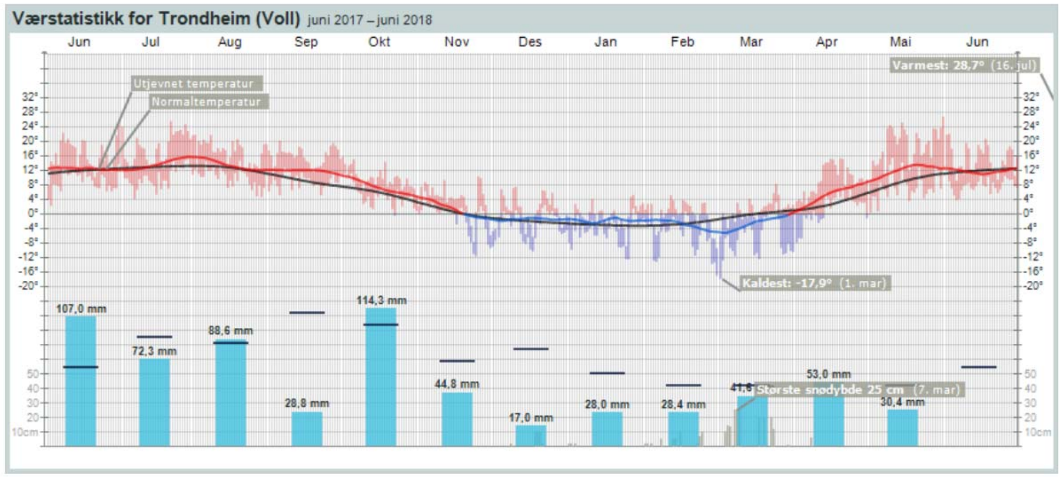
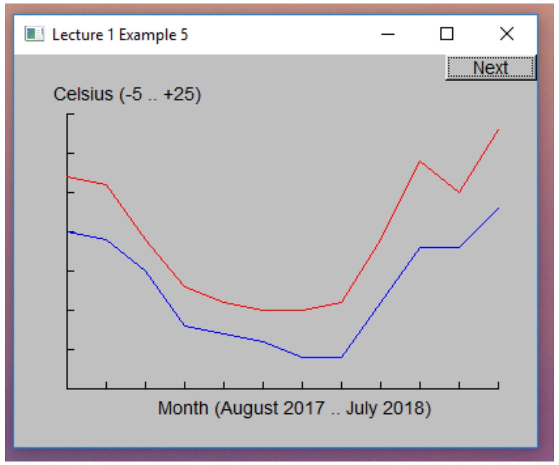
TDT4102 - Procedural and Object-Oriented Programming

Course Introduction, Some C++ fundamentals, vector and «Hello Graphical World!»

```
// two vectors of max and min temperatures in Trondheim
vector<int> maxTemp{ 17, 16, 9, 3, 1, 0, 0, 1, 9, 19, 1
vector<int> minTemp{ 10, 9, 5, -2, -3, -4, -6, -6, 1, 8,
```





Norwegian University of Science and Technology

NTNU

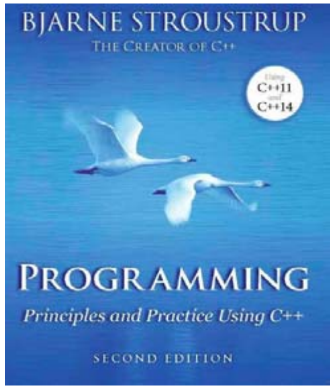
TDT4102: Norsk ~~eller~~/og engelsk, forklaring

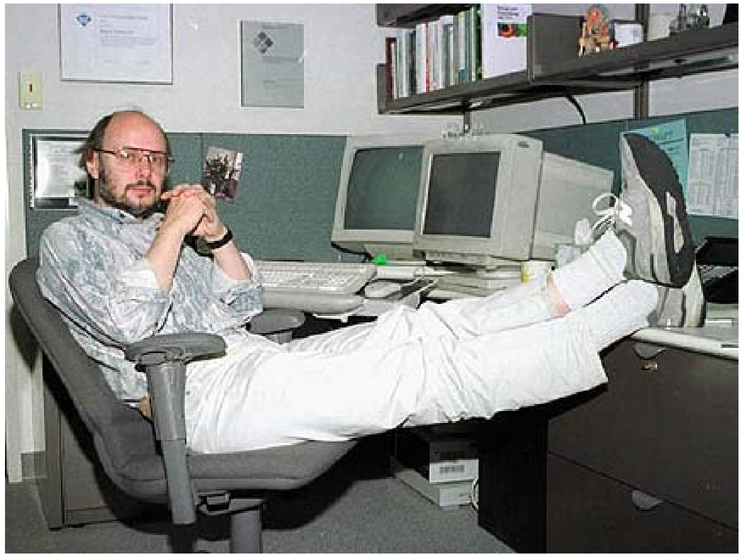
- Engelsk lærebok (vanlig ved NTNU)
- Engelske variabelnavn i kode (ganske vanlig, smak)
- Norsk undervisning, forelesninger, hjelp, **blackboard**
- Eksamen på bokmål, nynorsk og engelsk
- "Lysark" (slides)
  - Engelsk eksternt (feedback, ambisjoner)
  - Engelsk med terminologi og utdyping på norsk til våre studenter
    - Bare på blackboard (BB)

<u>Totally renewed course!</u>

- **New and different textbook**
  - By Bjarne Stroustrup, the inventor of C++
- **<u>Graphics is here now !</u>**
- **User interfaces (GUI)**
- **More modern C++**
- **Renewed exercises**

- **<u>And the only textbook</u>**
  <u>you can bring at the final exam</u>





Overview — Lecture 1

- Practical course introduction (in Norwegian, only locally)
- Brief introduction to **computer and programming fundamentals**
- Programming environments, tools
- Motivation
- Some **simple C++ example programs** explained
- *Introduction* to some **C++ fundamentals**
  - Data types, numbers, text strings, loops, functions, vector (a «collection»)
- Initialization of variables

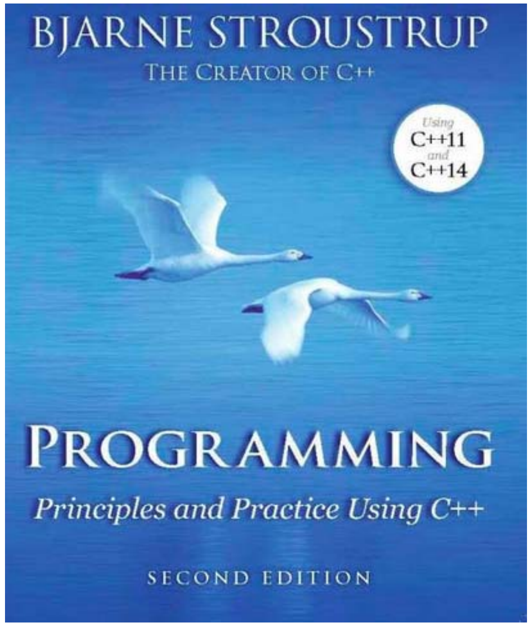
Prepping for exercise 0, 1 and 2

About the course textbook

- **Programming: Principles and Practice Using C++**
  2nd edition
  by *Bjarne Stroustrup*
  Addison-Wesley 2014.

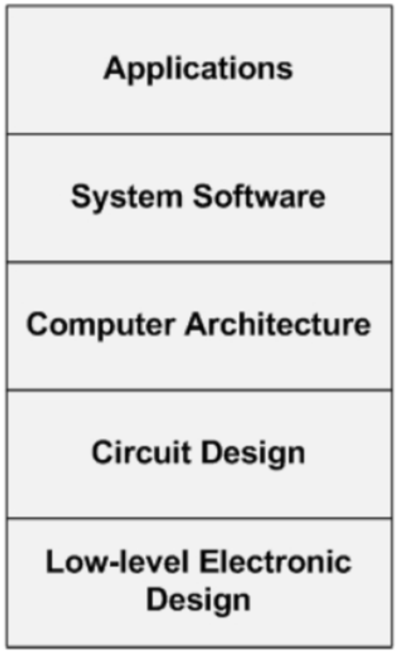
- Some of the slides in this course are based on the authors slides available from the textbook webpage. **Many thanks!**

- Students in course TDT4102 should follow the instructions given at Blackboard and do not need to study all details at the textbook webpage

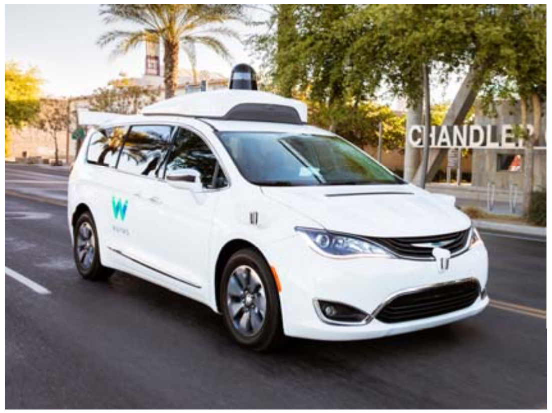


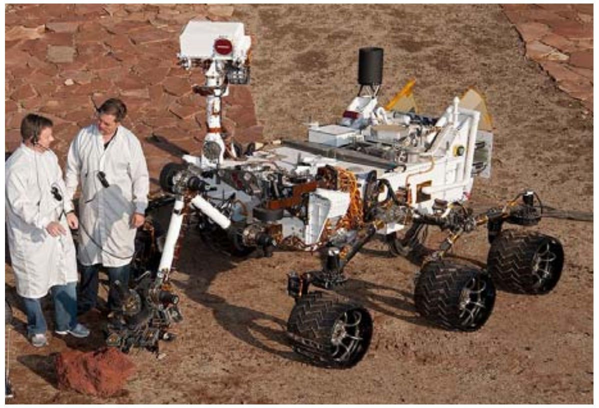
Computer fundamentals

- Types of computers
- Computer hardware (HW)
  - Levels: Architecture, design, logic, gates, electronics, physics
- Computer software (SW)
  - Languages, levels
  - Types of SW
    - Driver, library, language, tool, framework, app, cloud, …
  - Portability
    - Windows, Mac, Unix (Linux), …



