Lecture 1: Basics in C++

Jonas Kusch and Martina Prugger

University of Innsbruck

September 28, 2022

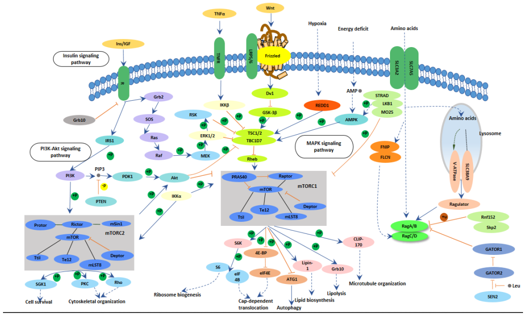
Martina (martina.prugger@uibk.ac.at)

CV:

- studies in Mathematics in Innsbruck
- PhD in Mathematics in Innsbruck
- senior scientist biochemistry Innsbruck

Research:

- systems biology
- chemical master equation
- low-rank methods



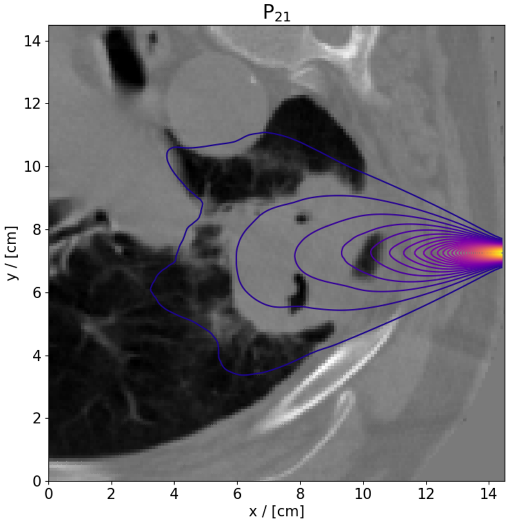
Jonas (jonas.kusch@uibk.ac.at)

CV:

- studies Computational Engineering Science Aachen
- PhD Mathematics in Karlsruhe
- Postdoc in Innsbruck
- associate professor in Oslo/As

Research:

- computational radiation therapy
- data science
- low-rank methods



Learning goals of this course: You will be able to

- ☐ write and compile efficient code in C++
- ☐ understand and use standard datatypes
- ☐ use IO functionalities
- ☐ understand and use pointers (memory)
- ☐ use the debugger
- ☐ use and implement libraries
- ☐ write object-oriented code

Learning goals of this course: You will be able to

- ☐ write and compile efficient code in C++
- ☐ understand and use standard datatypes
- ☐ use IO functionalities
- ☐ understand and use pointers (memory)
- ☐ use the debugger
- ☐ use and implement libraries
- ☐ write object-oriented code

Today's learning goals: You will be able to

- ☐ understand main advantages and disadvantages of C++
- ☐ set up your project with VS code
- ☐ write and compile your own first code with C++
- ☐ use libraries, print outputs to terminal
- ☐ learn about types

Learning goals of this course: You will be able to

- ☐ write and compile efficient code in C++
- ☐ understand and use standard datatypes
- ☐ use IO functionalities
- ☐ understand and use pointers (memory)
- ☐ use the debugger
- ☐ use and implement libraries
- ☐ write object-oriented code

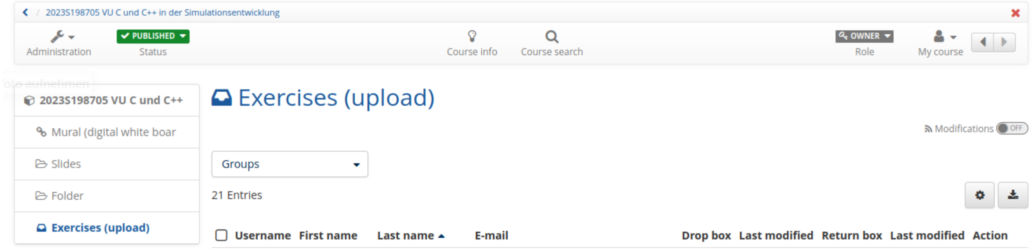
Today's learning goals: You will be able to

- ☐ understand main advantages and disadvantages of C++
- ☐ set up your project with VS code
- ☐ write and compile your own first code with C++
- ☐ use libraries, print outputs to terminal
- ☐ learn about types

Ask questions any time!

