Digitales Sommersemester 2021:

Datenvisualisierung und GPU-Computing

Einführung in die Programmierübungen: C++

Andreas Beckert

andreas.beckert@uni-hamburg.de

Regionales Rechenzentrum, Visual Data Analysis Group,

FB Informatik Arbeitsgebiet „Scientific Visualization and Parallel Processing“

Termine

- 4-stündige Übung alle 2 Wochen Mittwochs um 14:00 Uhr, mit Ausnahme der ersten 2 Wochen
- Termine:
  - Ü1: 14.04.2021, 14:00 – 16:00
  - Ü2: 21.04.2021, 14:00 – 16:00
  - Ü3: 05.05.2021, 14:00 – 18:00
  - Ü4: 19.05.2021, 14:00 – 18:00
  - Ü5: 02.06.2021, 14:00 – 18:00
  - Ü6: 16.06.2021, 14:00 – 18:00
  - Ü7: 30.06.2021, 14:00 – 18:00

Organisational stuff 1: programming exercises

- The practical programming exercises are designed so that they can be implemented on every **reasonably modern laptop or desktop**, with all major operating systems.
- The exercises are focused on the implementation of a simple visualization program using **C++**. If you are unfamiliar with C++ so far: It is very similar to Java, however, expect to spend some time to learn the basics. It is very well investigated time, though…
- You are expected to work through the assignments on your own and implement the tasks.

Organisational stuff: requirements to complete the course

To successfully complete the course, two parts are required:

1. You will need to **complete the programming exercises**. You will need to provide proof of your code at the end of the term, and you will need to prepare a short written summary of your implementation (about 5 pages), along with some screenshots, by the end of the term.
   Submission deadline: Wednesday 14 July 2021
2. You will need to pass the **written exam** (at the current time it is not clear yet how that will take place). The exam will encompass mainly theoretical questions from the lecture and assigned working material, but also questions that target the exercise material (e.g. writing of pseudo-code of an algorithm). I will provide examples for your preparation during the course of the term.

Organisational stuff: some words on working together

- If you want to **work together** to complete the programming exercises: please **go ahead and do so**! In particular in this time of isolation, we very much encourage you to stay in contact with your fellow students and discuss options for implementations, arising questions, and difficulties as much as possible.
- Some exercise will be done in Zoom Break-Out Groups
- However: we still want each of you to **implement your own code** to fully understand the material.
- At the end of the term, we expect the **written summaries to be your own**, individual work. Also, exam questions will assume that you fully understand the code you have written.
- Having said this, we also want you to **have fun** with the course – visualization is a great sub-discipline of computer science!!

Ziel und Ablauf der Übung

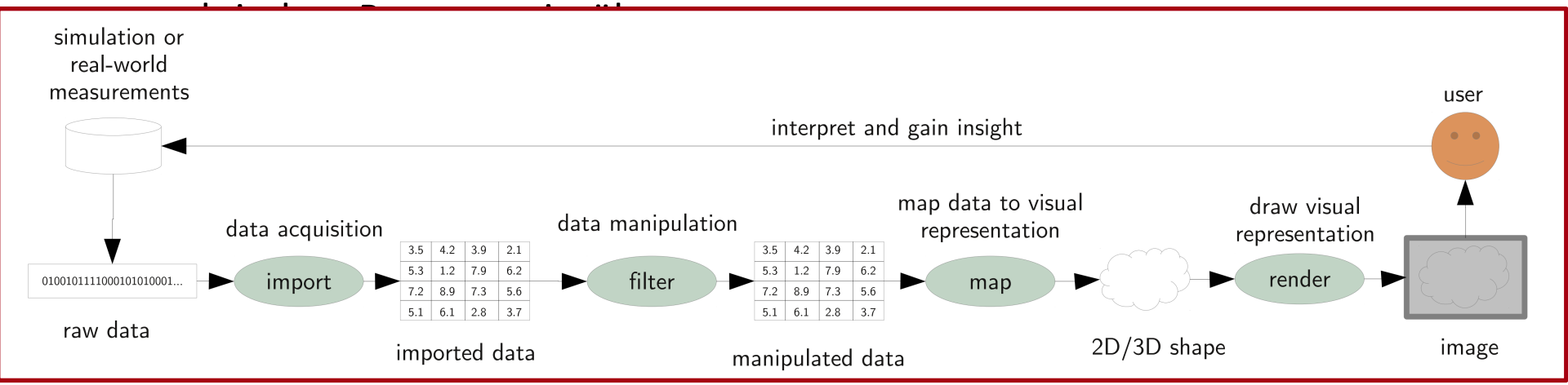
Ziel: Umsetzung von in der Vorlesung gelernten theoretischen Inhalten in praktischen Programmierübungen.

