
VELDI
KOMPETENS
VELDI

Introduction to C++

Part 1

-

Introduction

VELDI

History of C++

- Created by Bjarne Stroustrup from Denmark
- Initially created as ”C with Classes”
- Initial official release 1998
- Standards: (C++98,C++03,C++11,C++14,C++17,C++20)
- Used for example:
  - Software application
  - Embedded development
  - Mobile phone development
  - Operating system development
  - Etc.

Why should I learn C++ ?

- Together with C it is still the kings in embedded development.
- Can do powerful Application development
- Has a lot libraries and SDK available
  - Boost
  - QT
  - OpenCV
  - MATLAB
  - AWS
  - Etc.
- Can do low level coding for controlling hardware but also high lever coding in user interface level

VELDI

Course outline week 1

- **Week 1**
  - Introduction
  - First program ”Hello world”
  - Integer Datatype
  - ”if” and ”else” statement
  - IDE
- **Week 2**
  - Datatypes continued
  - Namespace
  - For and while loops
  - Switch and jump statements
  - Arrays
- **Week 3**
  - Functions
  - Pointers
  - Exceptions
  - Lists
- **Week 4**
  - Preprocessor
  - Classes and Objects
  - Constructor and Destructor
  - Class methods
  - Class inheritance
- **Week 5**
  - Dynamic memory
  - File handling
  - Multiple files and headers
  - Libraries

VELDI

Getting started with C++

- Compile time (not runtime) language
  - Can be compiled to almost all platforms
- Case sensitive
- Can mix C and C++ code
- High level language with lots of libraries
- Easy to use low-level features
- Starts from ”main” function

```
int main()
```

Course information

- The course will contain learning material, exercises and mandatory assignments
- The deadline for each assignment are on Sundays at 23:59
- The optional assignments are for those who want to deepen their C++ skills
- The lectures will be recorded and available afterwards
- Estimated own work time 2,5 – 4 hours/week

Schedule

- Five weeks starting today
- Live sessions via Zoom on Mondays 09:00-12:00
- Optional Q/A sessions on Wednesdays 09:00-10:00
- Optional Q/A sessions on Thursdays 13:00-14:00

VELDI

References and documents

- There is a lot of documentation available
- Books
  - There are a lot of books available
  - Might be hard to choose, depends a lot of what you are going to use C++ for, hardware, application, mobile development etc.
- Online documentation
  - http://www.cplusplus.com/
  - https://www.w3schools.com/cpp/
  - Etc.

C++ tools

- A lot of tools exists even online
- Online variants
  - www.onlinegdb.com (supports multiple files and debugging)
  - www.w3school.com (also good references and help)
  - www.cplusplus.com (also good references and help)
  - Etc.
- Variants to be run on a computer
  - Eclipse with Gnu Compiler Collection
  - Code::Blocks
  - Etc.
- Preferred IDE to install on your computer

  www.codeblocks.org (Recommended: *mingw-nosetup.zip)

VELDI

Introduction to C++

Part 2

-

Hello World

VELDI

Syntax of a basic program

- First the “#includes” appears telling the compiler to include that function, initially we uses <iostream>
- Then we need to have a main function that will be called at start. “int main()”
- Empty data in the () means that we have no argument to this function in this case.
- We need to have a beginning bracket “{“ and an ending bracket “}”
- The main function needs to be of type int and return something to the operative system normally 0 telling ok, if you don’t enter any return-statement in main 0 will be default

Comments

- We will here focus on 2 comment type
- Line comment – uses “//” and will be valid on the given line until a line break appear
- Block comment – uses “/*” in the beginning of the block and “*/” at the end of the block

```
// This is a line comment, valid until a line break appear
int i=0;

/*
 This is
 a
 block comment
*/

return 0;
```

Scope

- To set the scope of a block, you need to have the “{“ at the beginning and a “}” at the end
- If no scope are given the scope will be one line
- Will be required for example in the main() function to tell the scope where it starts and where it ends

```
int main()      //main function will require "{" and "}"
{               //beginning of scope
    int i=0;
    i++;
    i--;
    return i;
}               //end of scope
```