What we expect from you

- Programming is not a spectators sport. You learn by writing code.
- *Exercises during course*, hand in exercises via olat by Wednesday in the next week.
- *Programming project*
  - groups of three
  - use UIBK gitlab, everyone should work with git / have commits
  - presentation of code
  - topic of project will be presented later



Exercises (upload)

21 Entries

Username	First name	Last name	E-mail	Drop box	Last modified	Return box	Last modified	Action

Why C++? - Simple stencil code

Python

```
for i in arange(1,N-1):
  for j in arange(1,N-1):
      out[j+i*N] = inn[j+i*N] \
                              + alpha*(inn[j-1+i*N] - 2.0*inn[j+i*N] + inn[j+1+i*N])
```

```
Wall time (2nd order):  9.5 s
```

C++

```
for(int i=1;i<N-1;i++)
    for(int j=1;j<N-1;j++)
        out[j+i*N] = in[j+i*N]
                              + alpha*(in[j-1+i*N] - 2.0*in[j+i*N] + in[j+1+i*N]);
```

```
Wall time (2nd order): 0.0028 s
```

Performance

Drastic performance improvement (≈ 3000x).

- Reduces run time from say 1h to 1s.

Python is an **interpreted language.**

- A program, the interpreter, reads the python code and executes one statement after another.

C++ is a **compiled language** that uses a two-step approach

- **Compilation:** A program, the compiler, reads the C++ code and translates it into machine code.
- **Execution:** The machine code is directly executed by the CPU.

In reality Python uses an intermediate representation (bytecode).

Compilation

Can we compile Python?

```
x = 3 # x is a integer
if condition:
    x = 3.0 # x is a floating point number
y = 2*x
```

What operation should 2*x be compiled to?

- integer multiplication
- floating point multiplication

Compiler would need to understand all possible types that x can have.

- For an interpreted language this is much simpler (can be decided during run time)

In C++ type information need to be specified at compile time.

- Having type information available makes optimizing the machine code easier.

Performance

Fourth order stencil in Python

```
for i in arange(1,N-1):
    for j in arange(1,N-1):
        out[j+i*N] = inn[j+i*N] + alpha*(-1./12.*inn[j-2+i*N] \
          + 4./3.*inn[j-1+i*N] - 5./2.*inn[j+i*N] + 4./3.*inn[j+1+i*N] \
          - 1./12.*inn[j+2+i*N]);
```

```
Wall time (2nd order):  9.5 s
Wall time (4th order):  15.4 s
```

Fourth order stencil in C++

```
for(int i=1;i<N-1;i++)
    for(int j=2;j<N-2;j++)
        out[j+i*N] = inn[j+i*N] + alpha*(-1./12.*in[j-2+i*N]
          + 4./3.*in[j-1+i*N] - 5./2.*in[j+i*N] + 4./3.*in[j+1+i*N]
          - 1./12.*in[j+2+i*N]);
```

```
Wall time (2nd order): 0.0028 s
Wall time (4th order): 0.0028 s
```

Performance and use

In Python using the second order method is 1.6x faster than fourth order method. In C++ the second and fourth order methods take the same time.

- Performance is limited by fetching data from memory and not by the number of arithmetic operations the CPU can perform.

Writing a Python code to predict the performance of a C++ code is a bad idea!

Role of Python: Many software packages have embraced a high-level Python interface.

Python used to specify the problem but all the heavy lifting is done under the hood (usually in C++).

Makes the software approachable for users while maintaining good performance.