Types of Computers and Applications

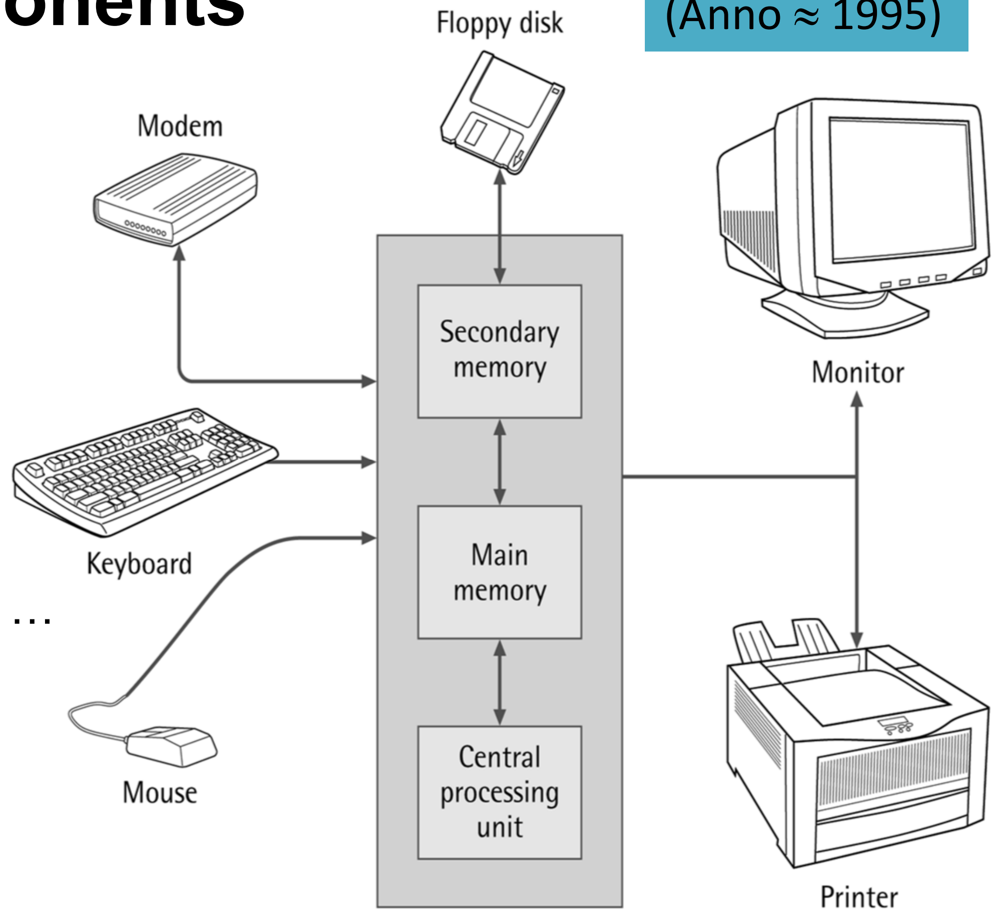




Main Computer Components

(Anno ≈ 1995)

- Central Processing Unit (CPU) (main processor)
- Memory (hierarchy)
  - Registers (in CPU)
  - Cache (1 or more levels)
  - Main memory (RAM)
  - Secondary Memory (disc, SSD)
- Input/output (IO)
  - Peripheral units
  - Keyboard, mouse, touch-screen, …
  - Network connection (Modem …)
  - Monitor (Terminal, screen)
  - <u>**Graphical User Interface (GUI)**</u>
- Software almost everywhere!!!



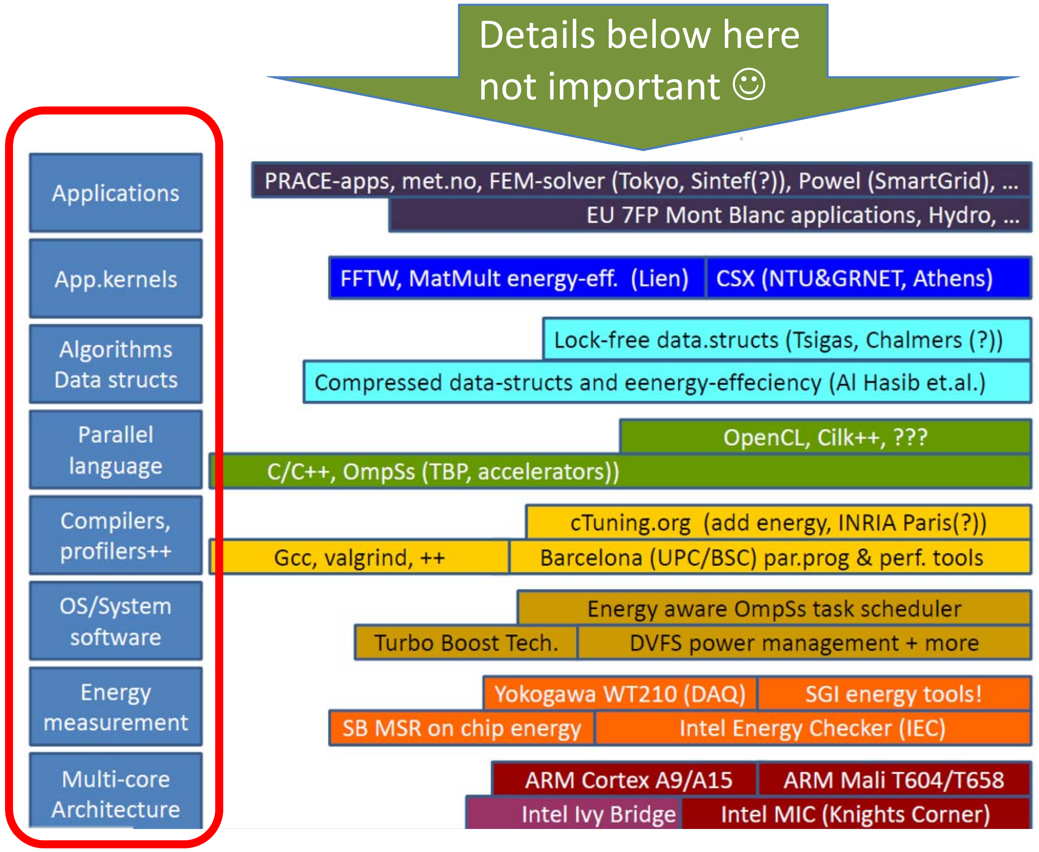
Motivation to learn programming

- Our civilization runs on software
  - Wake-up, breakfast (radio/newspaper), bus-ticket, paying for a coffee, finding the lecture hall etc…
- Most programs do not run on things that look like a PC
- Most engineering activities involve software
- C++ is the most widely used language in engineering areas

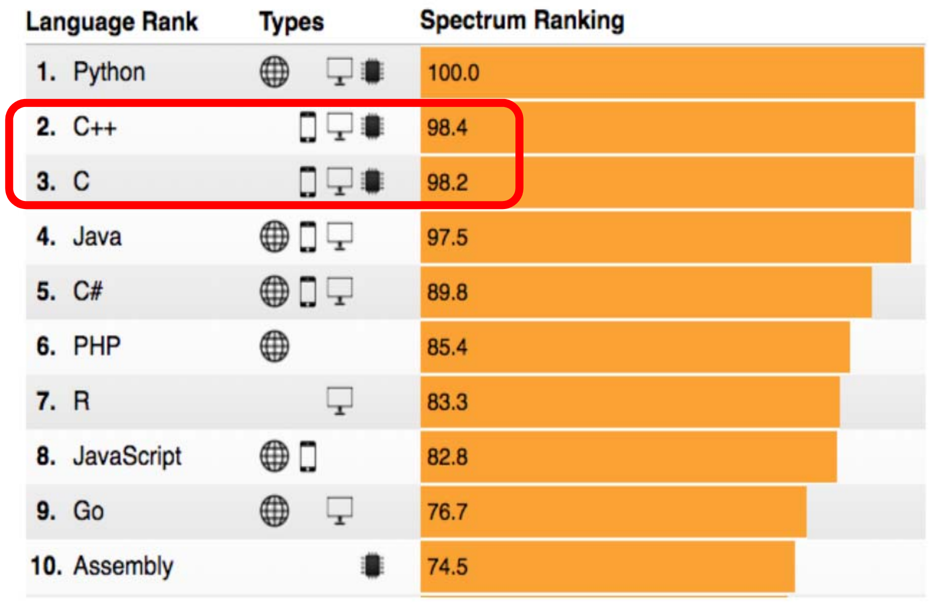
- See also: www.stroustrup.com/applications.html

