Vorkurs Informatik (D-ITET)

Malte Schwerhoff

September 2018

http://lec.inf.ethz.ch/itet/informatik0/2018

Kontext dieser Vorlesung

- **Informatik 0** (Vorkurs): Erste Programmiererfahrungen sammeln

Kontext dieser Vorlesung

- **Informatik 0** (Vorkurs): Erste Programmiererfahrungen sammeln
- **Informatik 1**: Theoretische und praktische Grundlagen der Informatik
- **Informatik 2**: Algorithmen und Datenstrukturen

Kontext dieser Vorlesung

- **Informatik 0** (Vorkurs): Erste Programmiererfahrungen sammeln
- **Informatik 1**: Theoretische und praktische Grundlagen der Informatik
- **Informatik 2**: Algorithmen und Datenstrukturen
- **Technische Informatik 1**: Logische und physikalische Strukturen digitaler Systeme
- **Technische Informatik 2**: Wichtige Komponenten von Betriebssystemen

Kontext dieser Vorlesung

- **Informatik 0** (Vorkurs): Erste Programmiererfahrungen sammeln
- **Informatik 1**: Theoretische und praktische Grundlagen der Informatik
- **Informatik 2**: Algorithmen und Datenstrukturen
- **Technische Informatik 1**: Logische und physikalische Strukturen digitaler Systeme
- **Technische Informatik 2**: Wichtige Komponenten von Betriebssystemen
- ... und weitere Fächer mit (mehr oder weniger direktem) Informatikbezug

Kursziele

- Erste Programmiererfahrung vermitteln
- An der „Informatik-Oberfläche“ kratzen

Kursziele

- Erste Programmiererfahrung vermitteln
- An der „Informatik-Oberfläche“ kratzen
- Dieser Kurs kann leider nicht jahrelange Programmiererfahrung (Schule, Hobby) ausgleichen ...

Kursziele

- Erste Programmiererfahrung vermitteln
- An der „Informatik-Oberfläche“ kratzen
- Dieser Kurs kann leider nicht jahrelange Programmiererfahrung (Schule, Hobby) ausgleichen ...
- ... aber er sollte Ihnen den Einstieg in Informatik 1 erleichtern

Kursaufbau: Leistungskontrolle

- Keine Prüfung
- Stattdessen müssen zwei Programmierprojekte bestanden werden

Kursaufbau: Stundenplan

Woche(n)	Programm
1	Vorlesung: Do 20.09., 13:15 – 15:00 C++-Tutorial, 1. Projekt

Kursaufbau: Stundenplan

Woche(n)	Programm
1	Vorlesung: Do 20.09., 13:15 – 15:00 C++-Tutorial, 1. Projekt
2	Präsenzstunden: Mo 24.09. HG F1, Di 25.09. HG E7, 17:00 – 19:00 1. Projekt

Kursaufbau: Stundenplan

Woche(n)	Programm
1	Vorlesung: Do 20.09., 13:15 – 15:00 C++-Tutorial, 1. Projekt
2	Präsenzstunden: Mo 24.09. HG F1, Di 25.09. HG E7, 17:00 – 19:00 1. Projekt
3	Abgabe 1. Projekt Vorlesung: Mi 03.10., 13:15 – 15:00 Präsenzstunden: Mo 01.10. HG F1, Di 02.10. HG E7, 17:00 – 19:00 2. Projekt

Kursaufbau: Stundenplan

Woche(n)	Programm
1	Vorlesung: Do 20.09., 13:15 – 15:00 C++-Tutorial, 1. Projekt
2	Präsenzstunden: Mo 24.09. HG F1, Di 25.09. HG E7, 17:00 – 19:00 1. Projekt
3	Abgabe 1. Projekt Vorlesung: Mi 03.10., 13:15 – 15:00 Präsenzstunden: Mo 01.10. HG F1, Di 02.10. HG E7, 17:00 – 19:00 2. Projekt
4-7	Präsenzstunden: Di 9./16./23./30.10, HG E7, 17:00 – 19:00 2. Projekt

Kursaufbau: Stundenplan

Woche(n)	Programm
1	Vorlesung: Do 20.09., 13:15 – 15:00 C++-Tutorial, 1. Projekt
2	Präsenzstunden: Mo 24.09. HG F1, Di 25.09. HG E7, 17:00 – 19:00 1. Projekt
3	Abgabe 1. Projekt Vorlesung: Mi 03.10., 13:15 – 15:00 Präsenzstunden: Mo 01.10. HG F1, Di 02.10. HG E7, 17:00 – 19:00 2. Projekt
4-7	Präsenzstunden: Di 9./16./23./30.10, HG E7, 17:00 – 19:00 2. Projekt
7	Abgabe 2. Projekt

1. Einführung

Informatik: Definition und Geschichte, Algorithmen, Turing Maschine, Höhere Programmiersprachen, Werkzeuge der Programmierung, Das erste C++ Programm und seine syntaktischen und semantischen Bestandteile

Was ist Informatik?