Today's learning goals: You will be able to

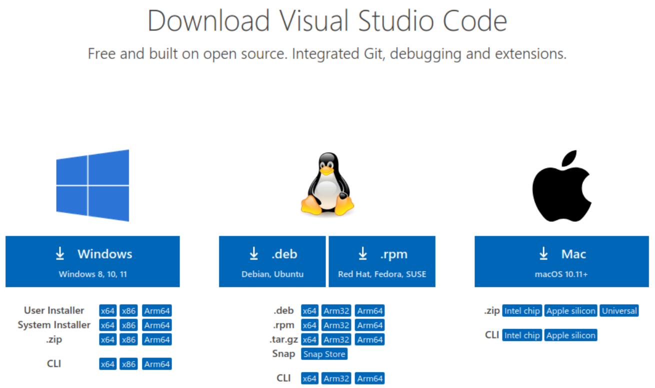
- ☑ understand main advantages and disadvantages of C++
- ☐ set up your project with VS code
- ☐ write and compile your own first code with C++
- ☐ use libraries, print outputs on terminal
- ☐ learn about types

Today's learning goals: You will be able to

- ☑ understand main advantages and disadvantages of C++
- ☐ set up your project with VS code
- ☐ write and compile your own first code with C++
- ☐ use libraries, print outputs on terminal
- ☐ learn about types

Set up VS code

```
https://code.visualstudio.com/download
```

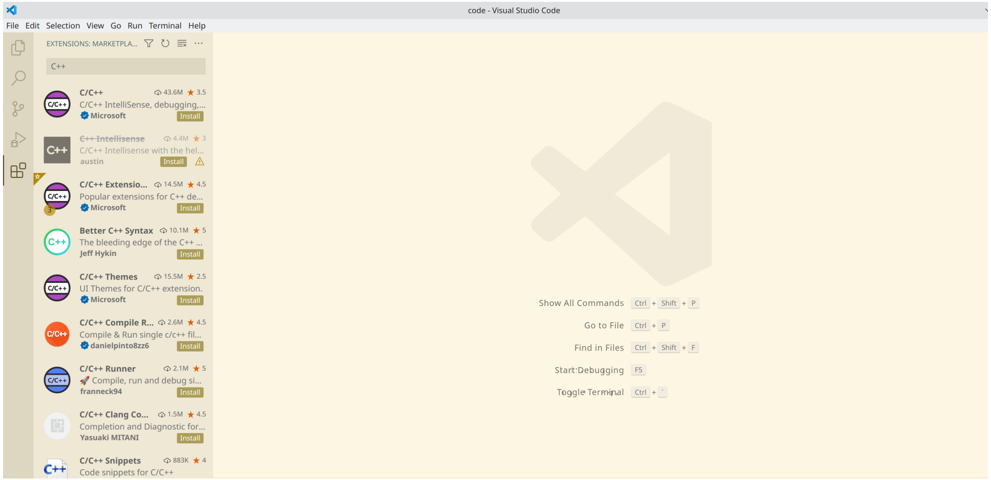


Download Visual Studio Code

Free and built on open source. Integrated Git, debugging and extensions.

```
sudo apt install ./code_1.76.0-1677667493_amd64.deb
```

Set up VS code


code - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXTENSIONS: MARKETPLA...
C++

C/C++ 43.6M ★ 3.5
C/C++ IntelliSense, debugging,...
Microsoft Install

~~C++ Intellisense~~ 4.4M ★ 3
C/C++ Intellisense with the hel...
austin Install

C/C++ Extensio... 14.5M ★ 4.5
Popular extensions for C++ de...
Microsoft Install

Better C++ Syntax 10.1M ★ 5
The bleeding edge of the C++ ...
Jeff Hykin Install

C/C++ Themes 15.5M ★ 2.5
UI Themes for C/C++ extension.
Microsoft Install

C/C++ Compile R... 2.6M ★ 4.5
Compile & Run single c/c++ fil...
danielpinto8zz6 Install

C/C++ Runner 2.1M ★ 5
🚀 Compile, run and debug si...
franneck94 Install

C/C++ Clang Co... 1.5M ★ 4.5
Completion and Diagnostic for...
Yasuaki MITANI Install

C/C++ Snippets 883K ★ 4
Code snippets for C/C++

Show All Commands	Ctrl + Shift + P
Go to File	Ctrl + P
Find in Files	Ctrl + Shift + F
Start Debugging	F5
Toggle Terminal	Ctrl + `

Today's learning goals: You will be able to

- [✓] understand main advantages and disadvantages of C++
- [✓] set up your project with VS code
- [ ] write and compile your own first code with C++
- [ ] use libraries, print outputs on terminal
- [ ] learn about types

Today's learning goals: You will be able to

- ☑ understand main advantages and disadvantages of C++
- ☑ set up your project with VS code
- ☐ write and compile your own first code with C++
- ☐ use libraries, print outputs on terminal
- ☐ learn about types

Let’s start with some code...

