Variables part 1

1. Content

2. Objective

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

- Create a Visual Studio application

- Define and use breakpoints in Visual Studio
- Explain what the Visual Studio build does and how to solve the reported errors
- Explain what an include directive does and know which ones are necessary to use std::cin, std::cout.
- Use std::cin and std::cout for console input and output
- Define comment lines in code
- Define and initialize different types of variables: int, double, char, bool
- Create literals of each type
- Assign another value to a variable
- Use operators with these types
- Add given code file to a project
- Find and solve errors and warnings related to the above topics
- Explain the basic game loop
- Draw basic graphic shapes on the window. (lines, rectangles, ellipses)

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

3. Source material

To be able to make this lab, you will need to look up many things. These are the preferred sources.

- Video tutorial
- Slides in the pdf document
- Website: learncpp.com

4. Exercises

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

```
// Name, Firstname - 1DAEXX
```

Give your **projects** the same **name** as mentioned in the title of the exercise, e.g. **Variables**. Other names will be rejected.

4.1. Variables

Use Visual Studio to create a new **project** with name **Variables01** in your **1DAExx_01_name_firstname** folder.

Use the document in 00_General "HowToCreateProject" that describes how to create a visual studio console project. Do not do step 4 now (framework).

4.1.1. Intro

The following steps are shown and explained by the teacher in a demo, take notes if needed.

- How to create the project.
- How to build a project.
- How to declare local variables and initialize them.
- How to write to the console using the insert operator<< and std::cout.

- How to read from the console using the extract operator>> and std::cin.
- How to indicate that lines have to be considered as comment by the build process.
- How to define/delete a breakpoint and why it is useful.
- How to solve errors reported by the build process.
- How to upload your solutions.

4.1.2. Naming conventions

In every programming community, naming conventions rules define how you name your variables, functions, etc…

For programming1 and 2 these rules are described in the document 0_General/Manuals and Guides/Coding Standards.pdf.

In this lab, we will only use local variables. These are the rules you have to apply:

- Always start with a lower case letter, each sub word a upper case letter.
- No underscores are used.
- It should have a meaningful descriptive name

Examples: number, letter, numberOfEnemies

4.1.3. Defining variables without initialization

In the main function, define following variables without initializing them:

integral types: int, char, bool

floating types: float, double

Mind using the naming convention for variables.

Send their value to the console.

Compile the cpp file.

You get errors: **In C++ you always need to initialize variables, giving them an initial value.**

4.1.4. Defining with default initialization

Change the code:

- Give them a value during definition using the default {} initialization.
- See their values using a break point and stepping through the code
- See their values in the watch window of visual studio
- Show their values on the console surrounded by parenthesis.

4.1.5. Defining with literal constants initialization

Now give them a value during definition again using the {} initialization but passing a literal constant value. Show the content of the variables on the console, again surrounded by parenthesis.

More info is available on Literals

4.1.6. Assigning a literal constant using the assignment operator

You can change the content of a variable using the assignment operator. Give the variables another value using the assignment operator (=) and again show their content on the console.

4.1.7. Assigning a user value using the "get from" (>>) operator

Ask the user to enter a value for each variable type. Show the resulting value on the console.

4.2. Expressions

In the following exercises, you will build expressions using a combination of operators and different types of operands. An expression has a result and a type, and you can assign the result to a variable of that type using the assignment operator=.

4.2.1. Binary operators +, -

Binary operators require 2 operands, hence the word "binary". These operands can be literals, variables or the result of another expression.

Build expressions using these operators and send the result to the console using the send to (<<) operator. Combine several operand types:

- integer variables and literals
- float variables and literals
- char variables and literals
- mixture of char and int variables and literals

4.2.2. Binary operators *, /

Try several operand types:

- integer variables and literals
- float variables and literals
- mixture of int and float variables and literals
- check what happens when you divide an integer by a smaller integer.