Computer software: levels and complexity

- SW complexity is huge and increasing
  - ➔ Software layers
  - ➔ Modularization
  - ➔ Collaboration
    ➔ Readable code
  - ➔ Maintainable code
- "Programming is understanding"
  - When you can program a task, you understand it
  - When you program, you spend significant time trying to understand the task you want to automate
- This course is «programming in the small»
  - But we will teach good some good habits for «programming in the large»


Details below here not important ☺
Applications
PRACE-apps, met.no, FEM-solver (Tokyo, Sintef(?)), Powel (SmartGrid), ...
EU 7FP Mont Blanc applications, Hydro, ...
App.kernels
FFTW, MatMult energy-eff. (Lien)
CSX (NTU&GRNET, Athens)
Algorithms Data structs
Lock-free data.structs (Tsigas, Chalmers (?))
Compressed data-structs and eenergy-effeciency (Al Hasib et.al.)
Parallel language
OpenCL, Cilk++, ???
C/C++, OmpSs (TBP, accelerators))
Compilers, profilers++
cTuning.org (add energy, INRIA Paris(?))
Gcc, valgrind, ++
Barcelona (UPC/BSC) par.prog & perf. tools
OS/System software
Energy aware OmpSs task scheduler
Turbo Boost Tech.
DVFS power management + more
Energy measurement
Yokogawa WT210 (DAQ)
SGI energy tools!
SB MSR on chip energy
Intel Energy Checker (IEC)
Multi-core Architecture
ARM Cortex A9/A15
ARM Mali T604/T658
Intel Ivy Bridge
Intel MIC (Knights Corner)

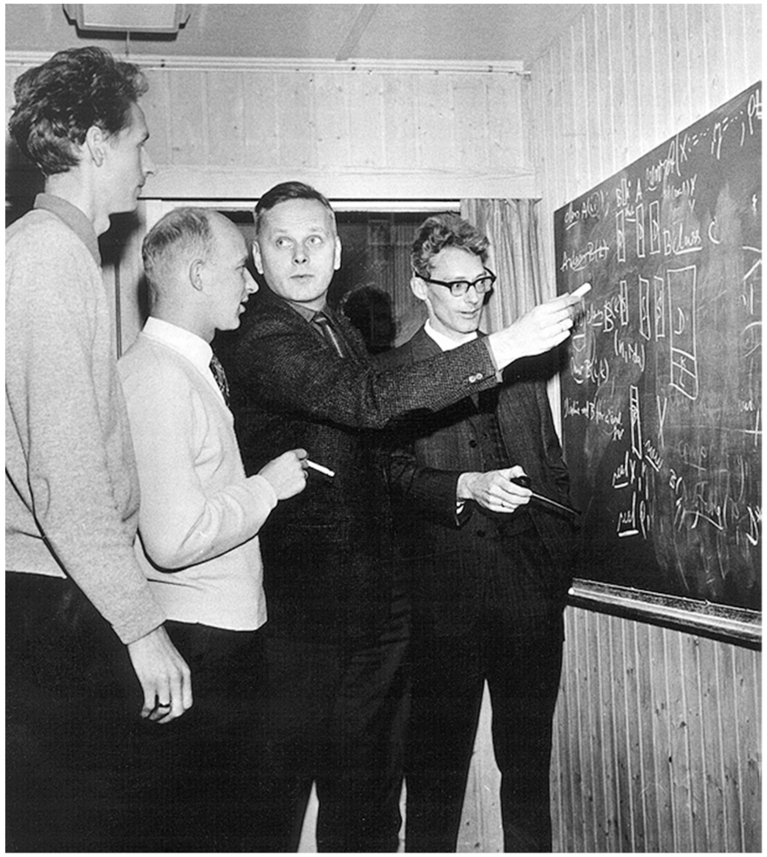
Why C++?



Language Rank	Types	Spectrum Ranking
1. Python	🌐 🖥 ▪	100.0
2. C++	📱 🖥 ▪	98.4
3. C	📱 🖥 ▪	98.2
4. Java	🌐 📱 🖥	97.5
5. C#	🌐 📱 🖥	89.8
6. PHP	🌐	85.4
7. R	🖥	83.3
8. JavaScript	🌐 📱	82.8
9. Go	🌐 🖥	76.7
10. Assembly	▪	74.5

From https://spectrum.ieee.org/static/interactive-the-top-programming-languages-2018

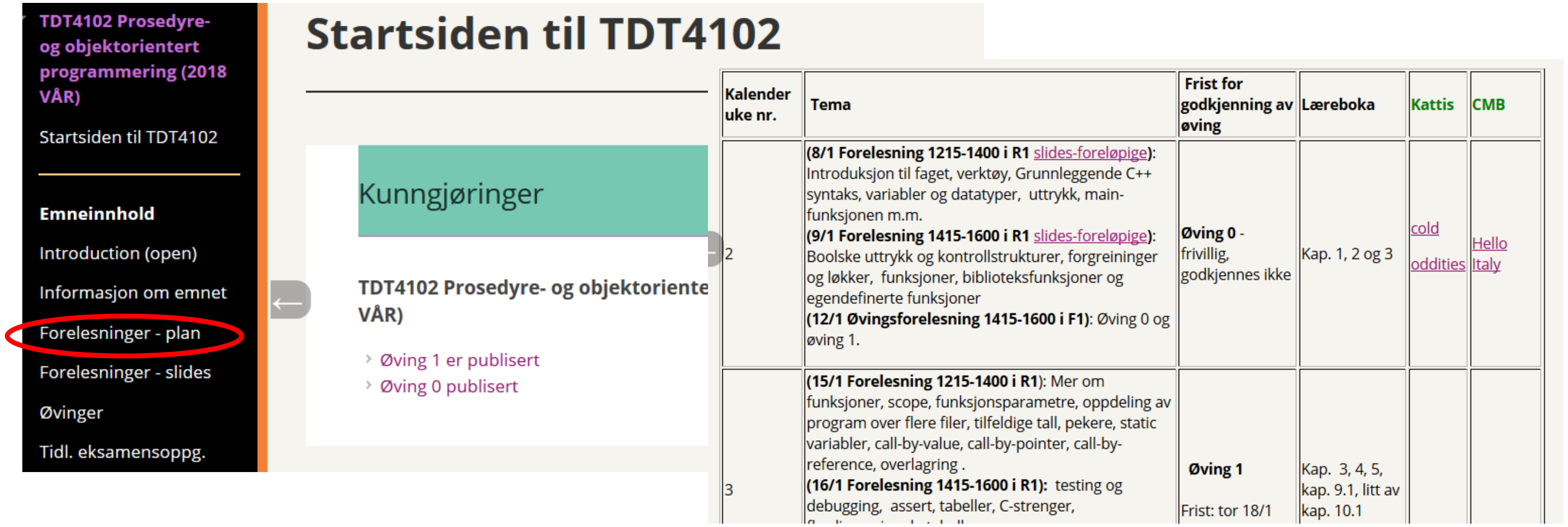
Who Invented Object Oriented Programming?



- Where?
- When?

TDT4102: Om kurset, praktisk info (TODO)

- All info som er knyttet til gjennomføring av TDT4102 våren 2019 ligger på fagets Blackboard-sider



TDT4102 Prosedyre- og objektorientert programmering (2018 VÅR)

Startsiden til TDT4102

Emneinnhold

Introduction (open)

Informasjon om emnet

Forelesninger - plan

Forelesninger - slides

Øvinger

Tidl. eksamensoppg.

Startsiden til TDT4102

Kunngjøringer

TDT4102 Prosedyre- og objektoriente VÅR)

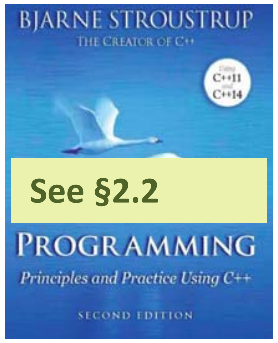
- Øving 1 er publisert
- Øving 0 publisert

Kalender uke nr.	Tema	Frist for godkjenning av øving	Læreboka	Kattis	CMB
2	**(8/1 Forelesning 1215-1400 i R1** slides-foreløpige**)**: Introduksjon til faget, verktøy, Grunnleggende C++ syntaks, variabler og datatyper, uttrykk, main-funksjonen m.m. **(9/1 Forelesning 1415-1600 i R1** slides-foreløpige**)**: Boolske uttrykk og kontrollstrukturer, forgreininger og løkker, funksjoner, biblioteksfunksjoner og egendefinerte funksjoner **(12/1 Øvingsforelesning 1415-1600 i F1)**: Øving 0 og øving 1.	**Øving 0** - frivillig, godkjennes ikke	Kap. 1, 2 og 3	cold oddities	Hello Italy
3	**(15/1 Forelesning 1215-1400 i R1)**: Mer om funksjoner, scope, funksjonsparametre, oppdeling av program over flere filer, tilfeldige tall, pekere, static variabler, call-by-value, call-by-pointer, call-by-reference, overlagring . **(16/1 Forelesning 1415-1600 i R1):** testing og debugging, assert, tabeller, C-strenger,	**Øving 1** Frist: tor 18/1	Kap. 3, 4, 5, kap. 9.1, litt av kap. 10.1		

Basic concepts with examples in C++ (Chap. 2)

- **Overview**
  - Program, include file, compilation, library, linking, object code, execution, debugging
  - Console output (cout), text string, newline, escape character
  - The special function main()