Let’s start with some code...

```cpp
#include <iostream> // header file library

int main(){     // main function is called when running code
    std::cout << "Hello World!" << std::endl;   // print "Hello World!"
    return 0;   // return terminates the program
}
```

We recall:

- comments are added by `// comment`
- start by including libraries (iostream for printing output)
- main instructions inside `int main(){ program code }`
- print output by `std::cout << "..."`, end line with `std::endl`
- every command must be ended with `;`
- end program with `return 0;`

Compiling code

```
$ g++ -o hello.out main.cpp
```

The compiler goes through your source code to

- check if the code has the correct syntax
- translate the code into instructions readable by the computer
- generates the executable `hello.out` which you can run with

  ```
  $ ./hello.out
  ```

Compiling code

```
$ g++ -o hello.out main.cpp
```

The compiler goes through your source code to

- check if the code has the correct syntax
- translate the code into instructions readable by the computer
- generates the executable `hello.out` which you can run with

  ```
  $ ./hello.out
  ```

☑ We will revisit the detailed mechanism of the compiler today and later in the semester. For now, let us play around with what we learned.

Now it's up to you...

Task

Write a program which outputs $sin(2)$ in the terminal.
Hint 1: The sin function is included in the `cmath` library.
Hint 2: To print a number with `cout` you do not need the quotation marks.

Now it's up to you...

Task

Write a program which outputs $sin(2)$ in the terminal.
Hint 1: The sin function is included in the `cmath` library.
Hint 2: To print a number with `cout` you do not need the quotation marks.

Recall

```cpp
#include <iostream> // header file library

int main(){     // main function is called when running code
    std::cout << "Hello World!" << std::endl;   // print "Hello World!"
    return 0;   // return terminates the program
}
```

Now it's up to you...

Task

Write a program which outputs $sin(2)$ in the terminal.
Hint 1: The sin function is included in the `cmath` library.
Hint 2: To print a number with `cout` you do not need the quotation marks.

Solution

```cpp
#include <iostream> // library includes cout, endl
#include <cmath> // library includes sin

int main(){
    std::cout << sin(2.0) << endl;
    return 0
}
```

Now it's up to you...

Task

Write a program which outputs *sin*(2) in the terminal.
Hint 1: The sin function is included in the `cmath` library.
Hint 2: To print a number with `cout` you do not need the quotation marks.

Solution

```
#include <iostream> // library includes cout, endl
#include <cmath> // library includes sin

int main(){
    std::cout << sin(2.0) << std::endl;
    return 0;
}
```

A simple IO library

The `fstream` library can be used to read/write data from and to files.

```cpp
#include <iostream>
#include <fstream> // library to read/write data

int main(){
    double tmp1, tmp2;
    std::fstream in("input.txt"); // specify input file

    in >> tmp1;  // read first value
    in >> tmp2; // read second value
    std::cout<<tmp1<<" "<<tmp2<<std::endl;

    return 0;
}
```

A simple IO library

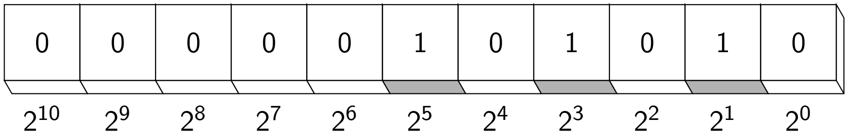
The `fstream` library can be used to read/write data from and to files.

```cpp
#include <iostream>
#include <fstream> // library to read/write data

int main(){
    std::ofstream out("result.txt"); // specify output file

    out<<1.0<<" "<<2.2<<std::endl;  // write values

    return 0;
}
```

Today's learning goals: You will be able to

- ☑ understand main advantages and disadvantages of C++
- ☑ set up your project with VS code
- ☑ write and compile your own first code with C++
- ☑ use libraries, print outputs on terminal
- ☐ learn about types

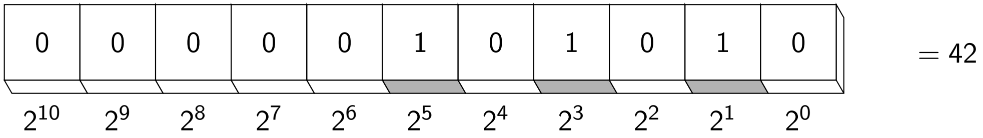
What are types and why do we need them?