4.2.3. Compound expressions

As mentioned above, an operand can be the result of an expression. This makes it possible to make combinations of expressions. Then it is also important to know the order in which to apply them, you can change the order by using parenthesis.

You can read more about this subject on Operator precedence and associativity

Make some exercises on compound expressions.

4.3. Applications

Create a new project with name **Applications** in your **1DAExx_01_name_firstname** folder.

The table below lists some practical exercises. Implement them. The left column shows the user input.

Read from console	Calculate and show on console	Screenshot

RGBA color in [0, 255] values separated by a white space character	This color in [0.0f, 1.0f] values	RGBA in [0, 255]? 255 0 100 200 RGBA in [0.0f, 1.0f]: 1 0 0.392157 0.784314
Distance in km	Distance converted to m and to cm	Distance in km? 2.5 2500 meters, 250000 cm
Angle in radians	Angle in degrees	Angle in radians? 3.1415 179.995 degrees
Angle in degrees	Angle in radians	Angle in degrees? 45 0.785398 radians
Seconds for one complete rotation of an object	Calculate angular velocity in degrees/second of the object.	Seconds of one rotation? 3 120 degrees/second
Speed (in km/h) Elapsed time	Distance in meters done after the elapsed time	Speed (km/h)? 3 Elapsed time (minutes)? 30 1500 meters
Consider an object in free fall accelerating downwards at a rate of 9.8 $m/s^{2.}$ Enter the number of seconds the object is falling	The velocity of this object after the entered number of seconds supposing there is no air resistance.	Seconds? 4 Velocity 39.2 m/sec
Radius of circle	Circumference and area	Radius of circle? 10 Circumference: 62.8319 Area: 314.159
Width and height of rectangle	Circumference and area	Width and height? 10 4 Circumference: 28 Area: 40
Base and height of triangle	Area	Base and height? 10 6 Area: 30

4.4. ErrorSolving

Create a new project with name **ErrorSolving** in your **1DAExx_01_name_firstname** folder.

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

This code contains 3 error types:

- Build errors

- Build warnings
- Runtime errors

The build process consists of at least 2 steps : compilation (creates obj files that contain the machine language instructions that correspond with your C++ code) and linking (links the obj pieces into one piece: the exe), more info: Compiling and linking

Both processes can result in errors. The problems reported by the **compilation** process lead you to the causing code line when you double click on the error report line. The **linker** process however doesn't lead you to the error in the code when double clicking on it, they are harder to solve, and you really need to read the error. This project contains 1 linker error.

Solve all build warnings and errors. Hereby proceed as follows:

1. Solve the first error
2. Then build again.
3. If the build reports errors or warnings, then start again at 1

4.5. SDLAndOpenGL

This project is an introduction to SDL and OpenGL. Two libraries that help us with keyboard and mouse input and drawing operations.

First look at the teacher demo (video) that demonstrates how to do drawing operations using our framework.