Example program Lec1Ex1.cpp



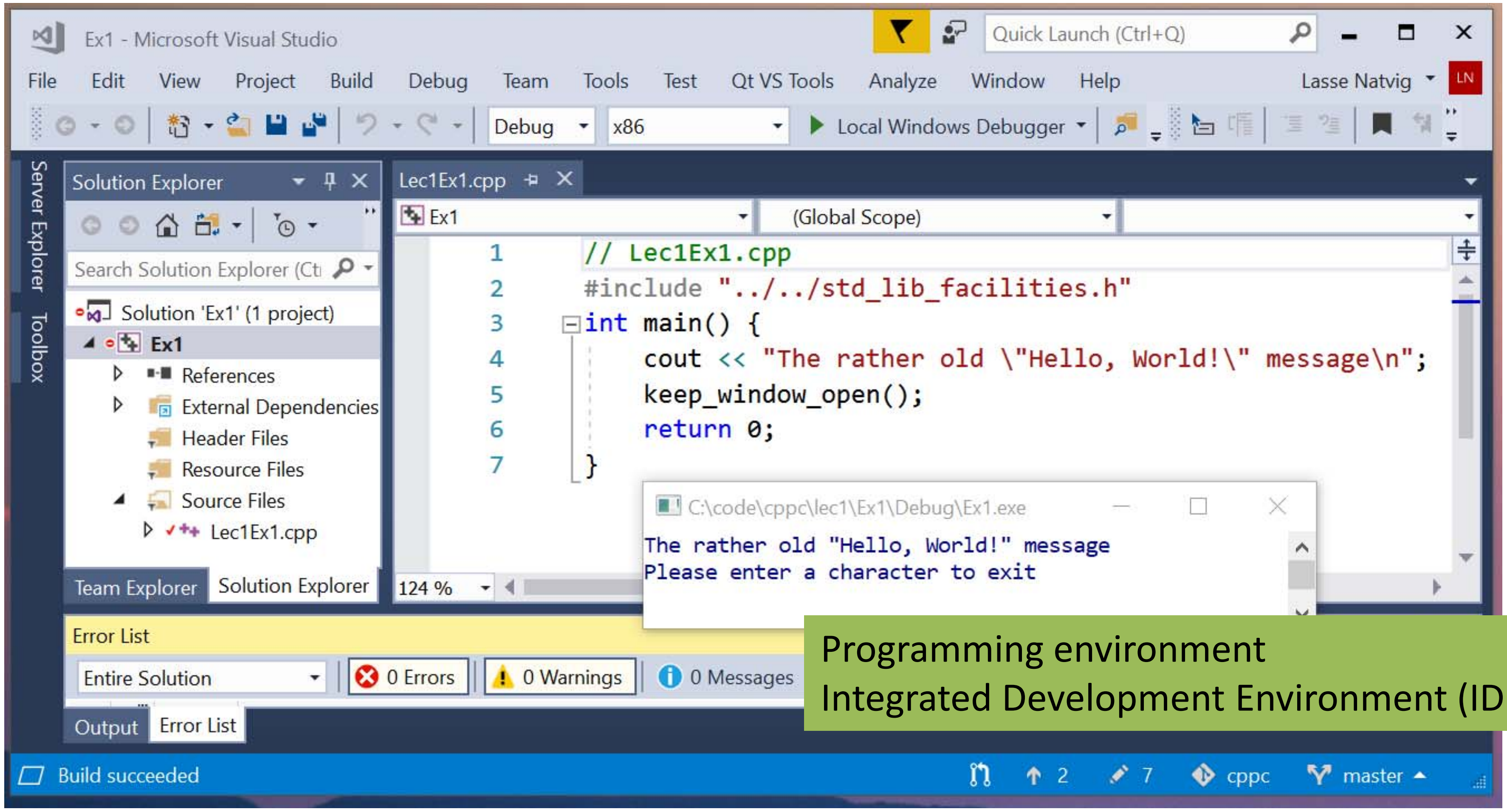
See §2.2

Source code

```cpp
// Lec1Ex1.cpp
#include "../../std_lib_facilities.h"
int main() {
    cout << "The rather old \"Hello, World!\" message\n";
    keep_window_open();
    return 0;
}
```

Output, result

```
The rather old "Hello, World!" message
Please enter a character to exit
```



```
// Lec1Ex1.cpp
#include "../../std_lib_facilities.h"
int main() {
    cout << "The rather old \"Hello, World!\" message\n";
    keep_window_open();
    return 0;
}
```

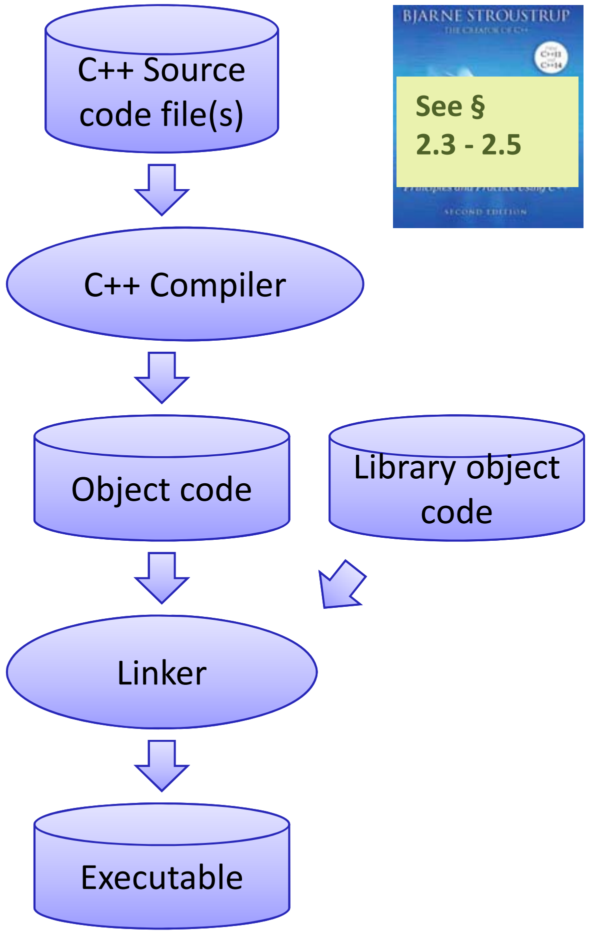
C:\code\cppc\lec1\Ex1\Debug\Ex1.exe

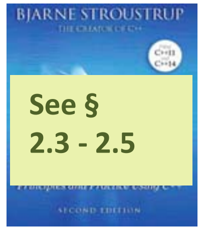
```
The rather old "Hello, World!" message
Please enter a character to exit
```

Programming environment
Integrated Development Environment (IDE)

From source code to executable

- The **compiler** translates into **object code**
  - sometimes called machine code
  - is simple enough for a computer to "understand"
- The **linker** links your code to system code
  - needed to execute
  - E.g., input/output libraries, operating system code, and windowing code
- The result is an **executable** program
  - a .exe file on windows or an a.out file on Unix