Ablauf: Konzeption und Implementierung eines simplen Visualisierungssystems in C++ mit OpenGL.

- Programmieren mit C++ und Qt
- Programmieren mit OpenGL
- Konzeption und Implementierung einzelner Stages der Visualisierungs-Pipeline: Datenquelle, Mapper, Rendering

Ziel und Ablauf der Übung

Ziel: Umsetzung von in der Vorlesung gelernten theoretischen Inhalten in



- Programmieren mit OpenGL
- Konzeption und Implementierung einzelner Stages der Visualisierungs-Pipeline: Datenquelle, Mapper, Rendering

Intro C++

Bjarne Stroustroup (1997, The C++ Programming Language):

*„The most important thing to do when learning C++ is to **focus on concepts** and not get lost in language-technical details. The purpose of learning a programming language is to become a better programmer; that is, to become more effective at designing and implementing new systems and at maintaining old ones. For this, an **appreciation of programming and design techniques is far more important than an understanding of details**; that understanding comes with time and practice.“*

C++

Was ist C:

- Prozedurale Programmiersprache
- Ursprünglich für Unix Systemprogrammierung entwickelt

Was ist C++:

- Objektorientierte Programmiersprache
- Entwicklung aus der Programmiersprache C
- Ursprung: Bjarne Stroustrup, 1979

Online-Literatur:

http://www.cplusplus.com/

Eigenschaften von C++

- .. **compiled** language
- .. **strongly-typed** unsafe language
- .. offers many paradigm choices (procedural, object-oriented, …)
- .. is portable (w.r.t. language and STL)
- .. is upwards compatible with C
- .. has large library support

- Mehr Eigenschaften auf http://www.cplusplus.com/info/description/

C++ Beispielprogramm: „Hello World“

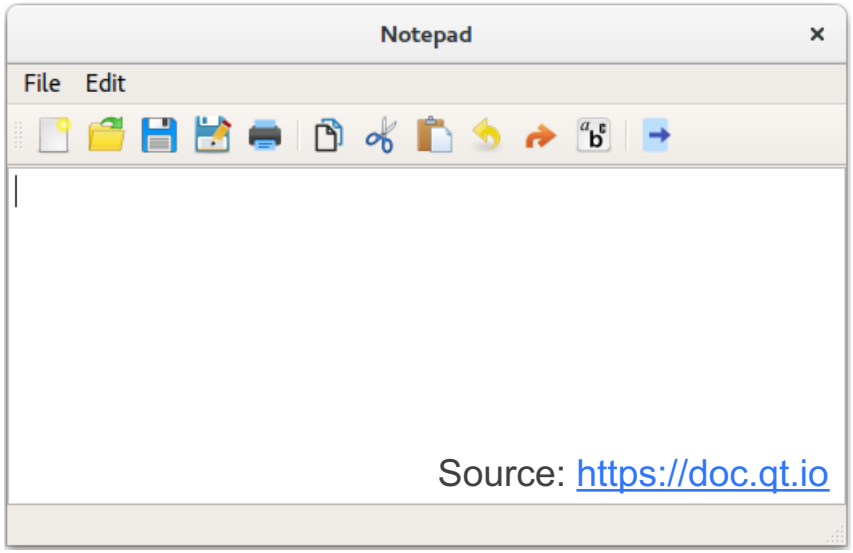
- http://cpp.sh/2dd

```
1 // my first program in C++
2 #include <iostream>
3
4 int main()
5 {
6   std::cout << "Hello World!";
7 }
```

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

Was ist Qt („cute“)?

- Anwendungsframework und GUI-Toolkit zur plattformübergreifenden Entwicklung von Programmen
- In C++ entwickelt, mittlerweile in vielen Sprachen nutzbar
- Open-source und kommerziell
- Erweitert C++ um weitreichende Bibliotheken und Funktionalitäten
- Sehr gut dokumentiert



- Für unsere Zwecke: Vereinfacht OpenGL-Programmierung stark
- Siehe: https://de.wikipedia.org/wiki/Qt_(Bibliothek)
- Siehe: https://doc.qt.io/

Basics von C++

Einführung in C++ aufbauend auf dem Online-Tutorial auf:
http://www.cplusplus.com/doc/tutorial/

- Online-Tutorial enthält detailliertere Informationen.
- Beispielprogramme sind auf http://cpp.sh/ verlinkt – experimentieren online und ohne Entwicklungsumgebung möglich.

Themen, die für uns relevant sind:

- Programmstruktur, Variablen, Operatoren
- Programmfluss, Funktionen
- Arrays, Zeiger, dynamischer Speicher, Datenstrukturen
- Klassen, Methoden, Vererbung
- Standard library vs. Qt library

Structure of a program

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

```cpp
// my first program in C++
#include <iostream>

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

Fügt den Inhalt der angegebenen Datei in dieses Dokument ein.