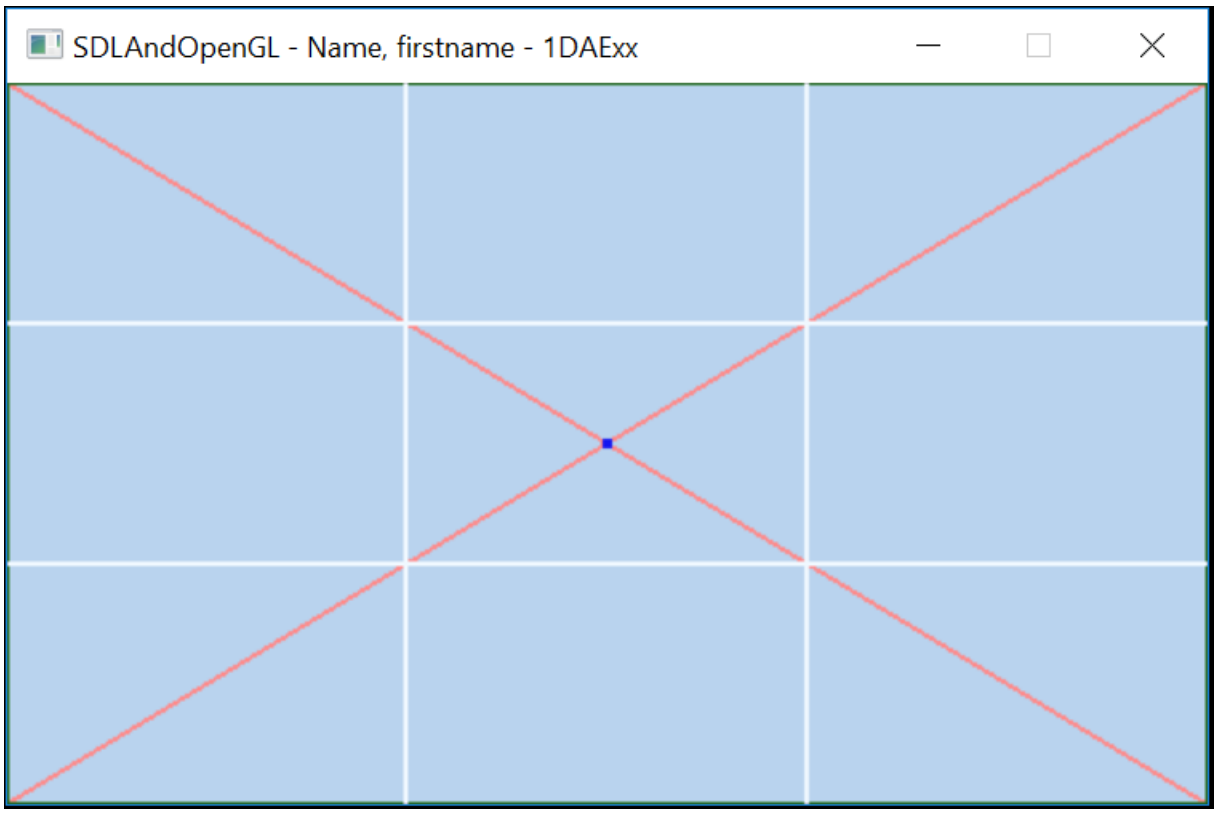
Create a new framework project with name **SDLAndOpenGL** in your **1DAExx_01_name_firstname** folder. Use the document in 00_General "HowToCreateProject" that describes how to create a visual studio console project..

Don't forget to delete the generated **SDLAndOpenGL.cpp** file.

Have a look at the code in main.cpp; Look where you can change the window title. Change the title of the window.

The teacher/video will you show how to e.g. draw a line and a rectangle.

Comment this drawing code and complete the project on your own with the necessary drawing instructions in utils.h to get next result.



Notice the:

- Gray background
- Green border
- 2 red diagonals
- 2 white vertical and 2 white horizontal lines that split the screen in 9 areas of the same size
- A blue point with size 4 in the center.

Use the following commands:

- **SetColor** to change the color to be used for the following drawings
- **DrawLine** to draw a … line!
- **DrawRect** to draw a … rectangle!