Was ist Informatik?

- Die Wissenschaft der **systematischen Verarbeitung von Informationen**,. . .

Was ist Informatik?

- Die Wissenschaft der **systematischen Verarbeitung von Informationen**,. . .
- . . . insbesondere der automatischen Verarbeitung mit Hilfe von Digitalrechnern.

(Wikipedia, nach dem „Duden Informatik“)

Informatik vs. Computer

Computer science is not about machines, in the same way that astronomy is not about telescopes.

Mike Fellows, US-Informatiker (1991)

Informatik vs. Computer

- Die Informatik beschäftigt sich heute auch mit dem Entwurf von schnellen Computern und Netzwerken. . .

Informatik vs. Computer

- Die Informatik beschäftigt sich heute auch mit dem Entwurf von schnellen Computern und Netzwerken. . .
- . . . aber nicht als Selbstzweck, sondern zur effizienteren **systematischen Verarbeitung von Informationen**.

Informatik $\neq$ EDV-Kenntnisse

EDV-Kenntnisse: *Anwenderwissen*

- Umgang mit dem Computer
- Bedienung von Computerprogrammen (für Texterfassung, E-Mail, Präsentationen,. . .)

Informatik $\neq$ EDV-Kenntnisse

Informatik: *Grundlagenwissen*

- Wie funktioniert ein Computer?
- Wie schreibt man ein Computerprogramm?

Algorithmus: Kernbegriff der Informatik

Algorithmus:

- Handlungsanweisung zur schrittweisen Lösung eines Problems

Algorithmus: Kernbegriff der Informatik

Algorithmus:

- Handlungsanweisung zur schrittweisen Lösung eines Problems
- Ausführung erfordert keine Intelligenz, nur Genauigkeit (sogar Computer können es)

Algorithmus: Kernbegriff der Informatik

Algorithmus:

- Handlungsanweisung zur schrittweisen Lösung eines Problems
- Ausführung erfordert keine Intelligenz, nur Genauigkeit (sogar Computer können es)
- Ältester nichttrivialer Algorithmus:
  Euklidischer Algorithmus, 3. Jh. v. Chr.

Algorithmus: Kernbegriff der Informatik

Algorithmus:

- Handlungsanweisung zur schrittweisen Lösung eines Problems
- Ausführung erfordert keine Intelligenz, nur Genauigkeit (sogar Computer können es)
- Ältester nichttrivialer Algorithmus: Euklidischer Algorithmus, 3. Jh. v. Chr.
- nach *Muhammed al-Chwarizmi*, Autor eines arabischen Rechen-Lehrbuchs (um 825)

“Dixit algorizmi…” (lateinische Übersetzung)

Binäre Suche: Problem & Idee

Problem: Finde ein Element in einer *sortierten* Liste

Binäre Suche: Problem & Idee

Problem: Finde ein Element in einer *sortierten* Liste

Idee: Wörterbuchsuche — mittig aufschlagen, falls nötig links/rechts weitersuchen

Binäre Suche: Beispiel

Beispiel: Finde 7 in der Liste $[1, 3, 4, 7, 9, 11, 12, 17]$

1	3	4	7	9	11	12	17

Binäre Suche: Beispiel

Beispiel: Finde 7 in der Liste $[1, 3, 4, 7, 9, 11, 12, 17]$

Binäre Suche: Beispiel

Beispiel: Finde 7 in der Liste $[1, 3, 4, 7, 9, 11, 12, 17]$

Binäre Suche: Beispiel

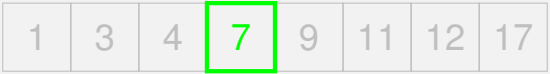
Beispiel: Finde 7 in der Liste $[1, 3, 4, 7, 9, 11, 12, 17]$

Binäre Suche: Beispiel

Beispiel: Finde 7 in der Liste $[1, 3, 4, 7, 9, 11, 12, 17]$

Binäre Suche: Beispiel

Beispiel: Finde 7 in der Liste $[1, 3, 4, 7, 9, 11, 12, 17]$



Binäre Suche: Pseudocode

- Eingabe: Liste sortierter Zahlen L, zu findende Zahl n
- Ausgabe: „Ja“ („Nein“) falls n (nicht) in L

Solange Ausgabe A noch unbekannt ist:

Binäre Suche: Pseudocode

- Eingabe: Liste sortierter Zahlen L, zu findende Zahl n
- Ausgabe: „Ja“ („Nein“) falls n (nicht) in L

Solange Ausgabe A noch unbekannt ist:
    Falls L leer dann: $A \leftarrow$ „Nein“