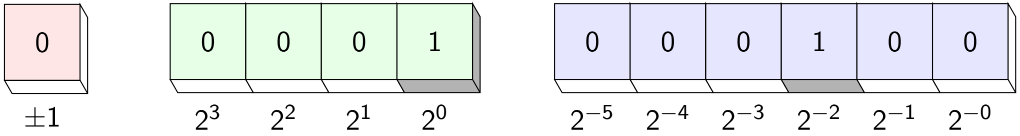
$= 42$

- Numbers are represented by bits.
- Integers usually use 16 bits (-32768 to 32767)

What are types and why do we need them?



- Numbers are represented by bits.
- Integers usually use 16 bits (-32768 to 32767)



- Real numbers: sign $\cdot$ mantissa $\cdot 2^{\text{exponent}}$
- `double` uses 64 bits

What are types and why do we need them?

- Whenever possible, we would like to tell the program how many bits we need.
- Remember that the C++ compiler requires us to define the type in advance

```
x = 3 # x is a integer
if condition:
    x = 3.0 # x is a floating point number
y = 2*x
```

- Types simplify readability.
- Main types are: `char`, `string`, `int`, `long`, `double`

Code example

```c
int main(){
    int i; // declaration
    i = 2; // definition
    double d = 2.3; // declaration and definition
    double a = 1.0, b = 2.3;
    return 0;
}
```

- Tell compiler that only 16 bits will be needed for i and 64 bits are needed for a, b, d.
- Good practice to declare variables at beginning.

Code example

```cpp
int main(){
    int i; // declaration
    i = 2; // definition
    double d = 2.3; // declaration and definition
    double a = 1.0, b = 2.3;
    return 0;
}
```

- Tell compiler that only 16 bits will be needed for i and 64 bits are needed for a, b, d.
- Good practice to declare variables at beginning.

Exercise

Besides the `cout` and `endl` commands that you have already seen, the `iostream` library also offers the `cin` command to read inputs from the command line via `cin >> INPUT;`. Write a program which asks the user to provide a number of type `double` and returns the `sin` of this value.

Now it's up to you...

```cpp
#include <iostream>
#include <cmath>

int main(){
    double input;
    std::cout<<"Enter a number: ";
    std::cin >> input;
    std::cout<<std::endl<<"sin("<<input<<") = "<<sin(input)<<std::endl;
    return 0;
}
```

Now it's up to you...

```cpp
#include <iostream>
#include <cmath>

int main(){
    double input;
    std::cout<<"Enter a number: ";
    std::cin >> input;
    std::cout<<std::endl<<"sin("<<input<<") = "<<sin(input)<<std::endl;
    return 0;
}
```

Exercise

Read in an integer and a double and print out the sum of these two values.

Now it’s up to you...

```
#include <iostream>

int main(){
    double inputInt;
    double inputDouble;
    std::cout<<"Enter two numbers: Integer ";
    std::cin >> inputInt;
    std::cout<<", Double ";
    std::cin >> inputDouble;
    int sum = inputInt + inputDouble;
    std::cout<<"sum is "<<sum<<std::endl;
    return 0;
}
```

Now it's up to you...

```cpp
#include <iostream>

int main(){
    double inputInt;
    double inputDouble;
    std::cout<<"Enter two numbers: Integer ";
    std::cin >> inputInt;
    std::cout<<", Double ";
    std::cin >> inputDouble;
    int sum = inputInt + inputDouble;
    std::cout<<"sum is "<<sum<<std::endl;
    return 0;
}
```

- Accuracy is lost since sum is an integer.
- use typecast `int sum = inputInt + int(inputDouble)`