```cpp
int main () { std::cout << "Hello World!"; }
```

```cpp
// my second program in C++
#include <iostream>
using namespace std;

int main ()
{
  cout << "Hello World! ";
  cout << "I'm a C++ program";
}
```

```cpp
// line comment
/* block comment */
```

Variables and types

- http://www.cplusplus.com/doc/tutorial/variables/

```
6  int main ()
7  {
8    // declaring variables:
9    int a, b;
10   int result;
11
12   // process:
13   a = 5;
14   b = 2;
15   a = a + 1;
16   result = a - b;
17
18   // print out the result:
19   cout << result;
20
21   // terminate the program:
22   return 0;
23 }
```

int a;
int b = 0;

Grunddatentypen: int, char, bool, float, double
Spezifizierung: short, long, signed, unsigned
Zuweisungs-Operatoren: =
Arithmetische-Operatoren: +, -, /, *
Vergleichs-Operatoren: ==, <, >, <=, >=, !=

a = 12;
a = 5 * b;

a = a + 1;
a++;
a += 1;

Constants

- http://www.cplusplus.com/doc/tutorial/constants/
- Literals, typed constants, preprocessor definitions

```
1 3.14159     // 3.14159
2 6.02e23     // 6.02 x 10^23
3 1.6e-19     // 1.6 x 10^-19
4 3.0         // 3.0
```

```
1 'z'
2 'p'
3 "Hello world"
4 "How do you do?"
```

```
 4 const double pi = 3.14159;
 5 const char newline = '\n';
 6
 7 int main ()
 8 {
 9   double r=5.0;
10   double circle;
11
12   circle = 2 * pi * r;
13   cout << circle;
14   cout << newline;
```

```
 4 #define PI 3.14159
 5 #define NEWLINE '\n'
 6
 7 int main ()
 8 {
 9   double r=5.0;
10   double circle;
11
12   circle = 2 * PI * r;
13   cout << circle;
14   cout << NEWLINE;
```

Operators

- http://www.cplusplus.com/doc/tutorial/operators/

expression	equivalent to...
`y += x;`	`y = y + x;`
`x -= 5;`	`x = x - 5;`
`x /= y;`	`x = x / y;`
`price *= units + 1;`	`price = price * (units+1);`

Example 1	Example 2

`x = 3;`
`y = ++x;`
`// x contains 4, y contains 4`
	`x = 3;`
	`y = x++;`
	`// x contains 4, y contains 3`

```
1 ( (5 == 5) && (3 > 6) )  // evaluates to false ( true && false )
2 ( (5 == 5) || (3 > 6) )  // evaluates to true ( true || false )
```

Basic input/output

- http://www.cplusplus.com/doc/tutorial/basic_io/

stream	description
cin	standard input stream
cout	standard output stream
cerr	standard error (output) stream
clog	standard logging (output) stream

```
1 cout << "Output sentence"; // prints Output sentence on screen
2 cout << 120;               // prints number 120 on screen
3 cout << x;                 // prints the value of x on screen
```

```
cout << "I am " << age << " years old and my zipcode is " << zipcode;
```

```
1 cout << "First sentence.\n";
2 cout << "Second sentence.\nThird sentence.";
```

Statements and flow control

- http://www.cplusplus.com/doc/tutorial/control/

```
{ statement1; statement2; statement3; }
```

```
1 if (x == 100)
2 {
3    cout << "x is ";
4    cout << x;
5 }
```

```
1  // countdown using a for loop
2  #include <iostream>
3  using namespace std;
4
5  int main ()
6  {
7    for (int n=10; n>0; n--) {
8      cout << n << ", ";
9    }
10   cout << "liftoff!\n";
11 }
```

for (*initializer***;** *condition; update***)**
{
statement
}

Vorzeitiges verlassen einer Schleife:

break;

```
if (condition)
{
  statement
}
```

```
if (condition)
{
  statement
}
else
{
  statement
}
```

```
switch (selector)
{
      case label:
            statement
      default:
            statement
}
```

switch example	if-else equivalent

`switch (x) {`
`  case 1:`
`    cout << "x is 1";`
`    break;`
`  case 2:`
`    cout << "x is 2";`
`    break;`
`  default:`
`    cout << "value of x unknown";`
`  }`
	`if (x == 1) {`
	`  cout << "x is 1";`
	`}`
	`else if (x == 2) {`
	`  cout << "x is 2";`
	`}`
	`else {`
	`  cout << "value of x unknown";`
	`}`

Functions

- http://www.cplusplus.com/doc/tutorial/functions/

```cpp
// function example
#include <iostream>
using namespace std;

int addition (int a, int b)
{
  int r;
  r=a+b;
  return r;
}

int main ()
{
  int z;
  z = addition (5,3);
  cout << "The result is " << z;
}
```

