

# VARIABLES PART A

## CONTENT

Objective .....	2
Source material .....	2
Exercises .....	2
Ex01:    prepare your folder structure .....	2
Folder structure .....	2
Creating your projects .....	2
Ex02:    HANDS-ON DEMO: Variables01 .....	3
project setup .....	3
Hands-on demo .....	3
Exercise: finish the demo code .....	3
Ex03:    Expressions .....	4
project setup .....	4
Binary operators +, - .....	5
Binary operators *, / .....	5
Compound expressions .....	5
Compound assignment and Unary operators .....	5
Ex04:    Applications .....	5
project setup .....	5
Ex05:    ErrorSolving .....	7
Ex06:    SDLAndOpenGL (second lab block, hands-on demo) .....	7
Ex07:    Coding Craftsmanship Challenge .....	8
Submission instructions .....	9
References .....	10
Structure of a program .....	10
cout, cin, endl, std, using .....	10
Literals .....	10
Operator precedence and associativity .....	10
Compiling and linking .....	10

## 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.

## SOURCE MATERIAL

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

- Theory slides (PDF, Leho)
- Notes taken during theory
- Website: [learncpp.com](http://learncpp.com) + cpp core guidelines (see Leho for url)

## EXERCISES

### EX01: PREPARE YOUR FOLDER STRUCTURE

By keeping all your exercises structured it will be much easier to find them again, for example when we refer to them in later exercises or you want to look something up.

---

### FOLDER STRUCTURE

Each lab you should **create a folder** with the correct name:

- **1DAEXX\_L01\_NAME\_FIRSTNAME**
  - Replace **xx** by your group number (01, 02, ..)
  - Replace **L01** with the correct lab number each week
  - All projects from this lab should be created inside this folder

---

### CREATING YOUR PROJECTS

Every lab exercise tells you which name you should use. Remember to place all of them in the folder you just created for this lab.

Have you forgotten how to create a project? Follow the **demo** at the start of the lab or find the tutorial on Leho under 00\_GENERAL.

**EX02: HANDS-ON DEMO: VARIABLES01**

---

**PROJECT SETUP**

- Create a new **CONSOLE PROJECT** in this lab's folder (see [EX0](#))
  - Call this project **Variables01**
- Your name, first name and group should be mentioned as comment at the top of each **cpp** file.

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

- Replace XX with your group number

**NAMING CONVENTIONS**

*In every programming community, naming conventions rules define how you name your variables, functions, etc... For programming1 and 2 these rules can be found in the "CODING STANDARDS" on Leho (00\_GENERAL) In this lab, we will only use **local variables** (keep them inside main). These are the rules you should apply:*

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

*Examples:        number, positionX, numberOfEnemies*

---

**HANDS-ON DEMO**

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.

---

**EXERCISE: FINISH THE DEMO CODE**

Now **continue in your Variables01 project from the demo** as instructed below. Check which steps were already covered in the demo and add / change code where needed.

**Note:** you will have to understand the difference between declaring and initializing a variable as handled in theory; take your notes with you if necessary.

---

**DEFINING/DECLARING 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 correct naming convention for variables.
- Write code to 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.**
  - Comment out (do not remove) the lines that cause issues.

---

## DEFINING/DECLARING VARIABLES WITH DEFAULT INITIALIZATION

Change the code:

- Give the variables a value during definition using the default initialization (see theory)
- Look at their values using a **break point** (!) and stepping through the code
- Look at their values in the **watch window** of visual studio
- Show their values on the **console** surrounded by parenthesis.

---

## DEFINING/DECLARING VARIABLES WITH LITERAL CONSTANTS INITIALIZATION

- Now give them a value during definition again using the {} initialization but passing a **literal** constant value.
  - More info is available on [Literals](#)
- Show the content of the variables on the console again, surrounded by parenthesis.

---

## ASSIGNING A LITERAL CONSTANT USING THE ASSIGNMENT OPERATOR

You can change the content of a variable **after** initializing it, by using the **assignment operator**.

- Give the variables another value using the assignment operator (=)
- Again, show their content on the console.

---

## ASSIGNING A USER'S INPUT VALUE USING THE "GET FROM" (>>) OPERATOR

- Ask the user to enter a value for each variable type.
- Show the resulting value on the console.
- For example, for the char type:

```
Please enter a character value:
```

→ example result after the user entered R:

```
Please enter a character value: R  
-> The character that was entered by the user is R.
```

## EX03: EXPRESSIONS

### PROJECT SETUP

- Create a new **CONSOLE PROJECT** using Windows Desktop Wizard in this lab's folder (see [EXO](#))
  - Call this project **Expressions**
- Your name, first name and group should be mentioned as comment at the top of each **cpp** file.

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

- Replace XX with your group number

*Before you start, make sure you are familiar with the terminology used in theory:  
Expressions, literals, operators and operands, assignment operator =*

- **The main goal of this part is to experiment with various expressions** in your code! Try, change, try again, use breakpoints and console inputs and outputs to test!