Now it’s up to you...

```cpp
#include <iostream>

int main(){
    double inputInt;
    double inputDouble;
    std::cout<<"Enter two numbers: Integer ";
    std::cin >> inputInt;
    std::cout<<", Double ";
    std::cin >> inputDouble;
    double sum = inputInt + inputDouble;
    std::cout<<"sum is "<<sum<<std::endl;
    return 0;
}
```

- Accuracy is lost since `sum` is an integer.
- use typecast `int sum = inputInt + int(inputDouble)`

Vector

- There are more datatypes, which are implemented in standard C++ libraries. Let's start with `std::vector`

```cpp
#include <iostream>
#include <vector>

int main(){
    std::vector<double> v{1, 2, 3};
    v[2] = 1.0;
    std::cout<<v[0]<<" "<<v[1]<<" "<<v[2]<<std::endl;
    return 0;
}
```

- Note that you need to put an `std::` in front of the vector (and `cout`, `endl`). This is a namespace which we will cover later.
- indexing starts at 0 in C++!
- `size`, `reserve`, `resize`

Today's learning goals: You will be able to

- ☑ understand main advantages and disadvantages of C++
- ☑ set up your project with VS code
- ☑ write and compile your own first code with C++
- ☑ use libraries, print outputs on terminal
- ☑ learn about types

The compiler

- We write code that humans understand but computers can't.
  $\Rightarrow$ translate into machine language
- The main tools are the *compiler* and the *linker*.
- *compiler*: C++ code into machine language file (object file, `main.o`, `main.obj`)
- *linker*: combine obj files and libraries (precompiled code)

```
$ g++ -c hello.cpp
$ g++ -o hello.out hello.o
$ ls
hello.cpp  hello.o  hello.out

```

We will revisit the compiler multiple times once our programs become more complicated.

Now it's up to you...

Current learning goals: After homework and self-study

- ☑ understand main advantages and disadvantages of C++
- ☑ set up your project with VS code
- ☑ write and compile your own first code with C++
- ☑ use libraries, print outputs to terminal (`iostream`, `cmath`, `fstream`, `cout`)
- ☑ learn about types (`int`, `double`, `vector`,...)

Now it's up to you...

Current learning goals: After homework and self-study

- ☑ understand main advantages and disadvantages of C++
- ☑ set up your project with VS code
- ☑ write and compile your own first code with C++
- ☑ use libraries, print outputs to terminal (`iostream`, `cmath`, `fstream`, `cout`)
- ☑ learn about types (`int`, `double`, `vector`,...)

Now it's up to you...

Current learning goals: After homework and self-study

- ☑ understand main advantages and disadvantages of C++
- ☑ set up your project with VS code
- ☑ write and compile your own first code with C++
- ☑ use libraries, print outputs to terminal (`iostream`, `cmath`, `fstream`, `cout`)
- ☑ learn about types (`int`, `double`, `vector`,...)

Now it's up to you...

Current learning goals: After homework and self-study

- ☑ understand main advantages and disadvantages of C++
- ☑ set up your project with VS code
- ☑ write and compile your own first code with C++
- ☑ use libraries, print outputs to terminal (`iostream`, `cmath`, `fstream`, `cout`)
- ☑ learn about types (`int`, `double`, `vector`,...)

Any questions / remarks ? :)

Now it's up to you...

Current learning goals: After homework and self-study

- ☑ understand main advantages and disadvantages of C++
- ☑ set up your project with VS code
- ☑ write and compile your own first code with C++
- ☑ use libraries, print outputs to terminal (`iostream`, `cmath`, `fstream`, `cout`)
- ☑ learn about types (`int`, `double`, `vector`,...)

Any questions / remarks ? :) {*jonas.kusch, martina.prugger*}*@uibk.ac.at*

Now it's up to you...

Current learning goals: After homework and self-study

- ☑ understand main advantages and disadvantages of C++
- ☑ set up your project with VS code
- ☑ write and compile your own first code with C++
- ☑ use libraries, print outputs to terminal (`iostream`, `cmath`, `fstream`, `cout`)
- ☑ learn about types (`int`, `double`, `vector`,...)

Any questions / remarks ? :) {*jonas.kusch, martina.prugger*}*@uibk.ac.at*

Next learning goals:

- ☐ control flow (`if`, `else`, ...)
- ☐ loops (`for`, `while`, ...)
- ☐ pointers