This tutorial is an example from the computer science module programming, which is a first year module. It demonstrates how the lab practicals are organized:

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

In the regular module, additional exercises for the basics would have been provided so far but we decided to jump into slightly more interesting tasks.

If you have any questions, we encourage you to approach the lecturer or the assistants, e.g., by raising your hand.

Learning Objectives

The learning objectives in the tutorial are of practical nature and aim to provide tools for you.

Tools

- Utilize Visual Studio to execute and modify a program

Theoretic

- Describe the impact 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 utilise Visual Studio to download, modify and build an existing program.

Steps

1. In the Start menu, search and start **Microsoft Visual Studio**. Once Visual Studio is loaded, you may see the information about "Connect to all your developer services", select: "Not now, maybe later" in the bottom. You may have to confirm again to start it. You will then be presented with the opening splash screen:

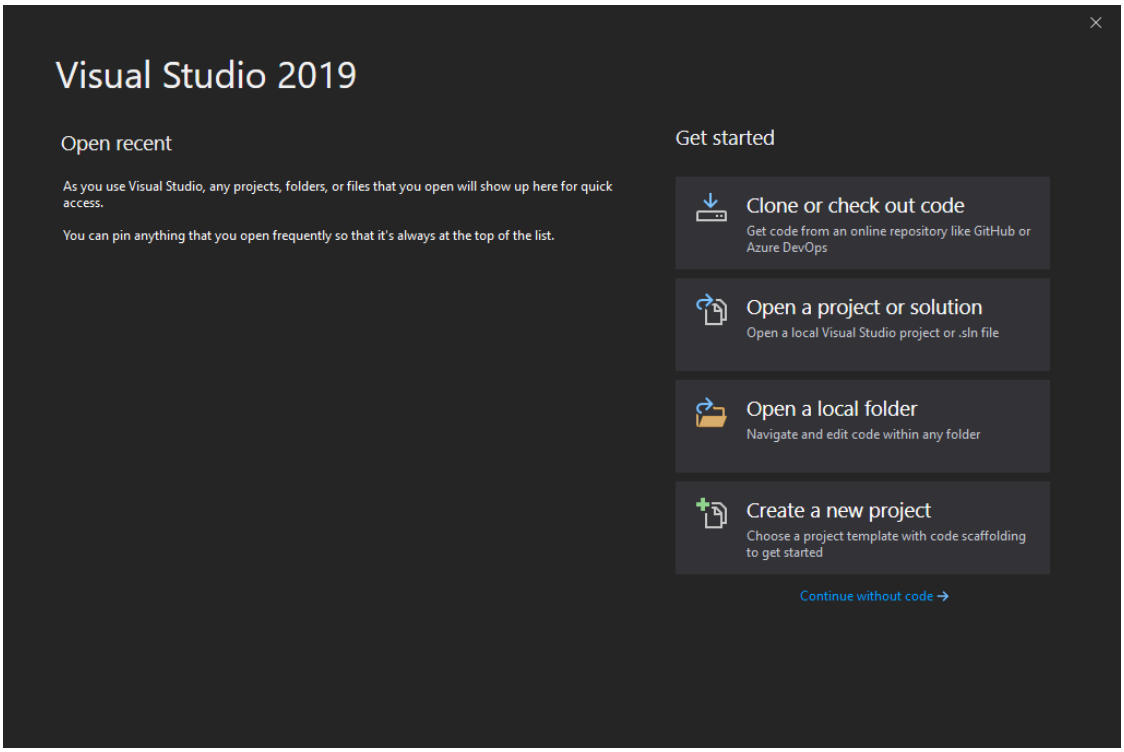

Figure 1: Microsoft Visual Studio Splash Launcher

2. Select "Clone or check out code".
3. Input the URL: *https://csgitlab.reading.ac.uk/di918039/visitday.git* and confirm.
4. Next, we will look at the basics of visual studio to understand the main editor and solution explorer. Have a look at *https://docs.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?view=vs-2019*.
5. In the **Solution Explorer**, select the file *SpringProject.sln*; a double-click will open this *solution*.
6. To compile[1] and run the program, you can select the **Build Configuration** (framed in red in Figure 2) and run the **Local Windows Debugger** at the top of the interface. The key F5 also triggers the compilation and execution.