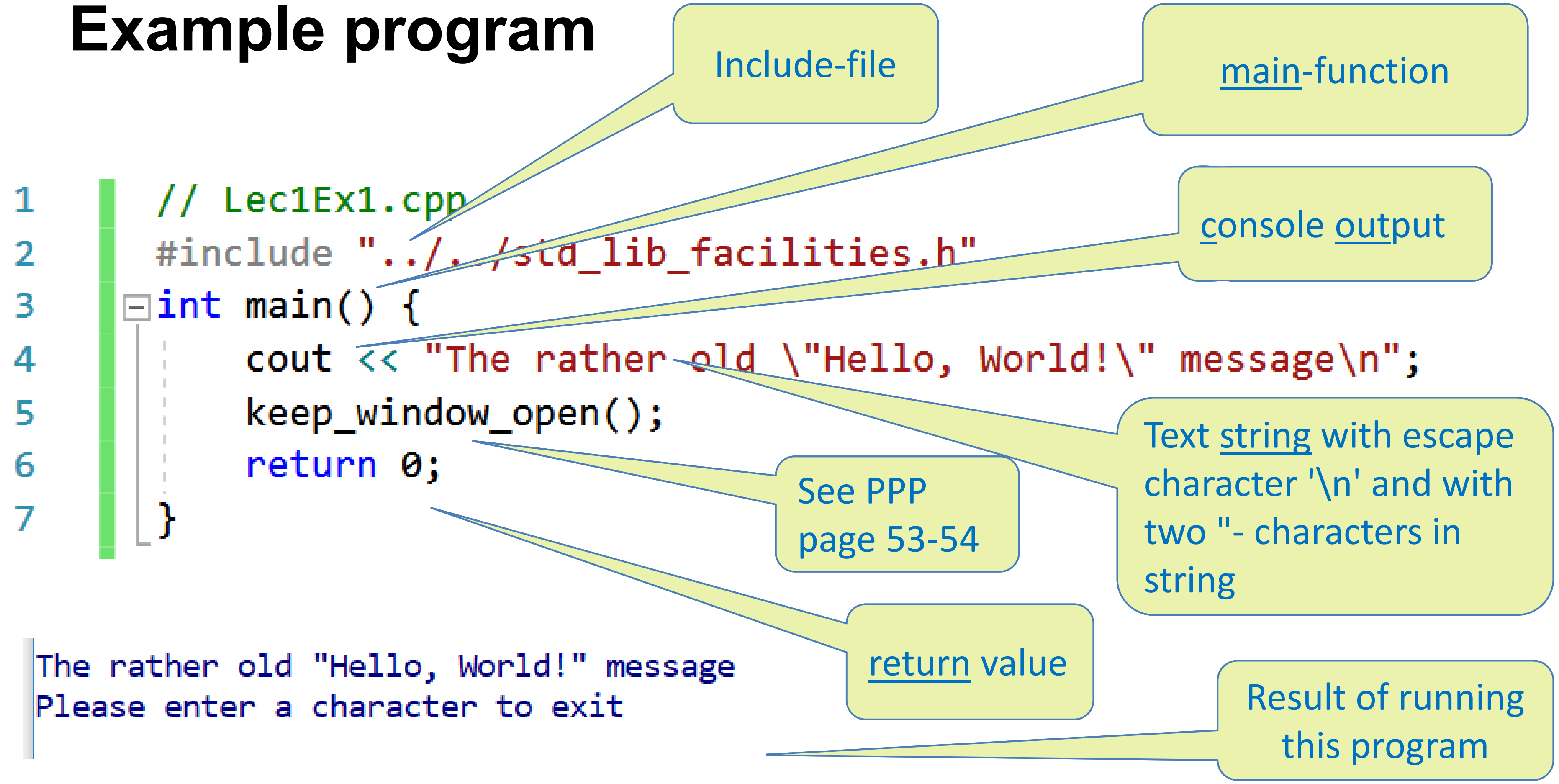




Live coding – Hello World

- New MS-VS project
- Int main() { }
- Return
- Output text
- Normal execution and debugger-execution
- Keep window open
- Compile, error messages
- Debugger - breakpoint

Example program



```
// Lec1Ex1.cpp
#include "../../std_lib_facilities.h"
int main() {
    cout << "The rather old \"Hello, World!\" message\n";
    keep_window_open();
    return 0;
}
```

Include-file
main-function
console output
Text string with escape character '\n' and with two "- characters in string
See PPP page 53-54
return value
Result of running this program

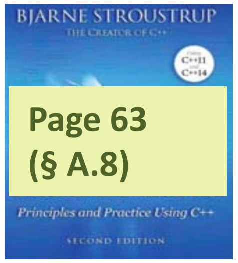
```
The rather old "Hello, World!" message
Please enter a character to exit
```

Basic concepts with examples in C++ (Chap. 3)

- Objects, types and values
- Variables
- Operations and operators
- Assignments and initialization
- (Console) input (cin)

The five fundamental types and operators

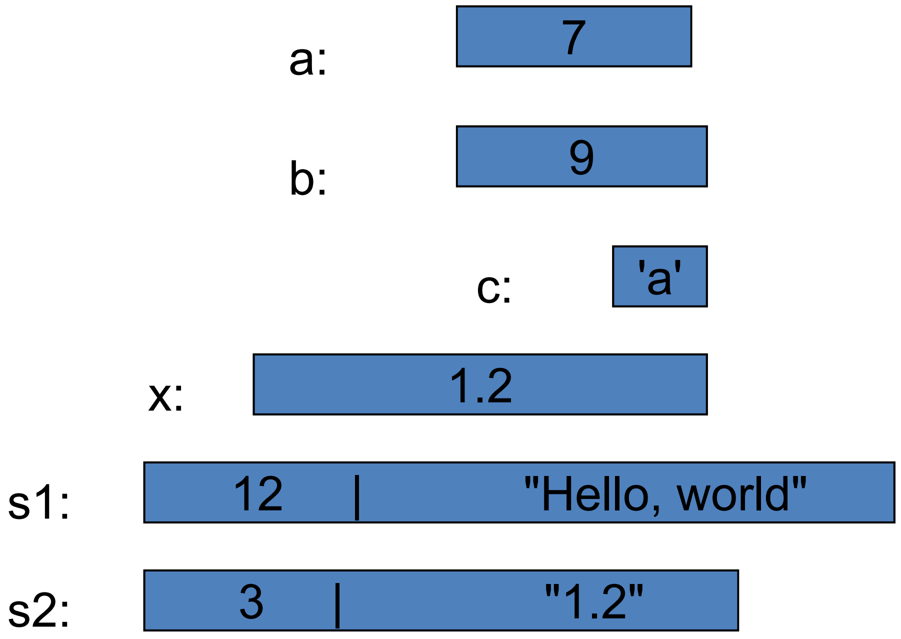
- `int` integer, `double` floating point number
- `char` character, `string` a string of characters
- `bool` boolean, logical variable
- (There are many more, see PPP § A.8, page 1099)
- **Operators**
  - = (assignment), + (addition), - (subtraction), * (multiplication), / (division)
  - % (Remainder, modulo)
  - ++ (increment), -- (decrement)
  - == (equals), != (not equal), ! (negation), > (greater than), >= (greater than or equal), <, <=
  - s >> x (read from s into x), s << x (write x into s)
  - Note; Some operators does not apply to all of these types (see PPP § 3.4 and more)



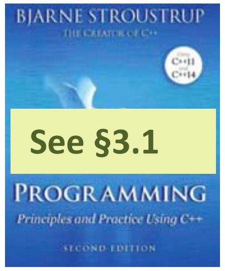
Page 63
(§ A.8)

Variables, declaration and initialization

```
int a = 7;

int b = 9;

char c = 'a';

double x = 1.2;

string s1 = "Hello, world";

string s2 = "1.2";
```



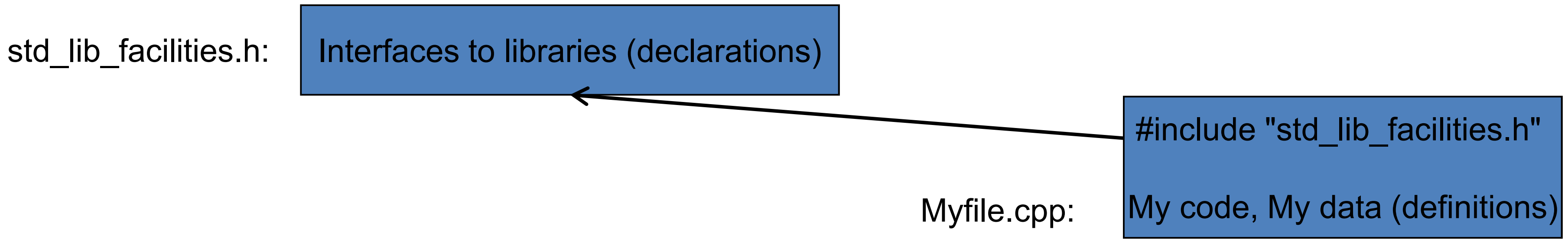
Console input and std_lib_facilities.h



- Console input (cin), example

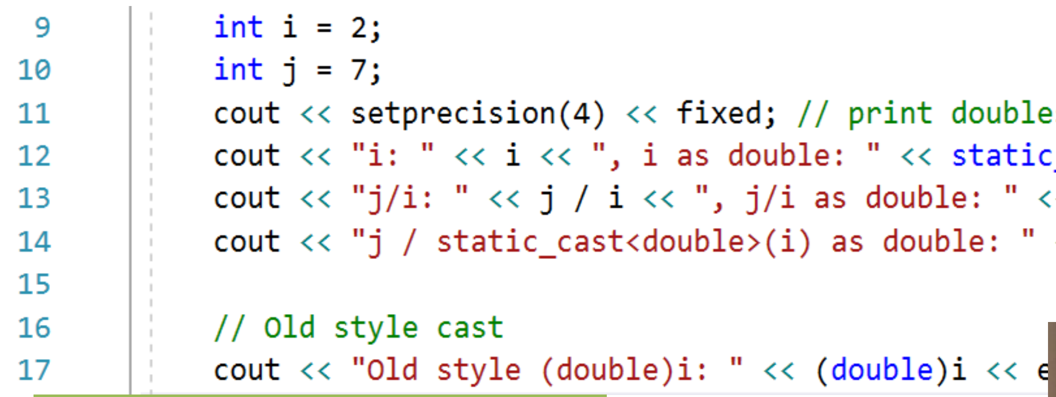
```
#include "std_lib_facilities.h"        // our course header
int main() {
    cout << "Please enter your first name (followed by 'enter'):\n";
    string first_name;
    cin >> first_name;
    cout << "Hello, " << first_name << '\n';
}
```

- std_lib_facilities.h" is the header for our course

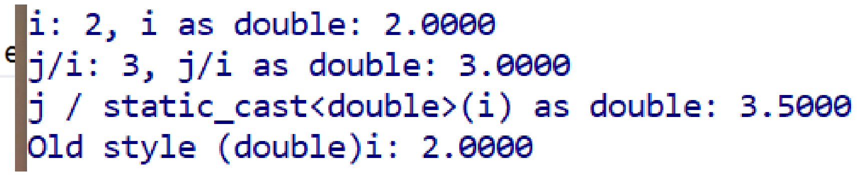

std_lib_facilities.h:
Interfaces to libraries (declarations)
#include "std_lib_facilities.h"
My code, My data (definitions)
Myfile.cpp:

Type conversion, casting Lec1Ex2.cpp

- Sometimes variable values must be converted between different types
  - Try to avoid it
  - The system may do it automatically
  - Do it explicitly in your program to increase readability!



```cpp
int i = 2;
int j = 7;
cout << setprecision(4) << fixed; // print doubles with 4 decimals
cout << "i: " << i << ", i as double: " << static_cast<double>(i) << endl;
cout << "j/i: " << j / i << ", j/i as double: " << static_cast<double>(j / i) << endl;
cout << "j / static_cast<double>(i) as double: " << j / static_cast<double>(i) << endl;

// Old style cast
cout << "Old style (double)i: " << (double)i << e
```



```
i: 2, i as double: 2.0000
j/i: 3, j/i as double: 3.0000
j / static_cast<double>(i) as double: 3.5000
Old style (double)i: 2.0000
```

More on this topic later!