Standard library <iostream>

- Will be used to print and read from command prompt
- Called by the “std::”
- ”cout”
  - Standard output stream
  - Can be used to print data to the terminal
  - Called by std::cout and << and a ; at the end
- ”cin”
  - Standard input stream
  - Can be used to read data from the keyboard
  - Called by std::cin and >> and a ; at the end
  - Sometimes a std::cin.sync() can be good to sync and clear old buffer before

Developing your very first C++ program

```
#include <iostream>

int main()
{
     std::cout << "Hello World";
     return 0;
}
```

“#include <iostream>”	- Required for “cout” printing
“int main”	- Here the program starts
“{“ and “}”	- give the “scope” of main function
“return 0”	- return “0”, since we have “int” type we shall return something.
std::cout << “Hello World”;	- print out text

- **Formatting can differ a bit, below is ok but looks quite bad**

```
#include <iostream>
int main(){std::cout << "Hello World";return 0;}
```

VELOI

Introduction to C++
Part 3
-
Integer data types

VELDI

Variables… what is that?

- Variables are used for storing information
- You can not use spaces in variable name, underscore can be used instead
- You can mix upper and lower characters in the name
- Example:

  ”int my_variable”

  ”int anotherVariable”
- Note: ”int Cow” and ”int cow” are 2 different variables

Integer Data Type

- **Integers**
  - Real numbers, positive and negative.
  - Can have prefix “signed” or “unsigned”
  - Signed can handle both positive and negative values (default)
  - Unsigned can handle only positive values
  - Declares with the keyword “int”

- **Declaration example**

```
int foo = 12;    //create variable foo and give it value 12
```

Working with integers

- Addition and subtraction
- A = B + C
- A += B means that A = A + B
  - visitors = 5; (5)
  - visitors = visitors + 1; (6)
  - visitors += 1; (7)
- “++” and “--” (Increase by one or decrease by one)
  - visitors++; (8) (Increase by one after)
  - --visitors; (7) (first decrease by one)

Working with integers

- Multiplication
- A = B * C
- A *= B means that A = A * B

```
room_width = 10;

room_lenght = 5;

room_area = room_width * room length;          (50)

teddy_bears = 5;

teddy_bears *= teddy_bears;                    (25)
```

Working with integers

- **Division**

  One important thing to know is that a result of a division will always round down

  - `toys = 9;`
  - `max_toys_in_box = 5;`
  - `required_boxes = toys / max_toys_in_boxes;  //(PROBLEM !!, will be 1, but we need 2 boxes)`

- **Modulo**

  The problem above can be solved by also using the modulo operator “%”

  It returns the remainder of a division of 2 values

  - `toys = 9;`
  - `max_toys_in_box = 5;`
  - `Modulo_boxes = toys % max_toys_in_boxes;  //will be 4 and since <>0 we need another box`

Working with integers

- **Copying values**

  Examples:

  - mugs = 5; (mugs 5, spoons (not set))
  - spoons = mugs; (mugs 5, spoons 5)
  - mugs -= 3; (mugs 2, spoons 5)
  - mugs = --spoons; (mugs 4, spoons 4) (NOTE mugs AND spoons changes)

Assigning variables

```
int a;
a=2;
a+=2;
a*=2;
a/=2;
```

- =
  - Assigns the value on the right side to the variable on the left side, example (variable=2)
- +=
  - Adds the value from the right side and assign it the variable on the left side, example (variable+=2) same as (variable=variable+2)
- *=
  - Multiply with the value on the right side and assigns the result to the variable on the left side, example (variable*=2) same as (variable=variable*2)
- /=
  - Divides the value on the left with the value on the right and assign it to the variable on the left, example (variable/=2) same as (variable=variable/2)

VELDI

Datatypes

- **Datatypes can be directly declared at initialization, or later in the code, until then it is uninitialized (unknown) value**
- **A good thing to do is to always declare variable value at initialization, since you might end up in a situation where you use an uninitialized variable**

```
#include <iostream>

int main()
{
    int i = 2;        //Initialized to 2
    int j;            //uninitialized, value unknown

    return i+j;       //return 2 + (unknown value)
}
```

