if statement above to make the coordinates of the rectangle alternate between two locations depending on the frame.

(b) Try to move the rectangle from left to right (using the `frame` variable). Can you stop the rectangle from moving out of the window? **Advanced**: Can you make it bounce back and forth between the boundaries of the window?

(c) Next, try to create 5 rectangles and place them next to each other. (You can choose the layout.) This can be achieved using the `for` loop. Remember that this loop runs code for initialization, then checks the condition and, if it is true, runs the loop body. After each **iteration**, it runs the code in the **iteration** expression.

The following example runs the loop body 5 times; the variable `i` can be used to distinguish the different iterations.

```
for(int i=0; i < 5; i++){
 2 3}
// loop body
}
```

Listing 4: Example Iteration

(d) Modify your code from the previous step to give each rectangle a different color. That is, the first rectangle should have a different color than the next one and so on.

(e) Modify your code again and make the color vary depending on the frame. **Advanced**: You may see that the color is flickering. For example, if you modified the green channel, the rectangle might get more and more green and then black again. Can you change the code so that the color alternates; that is, it gets gradually more intense (say for 20 frames) and then gradually less intense again?

(f) Organize a 2D shape of 5×5 rectangles next to each other, each with a different color.

3. We will discuss the outcome of the practical in groups of 5-6 people. Reflect on the task: Where did you struggle? What did you learn? (5 minutes)

4. Group discussion: The lecturer will discuss the outcome of the practical with the class; potentially, we will look at a program and discuss it. If you feel comfortable, feel free to present one of your programs to the rest of the class.

If you are already an experienced programmer and finish early you may check out the Easter egg program Section 3.

Hints

- The following calculates the remainder of division by two: 5 % 2 (result is 1).
- The file "`src/code-snippets.c`" contains our base version of the code and potential solutions for each different step. In order to use them, replace the body of `draw()` with the content of the respective step.
- The constant `SIZE` represents the width and height of the window.
- If you receive a compilation error referring to something called "`SDL`", check that you have opened the "Solution SpringProject" and not another C file.

3. Easter Eggs: A Bouncy Reading

If you have time and feel comfortable with the groupwork, you could try some fun programs that can be easily derived from our example.

In the directory rectangle, you will find an example program with a moving logo that you can try. The program `rectangle3.c`, for example, provides a logo that bounces from the corners and `rectangle-manual.c` allows you to control its movement using the arrow keys.

In order to use one of them, you must delete `code-snippets.c` and `draw.c` in the solution explorer. Then you can add one of the C files as follows: Right click in the Solution Explorer on Source File. Select "Add", then "Existing Item". Locate the rectangle directory and add one of the files. Note that there can be only one of the rectangle examples used at a given time, as they are all complete programs. Now you should be able to compile and run these programs.