```cpp
// void function example
#include <iostream>
using namespace std;

void printmessage ()
{
  cout << "I'm a function!";
}

int main ()
{
  printmessage ();
}
```

Functions

```
5  void duplicate (int& a, int& b, int& c)
6  {
7    a*=2;
8    b*=2;
9    c*=2;
10 }
11
12 int main ()
13 {
14   int x=1, y=3, z=7;
15   duplicate (x, y, z);
16   cout << "x=" << x << ", y=" << y << ", z=" << z;
17   return 0;
18 }
```

```
1  inline string concatenate (const string& a, const string& b)
2  {
3    return a+b;
4  }
```

Overloading functions

- http://www.cplusplus.com/doc/tutorial/functions2/

```
5  int operate (int a, int b)
6  {
7    return (a*b);
8  }
9
10 double operate (double a, double b)
11 {
12   return (a/b);
13 }
14
15 int main ()
16 {
17   int x=5,y=2;
18   double n=5.0,m=2.0;
19   cout << operate (x,y) << '\n';
20   cout << operate (n,m) << '\n';
21   return 0;
22 }
```

Name visibility

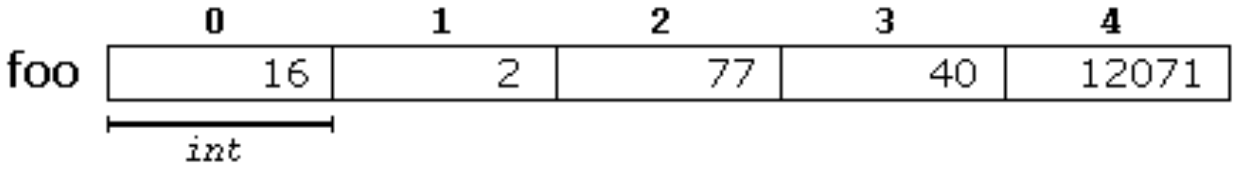
- http://www.cplusplus.com/doc/tutorial/namespaces/

```
1  int foo;         // global variable
2
3  int some_function ()
4  {
5    int bar;       // local variable
6    bar = 0;
7  }
8
9  int other_function ()
10 {
11   foo = 1;  // ok: foo is a global variable
12   bar = 2;  // wrong: bar is not visible from this function
13 }
```

Arrays

- http://www.cplusplus.com/doc/tutorial/arrays/

```
int foo [5] = { 16, 2, 77, 40, 12071 };
```




```
5  int foo [] = {16, 2, 77, 40, 12071};
6  int n, result=0;
7
8  int main ()
9  {
10   for ( n=0 ; n<5 ; ++n )
11   {
12     result += foo[n];
13   }
14   cout << result;
15   return 0;
16 }
```

Arrays

- Options to define multidimensional arrays (→ data grids)

multidimensional array

```
#define WIDTH 5
#define HEIGHT 3

int jimmy [HEIGHT][WIDTH];
int n,m;

int main ()
{
  for (n=0; n<HEIGHT; n++)
    for (m=0; m<WIDTH; m++)
    {
      jimmy[n][m]=(n+1)*(m+1);
    }
}
```

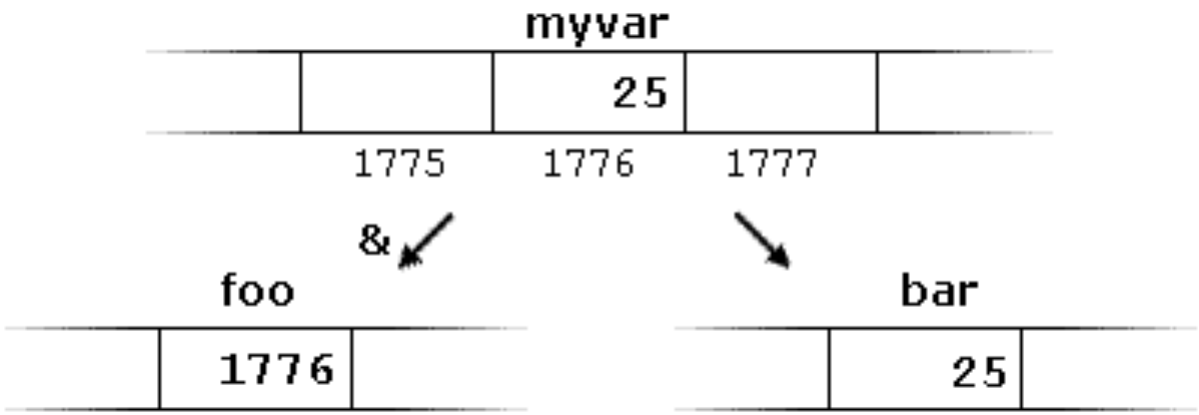
pseudo-multidimensional array