Concise operators

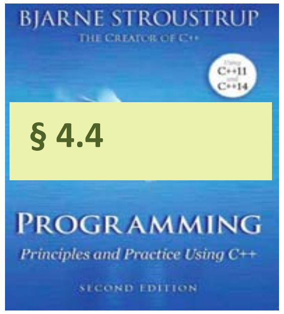
- For many binary operators,
  there are more concise operators

  - For example

a += c	means	**a = a+c**
a *= scale	means	**a = a*scale**
++a	means	**a += 1** or **a = a+1**

Introduction to control structures

- if-else statement
    - block of code
- for-loop
- while-loop
- do – while loop



If statement and compound statement

- if-else statement

```
if (condition)
    statement-1 // if condition is true, do statement-1
else
    statement-2 // if not, do statement-2
```

- **Compound statement** or block
  - A sequence of statements enclosed by curly braces { }
  - Must be used when statement-1 or -2 is more than one statement
- Many programmers advice that statement-1 and -2 in the if-statement always should be a compound statement (a block)

Iteration, while and for

- **While loop**

```
while (condition)
    statement // executes this as long as condition is true
```

- **For loop**

```
for (initialize; condition ; increment )
    statement // executes this as long as condition is true
```

  - initialize is executed before first iteration
  - increment is executed after every iteration
    - Normally (but not always) to increment a loop-counter

- Many programmers advice that statement (often called the loop body) should be a block

do - while loop

Syntax:

```
do
    statement // executes this at least once
while (condition);
```

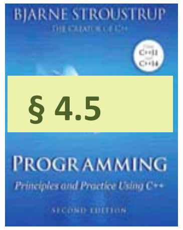
- Not covered in textbook but used in solution for exercise 4

bool

- Datatype for variables that can be true or false
- Can simplify program logic, increase readability

```
...some code
bool found = false;
while (!found) {
   ...some code searching for a given element
     if (element was found)
      found = true;
}
...code processing element
```

Functions – why?



- We define a function when we want to separate a computation because it
  - is logically separate
  - makes the program text clearer (by naming the computation)
  - is useful in more than one place in our program
  - eases testing, distribution of labor, and maintenance

Functions, syntax and example

```
Return_type  function_name ( Parameter list )  {
   // use each parameter in code
   return some_value;    // of Return_type
}
```

Parameters: type name pairs, separated by comma

Example:

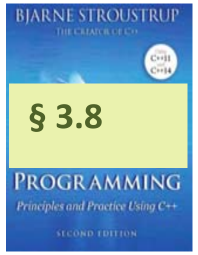
```
int square(int x) {
   return x*x;
}
```

Here: one single parameter passed by **copy-by-value**
Much more on that topic later!

```
// Example use of square
int num = 7;
int result = 0;
result = square(num);
// result is now 49
```

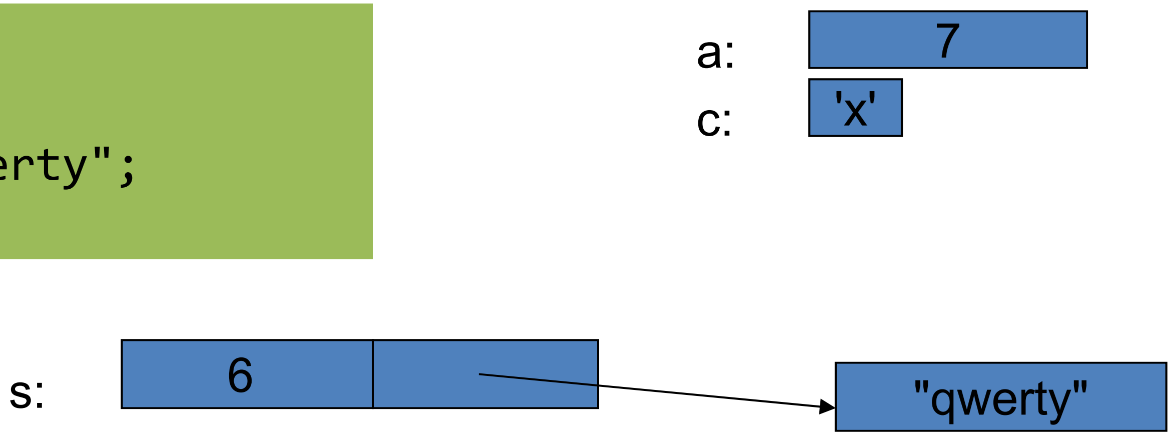
TDT4102: --- Øving 0 og 1 – skal være «dekket» ---

What is an object?



- An object is some memory that can hold a value of a given type
- A variable is a named object
- A declaration names an object

```
int a = 7;
char c = 'x';
string s = "qwerty";
```



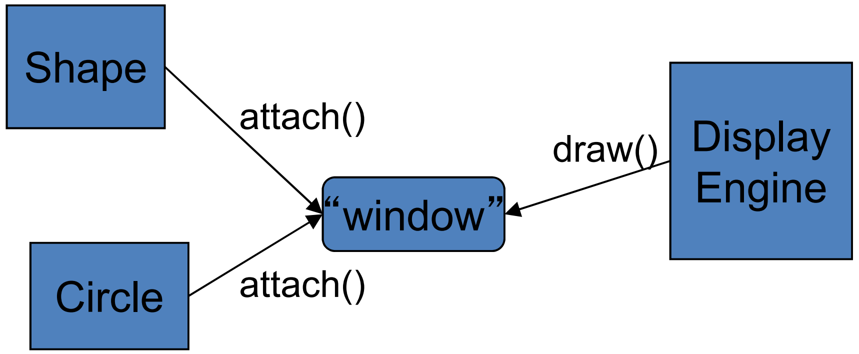
Objects – example, an early view of graphics

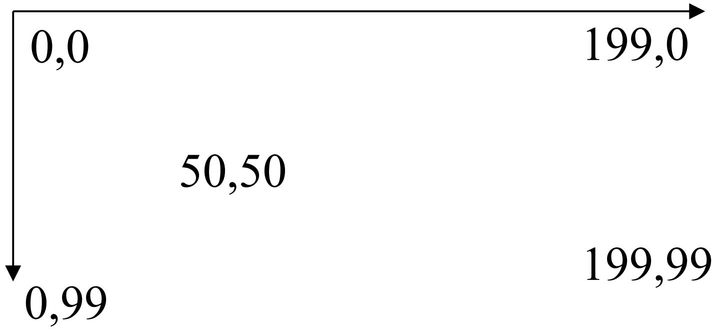
- Object orienteded (OO) programming
  - Organize your data and program into different objects types
  - Define operations for these types
    - Operators (like + for string), more on that later
    - As «member functions» (like length() for string)
  - Use and expand object types defined in the library
    - Examples: string, vector, …
  - … or your own user-defines types
- Graphics helps explaining OO!
  - (Read PPP § 12.1)

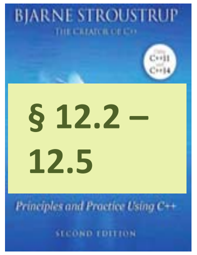
Intro to PPP Graph.h and Simple_window.h

- **Display model**
  - Objects (like Shape and Circle) are «attached to» a window
  - The «display-engine» takes the objects attached to the window and draws them on the screen
  - Display engine is also called GUI-library or system
  - Pixels (picture elements)

- **Coordinates** (x,y)
  - Integers (pixels)
  - (0,0) in top left corner
  - y-coordinate grows downwards!
  - (Fig: width = 200, height = 100)