- **The compiler will hopefully hint you something like:**

  “'j' is used uninitialized in this function”

Introduction to C++

Part 4

-

If and else statements

VELDI

Comparison Operators

- ==
  - If both sides are equal the condition will be True
  - Important to not use a single =, Then you assign a value
- !=
  - If the sides is not equal, the condition is True
- <
  - When value on left is less than right it will return True
- >
  - When value on left is bigger than right it will return True
- <=
  - If the left side is less or equal to the right side, the condition is True
- >=
  - If the left side is bigger or equal to the right side, the condition is True

VELDI

If statement

- Needs a condition to validate
- The condition will either be true or false
- If the condition is true, it will run the code inside the statement.

```
int number_of_volvos = 5;
int number_of_teslas = 6;

if (number_of_volvos < number_of_teslas)
{
    std::cout << "You should buy more Volvos";
}
```

```
Output You should buy more Volvos
```

else statement

- Should not have any condition to validate
- Will be run if the above if-statement is false

```
int number_of_volvos = 8;
int number_of_teslas = 6;

if (number_of_volvos < number_of_teslas)
{
    std::cout << "You should buy more Volvos";
}
else
{
    std::cout << "You have enough of Volvos for now";
}
```

```
Output You have enough of Volvos for now
```

else if statement

- **Will only be run if the above if statement is false and the condition in the elseif statement is true**

```
int number_of_volvos = 8;
int number_of_teslas = 8;

if (number_of_volvos < number_of_teslas)
{
     std::cout << "You should buy more Volvos";
}
else if (number_of_volvos == number_of_teslas)
{
     std::cout << "You have the same number of Teslas and Volvos";
}
else
{
     std::cout << "You have enough of Volvos for now";
}
```

```
Output You have the same number of Teslas and Volvos
```

Single line statement

- If you only have one line in the statement you can skip the “{“ and “}”

```
int number_of_volvos = 8;
int number_of_teslas = 8;

if (number_of_volvos < number_of_teslas)
     std::cout << "You should buy more Volvos";
else if (number_of_volvos == number_of_teslas)
     std::cout << "You have the same number of Teslas and Volvos";
else
     std::cout << "You have enough of Volvos for now";
```

- But use this with care, recommended is to always use the “{“ and “}” in statements
- If you enter 2 lines the secons line will be out of the statement scope

Logical operators

- && The AND operand. If both statements are true, the condition will be true
- || The OR operand. If either statement is true, the condition will be true.
- ! The NOT operand. If the statement is false, the condition will be true
- ^ Bitwise exclusive OR. Return true if only one of the statements are true

```
int number_of_volvos = 6;
int number_of_teslas = 0;

if (number_of_volvos && number_of_teslas) {
     //This will be false, due to no Teslas
}
if (number_of_volvos || number_of_teslas) {
     //This will be true, because we have Volvos
}
if (!number_of_teslas) {
     //This NOT will be true, due to no Teslas
}
```

VELOI

Be careful to not declare with “=“

- If you only have one “=“ in a if statement you will declare the variable and the condition will always be true

```
int volvos = 5;
int teslas = 6;

if (volvos = teslas)
{
    std::cout << "You have now declared volvos to teslas";
}
```

```
Output You have now declared volvos to teslas
```

- Variable volvos will now have same value as teslas =6

Introduction to C++

Code::Blocks

-

Basic usage of IDE

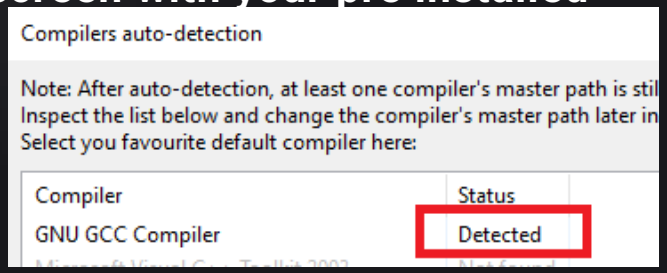
VELDI

What is an IDE?