```
#define WIDTH 5
#define HEIGHT 3

int jimmy [HEIGHT * WIDTH];
int n,m;

int main ()
{
  for (n=0; n<HEIGHT; n++)
    for (m=0; m<WIDTH; m++)
    {
      jimmy[n*WIDTH+m]=(n+1)*(m+1);
    }
}
```

Pointers (Zeiger)

- http://www.cplusplus.com/doc/tutorial/pointers/

```
1 myvar = 25;
2 foo = &myvar;
3 bar = myvar;
```



Pointers (Zeiger)

```
1  // more pointers
2  #include <iostream>
3  using namespace std;
4
5  int main ()
6  {
7    int firstvalue = 5, secondvalue = 15;
8    int * p1, * p2;
9
10   p1 = &firstvalue;  // p1 = address of firstvalue
11   p2 = &secondvalue; // p2 = address of secondvalue
12   *p1 = 10;          // value pointed to by p1 = 10
13   *p2 = *p1;         // value pointed to by p2 = value pointed to by p1
14   p1 = p2;           // p1 = p2 (value of pointer is copied)
15   *p1 = 20;          // value pointed to by p1 = 20
16
17   cout << "firstvalue is " << firstvalue << '\n';
18   cout << "secondvalue is " << secondvalue << '\n';
19   return 0;
20 }
```

Pointers and arrays

```
1 int myarray [20];
2 int * mypointer;
```

```
 1 // more pointers
 2 #include <iostream>
 3 using namespace std;
 4
 5 int main ()
 6 {
 7   int numbers[5];
 8   int * p;
 9   p = numbers;  *p = 10;
10   p++;  *p = 20;
11   p = &numbers[2];  *p = 30;
12   p = numbers + 3;  *p = 40;
13   p = numbers;  *(p+4) = 50;
14   for (int n=0; n<5; n++)
15     cout << numbers[n] << ", ";
16   return 0;
17 }
```

Dynamic memory

- http://www.cplusplus.com/doc/tutorial/dynamic/

```
 8  int i,n;
 9  int * p;
10  cout << "How many numbers would you like to type? ";
11  cin >> i;
12  p= new (nothrow) int[i];
13  if (p == nullptr)
14    cout << "Error: memory could not be allocated";
15  else
16  {
17    for (n=0; n<i; n++)
18    {
19      cout << "Enter number: ";
20      cin >> p[n];
21    }
22    cout << "You have entered: ";
23    for (n=0; n<i; n++)
24      cout << p[n] << ", ";
25    delete[] p;
26  }
```

```
1 int * foo;
2 foo = new int [5];
```

```
1 delete pointer;
2 delete[] pointer;
```

Structures

- http://www.cplusplus.com/doc/tutorial/structures/

```
struct type_name {
member_type1 member_name1;
member_type2 member_name2;
member_type3 member_name3;
.
.
} object_names;
```

```
1 struct movies_t {
2   string title;
3   int year;
4 };
5
6 movies_t amovie;
7 movies_t * pmovie;
```

pmovie->title

```
1 struct product {
2   int weight;
3   double price;
4 } ;
5
6 product apple;
7 product banana, melon;
```

```
1 apple.weight
2 apple.price
3 banana.weight
4 banana.price
5 melon.weight
6 melon.price
```

Classes

- http://www.cplusplus.com/doc/tutorial/classes/
- Classes = structures with members
- Access specifiers: public, protected, private

```
class class_name {
  access_specifier_1:
    member1;
  access_specifier_2:
    member2;
  ...
} object_names;
```

```
1 class Rectangle {
2     int width, height;
3   public:
4     void set_values (int,int);
5     int area (void);
6 } rect;
```

```
1  // example: one class, two objects
2  #include <iostream>
3  using namespace std;
4
5  class Rectangle {
6      int width, height;
7    public:
8      void set_values (int,int);
9      int area () {return width*height;}
10 };
11
12 void Rectangle::set_values (int x, int y) {
13   width = x;
14   height = y;
15 }
16
17 int main () {
18   Rectangle rect, rectb;
19   rect.set_values (3,4);
20   rectb.set_values (5,6);
21   cout << "rect area: " << rect.area() << endl;
22   cout << "rectb area: " << rectb.area() << endl;
23   return 0;
24 }
```

What if ommited?