## 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.
- Experiment with combining several **operand** types:
  - integer variables and literalsfor example:

```
int number{ 4 }; // integer variable

//integer variable (number) + literal (9):
std::cout << "The result of number + 9 = ";
//TODO: add the correct expression in above code to show result
```

```
The result of number + 9 = 13
```

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

---

## BINARY OPERATORS \*, /

Experiment with various operand types in code:

- 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.

---

## COMPOUND EXPRESSIONS

- Make some exercises on **compound expressions** (= combination of expressions).
- Make sure you experiment with the order in which they are executed!
  - You can read more about this subject on [Operator precedence and associativity](#)

---

## COMPOUND ASSIGNMENT AND UNARY OPERATORS

- Experiment with **compound assignment operators** (see theory).
  - Which operators can use this principle?
- Experiment with **unary operators**.
  - Which operators can use this principle?
  - What happens if you use them in compound expressions?
  - Try placing the unary operators before + after the var in an expression; what's the difference?

---

## EX04: APPLICATIONS

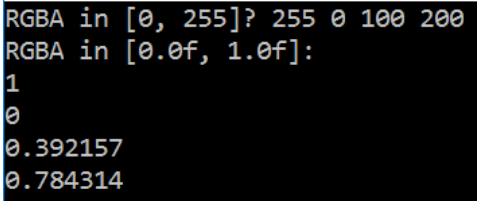
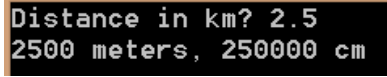
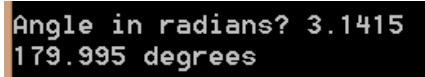
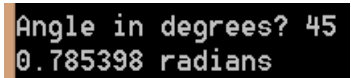

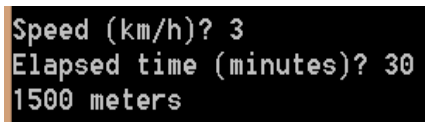
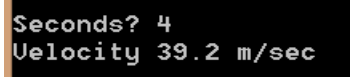
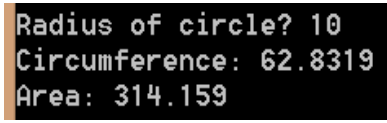
---

### PROJECT SETUP

- Create a new **CONSOLE PROJECT** using windows desktop wizard in this lab's folder (see [EX0](#))
  - Call this project **Applications**
  - Your name, first name, group should be mentioned as comment at the top of each **cpp** file.

- Download **01\_EX4.3\_Applications\_example solution.exe** on Leho and execute; this is an example of what your solution should look like
- For example for the first exercise, the user must enter 4 values in the range 0 – 255, separated by a space or enter.
  - In the screenshot below, the user entered the values 255, 0, 100 and 200 (these are not literals but user input!)

The table below lists some practical exercises. Implement them:

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	
Distance in km	Distance converted to m and to cm	
Angle in radians	Angle in degrees	
Angle in degrees	Angle in radians	
Seconds for one complete rotation of an object	Calculate angular velocity in degrees/second of the object.	
Speed (in km/h) Elapsed time	Distance in meters done after the elapsed time	
Consider an object in free fall accelerating downwards at a rate of 9.8 m/s <sup>2</sup> .  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.	
Radius of circle	Circumference and area	

Width and height of rectangle	Circumference and area	<pre>Width and height? 10 4 Circumference: 28 Area: 40</pre>
Base and height of triangle	Area	<pre>Base and height? 10 6 Area: 30</pre>

## EX05: ERRORSOLVING

Create a new **CONSOLE PROJECT** called **ErrorSolving** in your `1DAEXX_01_NAME_FIRSTNAME` folder.

Open the folder of your project in windows explorer and overwrite the generated file `ErrorSolving.cpp` by the given **ERRORSOLVING.CPP** file. Click “reload all” if Visual Studio asks you to.