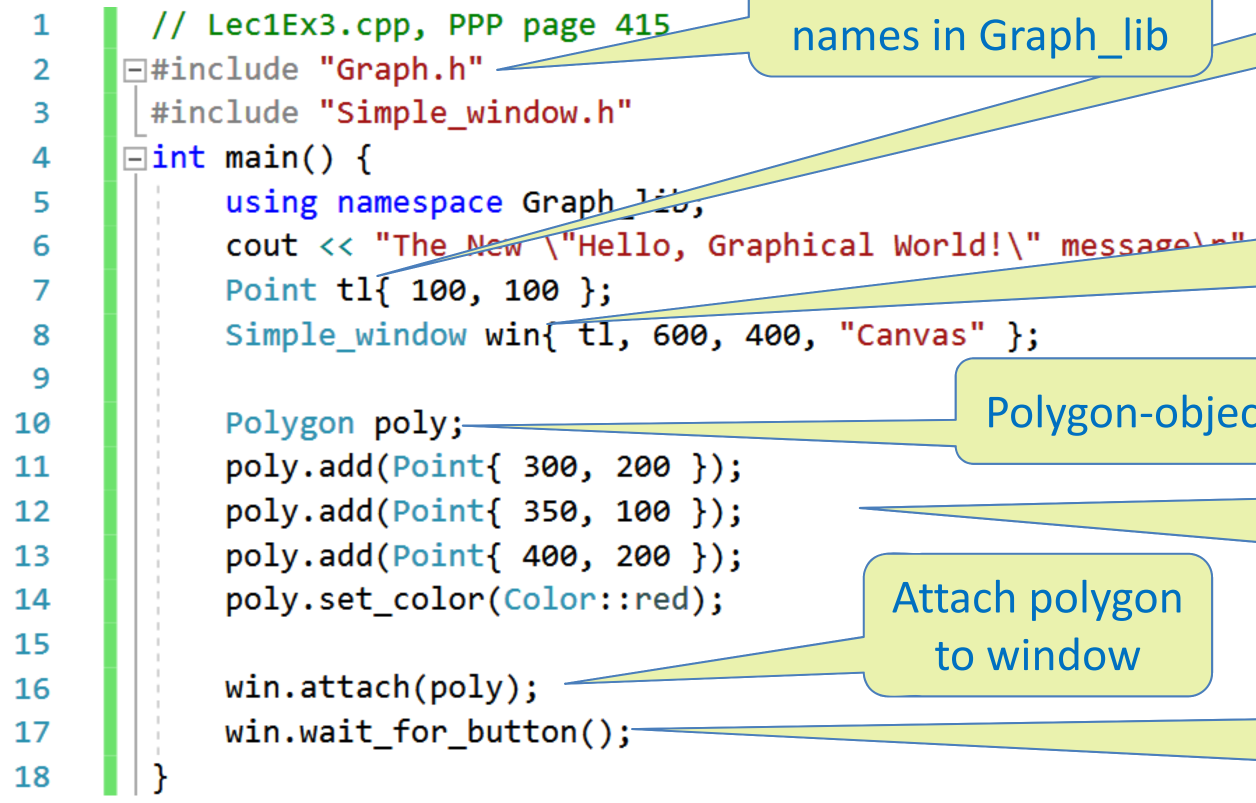






§ 12.2 – 12.5

Example program Lec1Ex3.cpp



```cpp
// Lec1Ex3.cpp, PPP page 415
#include "Graph.h"
#include "Simple_window.h"
int main() {
    using namespace Graph_lib;
    cout << "The New \"Hello, Graphical World!\" message\n";
    Point tl{ 100, 100 };
    Simple_window win{ tl, 600, 400, "Canvas" };

    Polygon poly;
    poly.add(Point{ 300, 200 });
    poly.add(Point{ 350, 100 });
    poly.add(Point{ 400, 200 });
    poly.set_color(Color::red);

    win.attach(poly);
    win.wait_for_button();
}
```

Gives access to names in Graph_lib

Point-object with coor-dinates of top left corner

Simple_window- object of width 600 and height 400, with title Canvas

Polygon-object

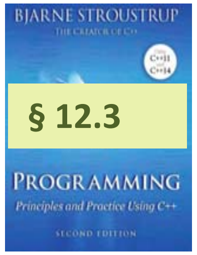
Adding three Point-objects to the polygon and set its color

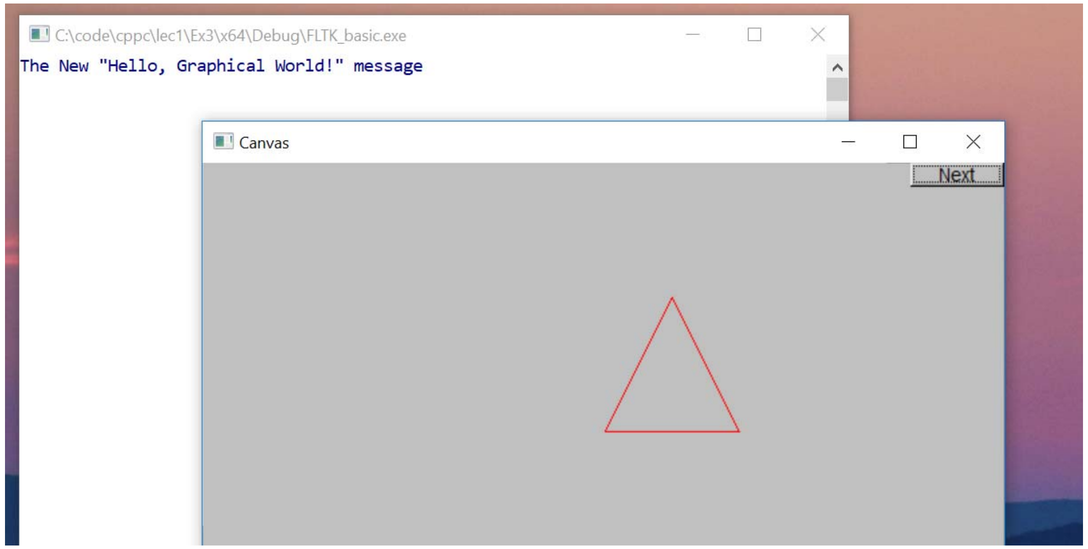
Attach polygon to window

«Trick»: show graphics and wait for user to press the next button

Example program Lec1Ex3.cpp

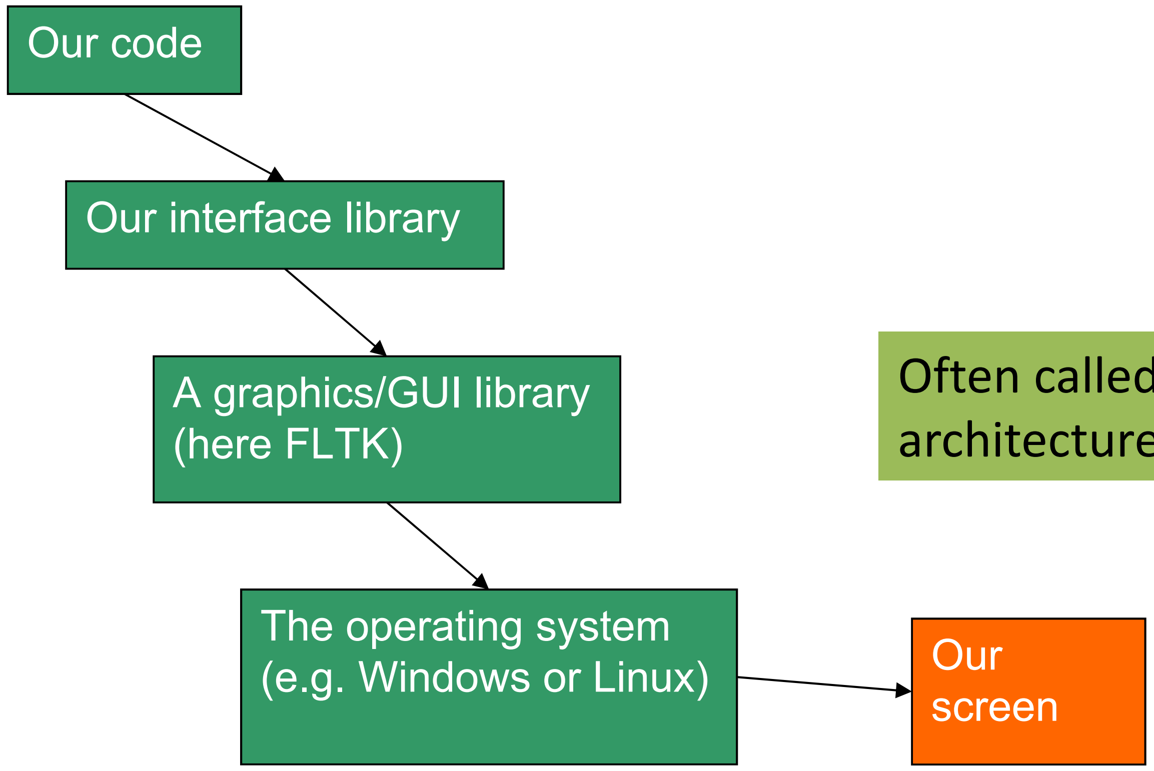
```cpp
// Lec1Ex3.cpp, PPP page 415
#include "Graph.h"
#include "Simple_window.h"
int main() {
    using namespace Graph_lib;
    cout << "The New \"Hello, Graphical World!\" message\n";
    Point tl{ 100, 100 };
    Simple_window win{ tl, 600, 400, "Canvas" };

    Polygon poly;
    poly.add(Point{ 300, 200 });
    poly.add(Point{ 350, 100 });
    poly.add(Point{ 400, 200 });
    poly.set_color(Color::red);

    win.attach(poly);
    win.wait_for_button();
}
```

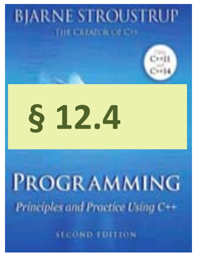

§ 12.3



```
The New "Hello, Graphical World!" message
```

Using a GUI library, «behind the scenes»

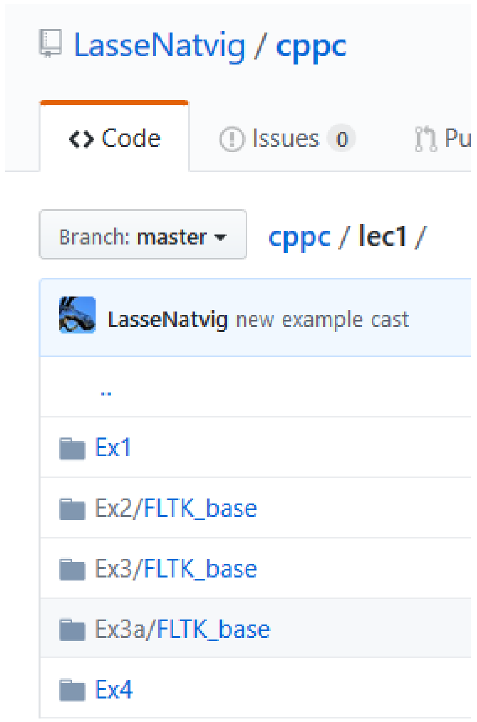

Our code
Our interface library
A graphics/GUI library (here FLTK)
The operating system (e.g. Windows or Linux)
Our screen



Often called “a layered architecture”

Guide to the example programs

- All lecture example programs used in the slides are available from
  - https://github.com/LasseNatvig/cppc
- Directory structure and file names follow structure and names in the lectures
- Use web-browser to read the files
- Download your local copy, compile in your system and step through with debugger to help understanding
- Send suggestions for improvements to tdt4102-fagans@idi.ntnu.no


LasseNatvig / cppc

Branch: master ▾ cppc / lec1 /

LasseNatvig new example cast

..