```
1  // example: class constructor
2  #include <iostream>
3  using namespace std;
4
5  class Rectangle {
6      int width, height;
7    public:
8      Rectangle (int,int);
9      int area () {return (width*height);}
10 };
11
12 Rectangle::Rectangle (int a, int b) {
13   width = a;
14   height = b;
15 }
16
17 int main () {
18   Rectangle rect (3,4);
19   Rectangle rectb (5,6);
20   cout << "rect area: " << rect.area() << endl;
21   cout << "rectb area: " << rectb.area() << endl;
22   return 0;
23 }
```

Constructor

```cpp
// destructors
#include <iostream>
#include <string>
using namespace std;

class Example4 {
    string* ptr;
  public:
    // constructors:
    Example4() : ptr(new string) {}
    Example4 (const string& str) : ptr(new string(str)) {}
    // destructor:
    ~Example4 () {delete ptr;}
    // access content:
    const string& content() const {return *ptr;}
};

int main () {
  Example4 foo;
  Example4 bar ("Example");

  cout << "bar's content: " << bar.content() << '\n';
  return 0;
}
```

Destructor

Übung: Unterschiede zwischen C++ und Java

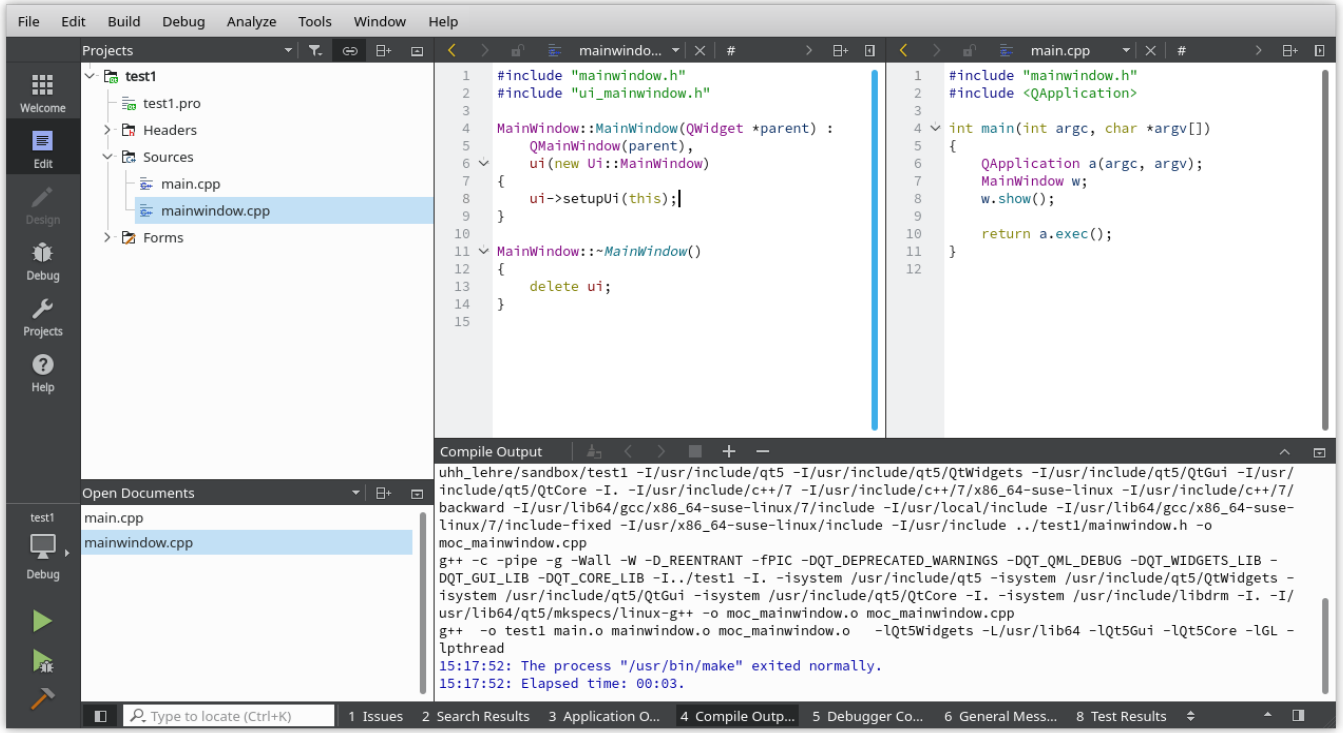
Diskutiert in Break-Out Gruppen die Gemeinsamkeiten und Unterschiede zwischen C++ und Java.

- Was sind die **Besonderheiten** der jeweiligen Sprache?
- Welche **Vorteile** und **Nachteile** haben die jeweiligen Sprachen?
- Wie unterschieden sich die Sprachen in den folgenden Punkten?
  - z.B. Header- und Sourcefile, Deklarationen und Definitionen, Syntax, Ausführung von Code (Stichwort: Compiler), Systemnahe und systemferne Sprache, Speichermanagement

→ Diskutiert diese oder auch andere Punkte für ca. 10 Minuten in eurer Gruppe.

Entwicklungsumgebung: QtCreator unter Linux

- Integrated Development Environment (IDE), speziell auf Qt zugeschnitten
- Für unsere Zwecke praktisch: Open-source, einfach zu bedienen und bietet Unterstützung für Verwendung von Qt-Elementen in C++.
- Siehe: https://doc.qt.io/qtcreator/index.html



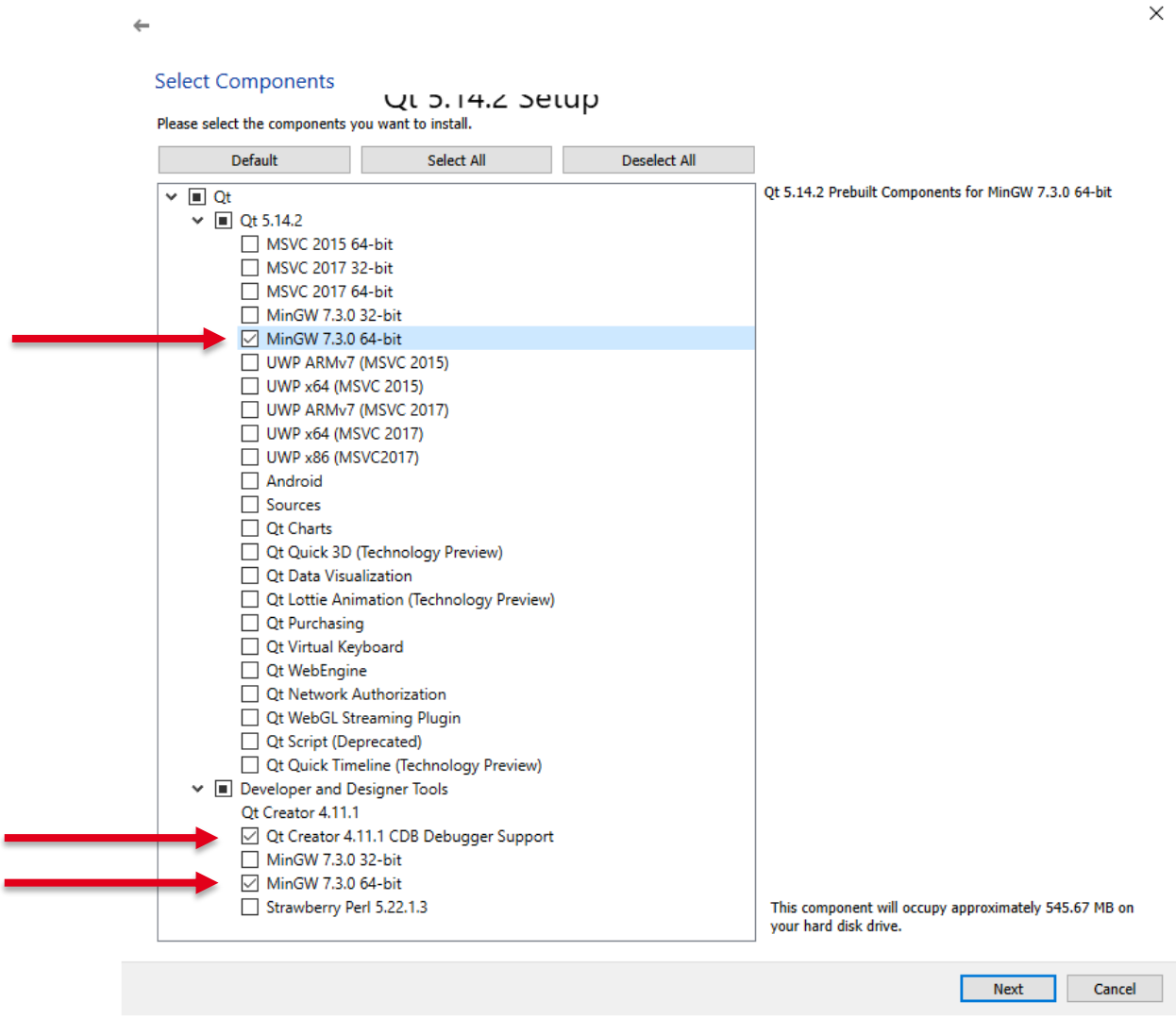
Vorbereitung für die nächste Übung: Installation von Qt

- Linux: In den gängigen Distributionen (OpenSuSE, Ubuntu, ..) lässt sich der QtCreator aus den Standard-Repositories installieren. Das sollte alle für die Entwicklung benötigten Bibliotheken mitinstallieren.