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

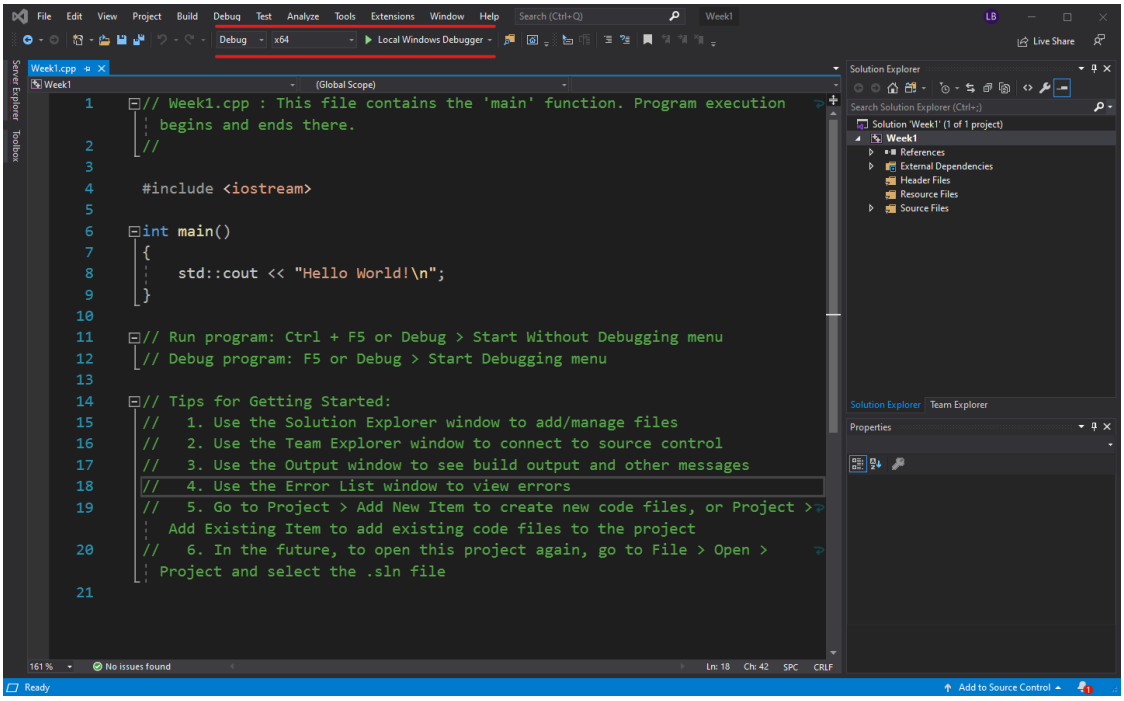

Figure 2: Running a program in Visual Studio

This action will open an output interface in the bottom, in which the output of the compilation are 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 in 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.

7. You should see a new window opening with a colored rectangle. This is our application. Next, close the window of the started program.

8. 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

9. 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:

```
1  // this "function" draws a rectangle
2  void draw(int frame){
3    // define the properties of the color used for the rectangle
4    rgb_t color;
5    color.red = 255;
6    color.green = 255;
7    color.blue = 0;
8
9    // define the properties of the rectangle
10   SDL_Rect rect1;
11   rect1.x = 100; // coordinate x
12   rect1.y = 200; // coordinate y
13   rect1.h = 100; // height
14   rect1.w = 100; // width
15   // draw the rectangle with the given properties of the dimensions and color
16   draw_rect(rect1, color);
17 }
```

This code uses our concept of "Sequence" to run several instructions after another sequentially. It defines first a color (Lines 4-7) using the RGB color mode (0 means no color in the respective channel and 255 means maximum brightness), and then the dimensions of a rectangle (Lines 10-14) and finally draws the

rectangle by calling the "function" `draw_rect()` in Line 16.

The variable `frame` contains a number that is incremented every time the image is re-drawn, that may be used to change the graphics over time.

10. Change some values as you like and rerun the code to see the impact. Maybe change one of the fixed numbers to `frame`, too?

Further Reading

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

Therefore, we will modify our little `draw()` function in the project that we used before exploring several options; you are flexible how to actually perform the goal and can still 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 neighbors that has a similar skill level, and sit in front of one computer. (We will inquire the skill level and organize the grouping.)
2. Work through the following tasks: (time: 30 minutes)

   a) First, modify the code that the graphics changes over time. Therefore, we will use the variable `frame` that contains how often the image has been re-drawn.

   After your initial testing, think about how you could make it alternate between two states every other frame. Therefore, we can use the C constructs for **selection** which is illustrated in the following code:

```
1  if ( condition ){ // if condition != 0, then
2    // run this code
3  } else {
4    // run this other code
5  }
```

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

```
1  if ( 5 < 10 ){
2    rect.x = 100;
3  } else {
4    rect.x = 200;
5  }
```

   Which means that if 5 is smaller than 10, which it is, it sets the x coordinates 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 rectangle alternate in the coordinates 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 to move out of the window? **Advanced**: Can you make it bounce back and forth between the boundaries of the window?

c) Next, we want to create 5 rectangles and place them next to each other (layout is your choice). This can be achieved using the `for` loop. We 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.

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++){
  // loop body
}
```

d) Modify the previous code to change the color for the different rectangles, i.e., the first rectangle should have a different color than the next one and so on.

e) Modify the previous code again and include varying the color depending on the frame. **Advanced:** you may see that the color is flickering, e.g., if you modify the green channel it gets more and more green and then black again. Can you change the code so that the color alternates, i.e., gets more intense (say 20 frames) and then less intense again?

f) Organize a 2D shape of 5x5 rectangles next to each other, each with a different color.

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

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

If you are already an experienced programmer and finish early you may checkout the easter egg program.

Hints

- The following calculates the remainder of the division by two: 5%2 (result is 1).
- The file "`src/code-snippets.c`" contains our base version and potential solutions for the different steps. 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.
- When you receive a compilation error pointing to some "SDL" check that have opened the "Solution SpringProject" and not another C file.

2.1 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 for a moving logo that you can try. The `rectangle3.c`, for example, provides a logo that bounces from the corners and `rectangle-manual.c` allows you to control the movement using the arrows.

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 codes. Now you should be able to compile and run these codes.