Ex1

Ex2/FLTK_base

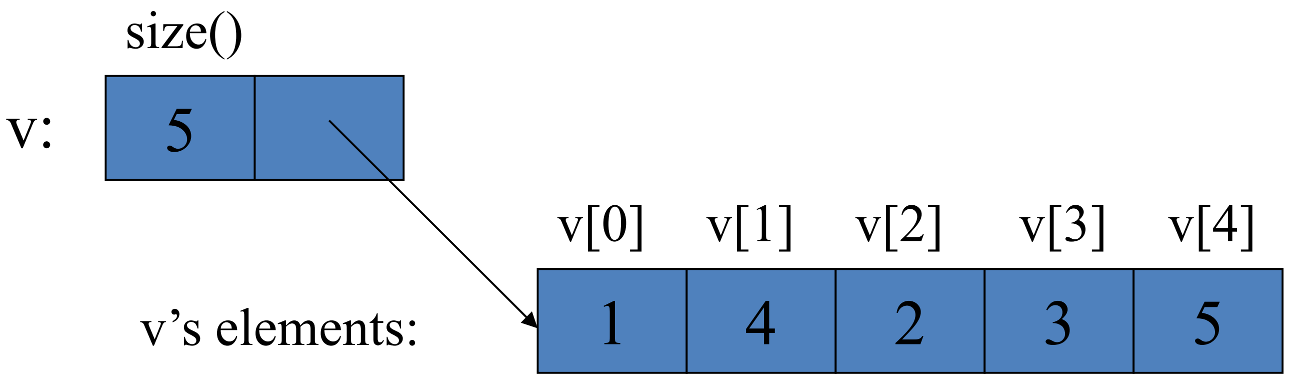
Ex3/FLTK_base

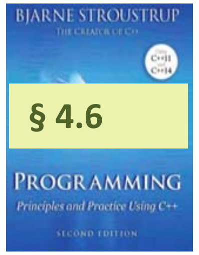
Ex3a/FLTK_base

Ex4

Introduction to <vector>

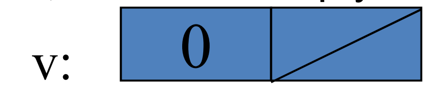
- a collection of data, an example of a C++ container
- the most useful standard library data type
- a vector<T> holds an sequence of values of type T
- `vector<int> v{1, 4, 2, 3, 5};`



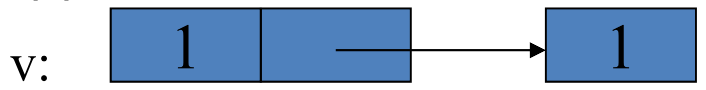


Growing a vector, push_back

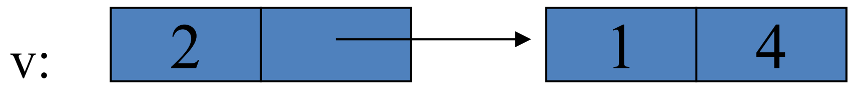
vector<int> v; // start off empty



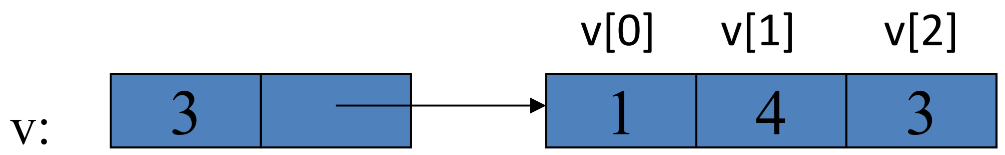
v.push_back(1); // add an element with the value **1**



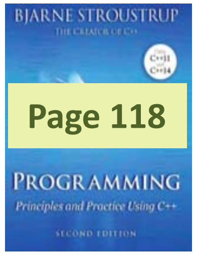
v.push_back(4); // add an element with the value **4** at end ("the back")



v.push_back(3); // add an element with the value **3** at end ("the back")



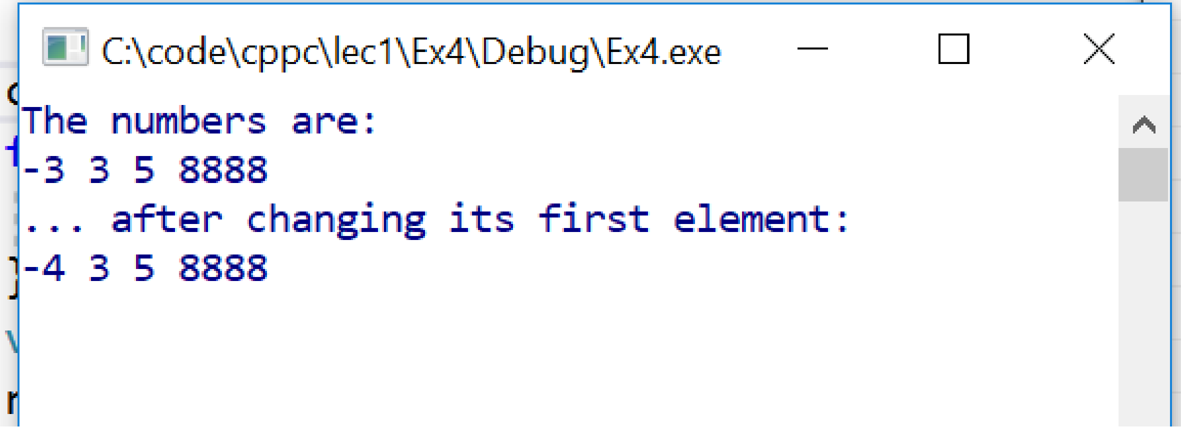
<vector>



- Vector of any type
  - int, double, string, vector … user defined types
  - More examples later (See also PPP §7.8.1)
- vector<myType> vName(5)
  - Declares a variable named vName that is a vector storing 5 elements of type myType, each having their default value
    - 0 for int
    - "" for string.
- Vector example: **Lec1Ex4.cpp**
  - Declaration, initialization, iteration, push_back, size, function returning vector

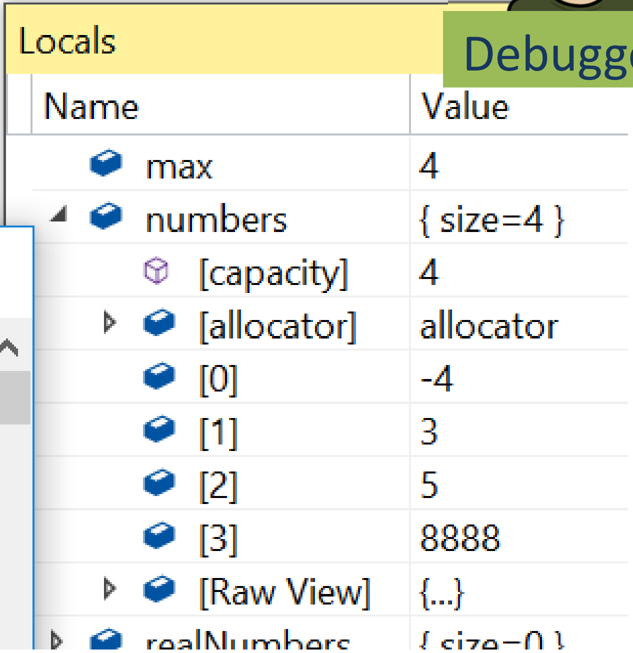
Example program Lec1Ex4.cpp ***with debugger***

```cpp
numbers[0] = -4;
cout << "\n... after changing its first element:\n";
unsigned int max = numbers.size();
for (unsigned int i = 0; i < max; i++) {
    cout << numbers[i] << " ";
}
```



C:\code\cppc\lec1\Ex4\Debug\Ex4.exe

```
The numbers are:
-3 3 5 8888
... after changing its first element:
-4 3 5 8888
```



Locals

Name	Value
max	4
numbers	{ size=4 }
[capacity]	4
[allocator]	allocator
[0]	-4
[1]	3
[2]	5
[3]	8888
[Raw View]	{...}
realNumbers	{ size=0 }

TDT4102: --- Stopp forelesning 1 ---

- Vi vil antakelig ikke komme så langt som dette
- Reserve
  - Spørsmål fra salen?
    - C++?
    - Administrativt?
  - Mer tid på noen av eksempelprogrammene?