- Windows: Qt lässt sich einfach per Installer installieren, inklusive QtCreator und der MinGW C++ Compiler-Suite. Leider musste ich feststellen, dass Qt sich früher in diesem Jahr entschieden hat, einen „Qt Account“ zu erfordern. Ich persönlich mag nicht, überall Accounts anzulegen. Wer damit kein Problem hat kann sich Qt über den Online-Installer installieren. Ansonsten gibt es ein Workaround: Den Offline-Installer herunterladen (> 2 GB!). Dann vor dem Start des Installers die Internetverbindung kappen (z.B. WiFi ausstellen). Der Installer fragt zwar nach dem Account, übergeht das dann aber.
  - https://www.qt.io/offline-installers

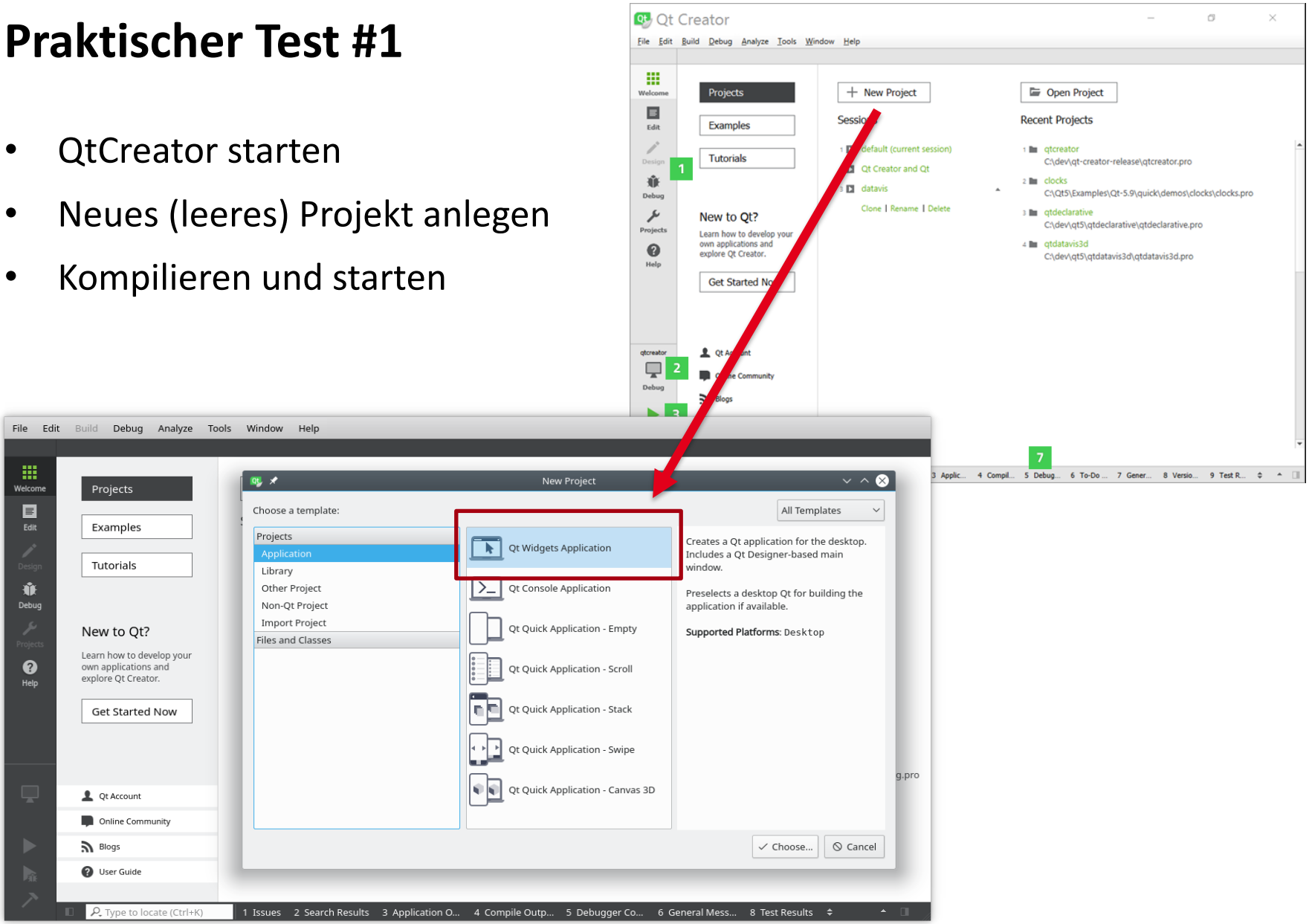
- MacOS: ???

Installation von Qt unter Windows



Praktischer Test #1

- QtCreator starten
- Neues (leeres) Projekt anlegen
- Kompilieren und starten



Praktischer Test #2

- „Hello World“ im QtCreator:

```
1 // my first program in C++
2 #include <iostream>
3
4 int main()
5 {
6   std::cout << "Hello World!";
7 }
```