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

## EX06: SDLANDOPENGL (SECOND LAB BLOCK, HANDS-ON DEMO)

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

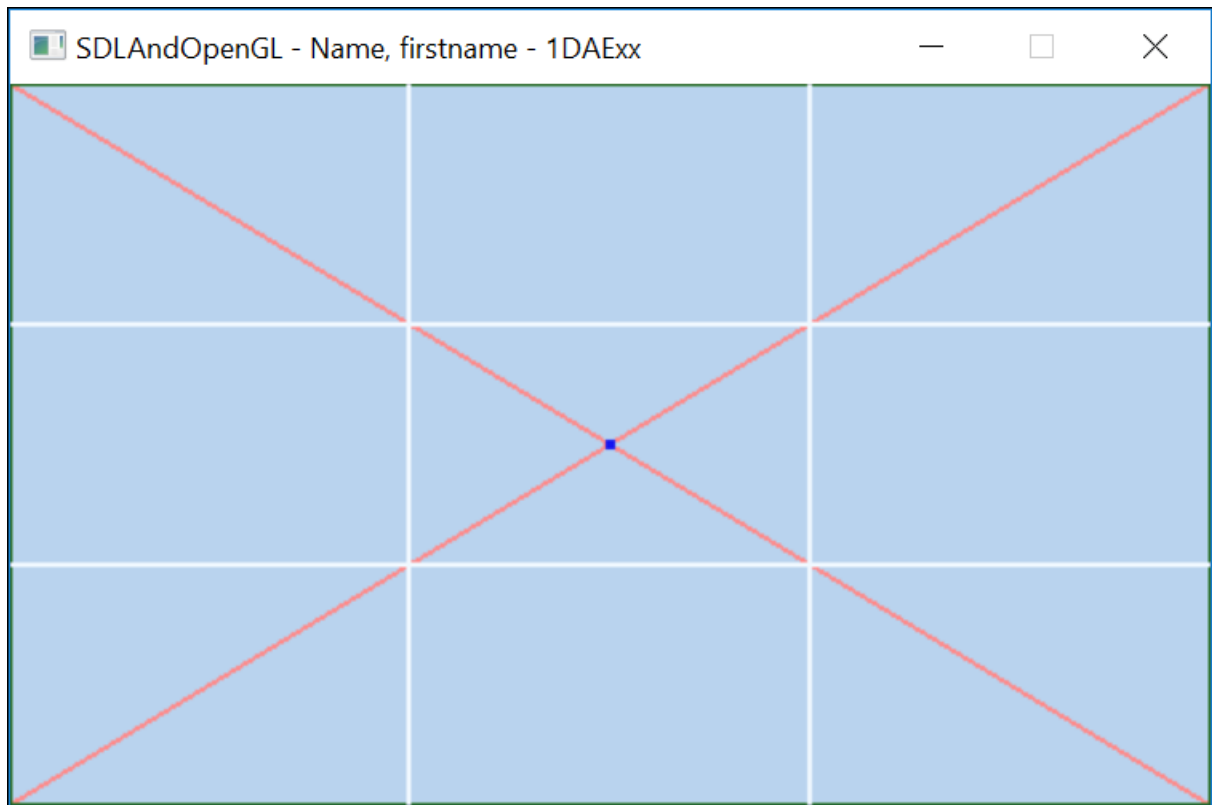
**First look at the teacher demo that demonstrates how to do drawing operations using our framework.**

Create a new **Prog1 Game Project** with name `SdlAndOpenGL` in your `1DAEXX_01_NAME_FIRSTNAME` folder. Check the instructions under **MANUALS & GUIDES** under **00\_GENERAL** on Leho if you do not know how to do this.

Try to find where you can change the window title and size. Change both.

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

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



Notice the:

- Grayish 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

#### EX07: CODING CRAFTSMANSHIP CHALLENGE

*Level up your game development powers! This is where creativity meets code, and it bridges the gap between the traditional step-by-step written lab assignments and a much deeper understanding of the material.*

*In this weekly exercise you're tasked with inventing an exercise that merges all c++ programming and game development skills you obtained so far in this course into a (small!) unique project. Let your imagination run wild and work on a programming project YOU love!*

*Do not underestimate the value of this exercise; experimenting is by far the best way to master programming!*

Create a **Prog1 Game Project** called CodingCraftmanship01 in the 1DAEXX\_01\_NAME\_FIRSTNAME folder.

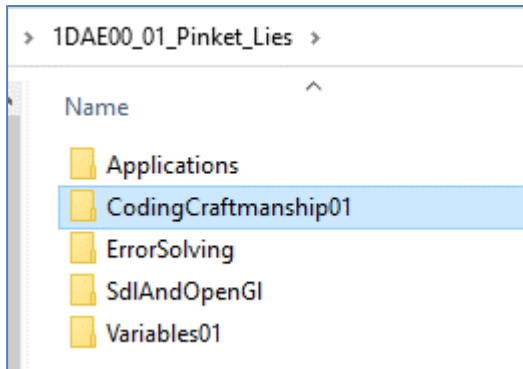
Use the provided **drawing functions in utils.h**. Combine your **variables** knowledge and drawing skills to make your first most creative drawing ever. Make sure you **use the window dimensions** so that your drawing resizes with your window size!

*Pssst: no inspiration? Google 'daily minimal' images, or get inspired by your favorite game (characters).*



## SUBMISSION INSTRUCTIONS

You have to upload the folder `1DAExx_01_name_firstname`. This is what that folder structure should look like:



- **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
- Compress this `1DAExx_01_name_firstname` folder to a zip or rar file.
- 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!

## REFERENCES

### STRUCTURE OF A PROGRAM

[http://www.cplusplus.com/doc/tutorial/program\\_structure/](http://www.cplusplus.com/doc/tutorial/program_structure/)

### 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/>

### LITERALS

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

### OPERATOR PRECEDENCE AND ASSOCIATIVITY

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

### COMPILING AND LINKING

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