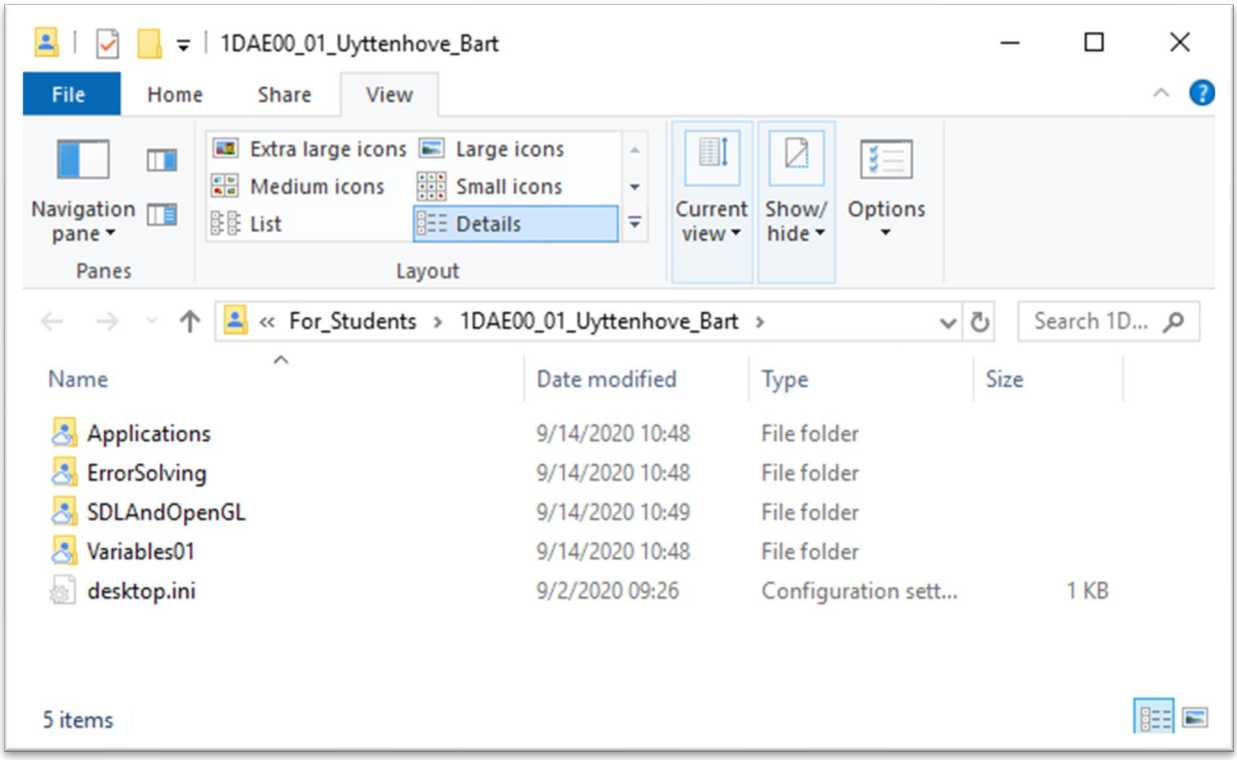
4.6. Extra – Be creative

Create a new project with name **CreativeGraphics01** in your **1DAExx_01_name_firstname** folder.

Use the provided drawing functions in utils.h. Combine your variables knowledge and drawing skills to make your first most creative drawing ever!

5. Submission instructions

You have to upload the folder *1DAExx_01_name_firstname. This is how that folder should look like:*



*However first **clean up** each project. Perform the steps below **for each project** in this folder:*

- **Close** the Visual Studio.
- **Remove** the **debug**, **x64** (if present) and the (hidden) **.vs** folder

See also the video "01_Demo_Prep_for_Upload" demonstrating this.

Compress this *1DAExx_01_name_firstname* folder to a zip file, rename the file to contain both team names:

1DAExx_01_name1_firstname1 name2_firstname2.zip

Upload it before the start of the first lab next week. You will get feedback on it.

Make sure to take the quiz. This enables you to check if you understood this first lesson!

6. References

6.1. Structure of a program

http://www.cplusplus.com/doc/tutorial/program_structure/

6.2. cout, cin, endl, std, using

A first look at cout, cin, endl, the std namespace, and using statements.

http://www.learncpp.com/cpp-tutorial/1-3a-a-first-look-at-cout-cin-endl-namespaces-and-using-statements/

6.3. Literals

http://www.learncpp.com/cpp-tutorial/28-literals/

6.4. Operator precedence and associativity

http://www.learncpp.com/cpp-tutorial/31-precedence-and-associativity/

6.5. Compiling and linking

http://www.cprogramming.com/compilingandlinking.html