- IDE – Integrated Development Environment
- An IDE is a program that you can help you to create and develop your program
- You create the code in the Editor
- You Compile the code with the built in compiler
- You can run your created program
- You can also debug the behaviour of your program in the IDE
- Modern IDE:s can also direct in the editor show if you write wrong code. And can help you to write the code with “content assist”

Code::Blocks IDE

- If you are “none admin” on your computer you can use the portable version with mingw installed, just unzip it and then run the “CbLauncher.exe”
- First time you will be given a startup screen with your pre installed gcc compiler detected

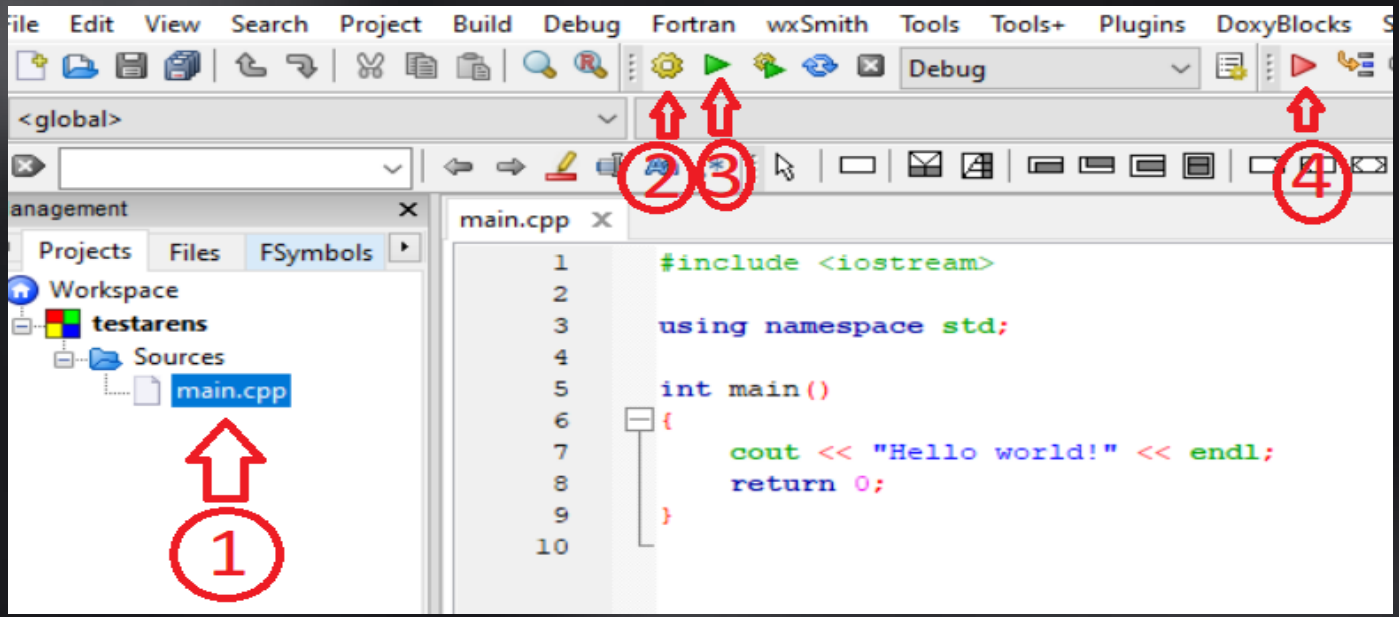


- Then a file association dialog will appear, here you can choose the best option for you
- To start a new project select “File->New->Project…” and as default select a “Console application”
- In the setup guide select project title, path (Project filename will then be auto filled in), then just press Finish

VELDI

Code::Blocks IDE..

- (1) This is the default main.cpp file
- (2) This is the build button, to compile the project
- (3) This is the RUN button to run project
- (4) This is the button to press to start debug a project



Code::Blocks IDE....

- If your project are not there the next time you start up Code::Blocks just select “File->Open” and select the project file you created. Will end with “.cbp” as extension

- You can also select “File->Recent projects” and your project file should hopefully be there also

- For more info navigate to www.codeblocks.org and read the documentation there


veldikompetens.se
Thank you!
VELDI KOMPETENS