Binäre Suche: Pseudocode

- Eingabe: Liste sortierter Zahlen L, zu findende Zahl n
- Ausgabe: „Ja“ („Nein“) falls n (nicht) in L

Solange Ausgabe A noch unbekannt ist:
    Falls L leer dann: $A \leftarrow$ „Nein“
    Andernfalls:
        Wähle mittige Zahl L_m aus

Binäre Suche: Pseudocode

- Eingabe: Liste sortierter Zahlen L, zu findende Zahl n
- Ausgabe: „Ja“ („Nein“) falls n (nicht) in L

Solange Ausgabe A noch unbekannt ist:
    Falls L leer dann: $A \leftarrow$ „Nein“
    Andernfalls:
        Wähle mittige Zahl L_m aus
        Falls $L_m = n$ dann: $A \leftarrow$ „Ja“

Binäre Suche: Pseudocode

- Eingabe: Liste sortierter Zahlen L, zu findende Zahl n
- Ausgabe: „Ja“ („Nein“) falls n (nicht) in L

Solange Ausgabe A noch unbekannt ist:
    Falls L leer dann: $A \leftarrow$ „Nein“
    Andernfalls:
        Wähle mittige Zahl L_m aus
        Falls $L_m = n$ dann: $A \leftarrow$ „Ja“
        Falls $n < L_m$ dann: $L \leftarrow$ linke Hälfte von L

Binäre Suche: Pseudocode

- Eingabe: Liste sortierter Zahlen L, zu findende Zahl n
- Ausgabe: „Ja“ („Nein“) falls n (nicht) in L

Solange Ausgabe A noch unbekannt ist:
    Falls L leer dann: $A \leftarrow$ „Nein“
    Andernfalls:
        Wähle mittige Zahl L_m aus
        Falls $L_m = n$ dann: $A \leftarrow$ „Ja“
        Falls $n < L_m$ dann: $L \leftarrow$ linke Hälfte von L
        Ansonsten: $L \leftarrow$ rechte Hälfte von L

Binäre Suche: C++-Implementierung

```
std::string binary_search(const int* begin, const int* end,
                          const int n) {
  while (true) {
    if (begin >= end) return "no";
    const int* mid = begin + (end - begin) / 2;
    if (*mid == n) return "yes";
    else if (n < *mid) end = mid;
    else begin = mid + 1;
  }
}
```

Keine Panik: Der C++-Code wird hier nur gezeigt um den Unterschied zwischen einer Algorithmusbeschreibung in Pseudocode und einer konkreten Implementierung zu illustrieren. Die genutzten Sprachkonstrukte werden Sie erst in Informatik 1 kennenlernen.

Algorithmen: 3 Abstraktionsstufen

1. **Kernidee** (abstrakt)**:**
   Die Essenz eines Algorithmus' („Heureka-Moment“)

Algorithmen: 3 Abstraktionsstufen

1. **Kernidee** (abstrakt)**:**
   Die Essenz eines Algorithmus' („Heureka-Moment“)
2. **Pseudocode** (semi-detailliert)**:**
   Für Menschen gemacht (Bildung, Korrektheit- und Effizienzdiskussionen, Beweise)

Algorithmen: 3 Abstraktionsstufen

1. **Kernidee** (abstrakt)**:**
   Die Essenz eines Algorithmus' („Heureka-Moment“)
2. **Pseudocode** (semi-detailliert)**:**
   Für Menschen gemacht (Bildung, Korrektheit- und Effizienzdiskussionen, Beweise)
3. **Implementierung** (sehr detailliert)**:**
   Für Mensch & Computer gemacht (les- & ausführbar, bestimmte Programmiersprache, verschiedene Implementierungen möglich)

Warum Programmieren?

- Da hätte ich ja gleich Informatik studieren können ...

Warum Programmieren?

- Da hätte ich ja gleich Informatik studieren können ...
- Es gibt doch schon für alles Programme ...

Warum Programmieren?

- Da hätte ich ja gleich Informatik studieren können ...
- Es gibt doch schon für alles Programme ...
- Programmieren interessiert mich nicht ...

Warum Programmieren?

- Da hätte ich ja gleich Informatik studieren können ...
- Es gibt doch schon für alles Programme ...
- Programmieren interessiert mich nicht ...
- Weil Informatik hier leider ein Pflichtfach ist ...

Warum Programmieren?

- Da hätte ich ja gleich Informatik studieren können ...
- Es gibt doch schon für alles Programme ...
- Programmieren interessiert mich nicht ...
- Weil Informatik hier leider ein Pflichtfach ist ...
- ...

Mathematik war früher die Lingua franca der Naturwissenschaften an allen Hochschulen. Und heute ist dies die Informatik.

Lino Guzzella, Präsident der ETH Zürich, NZZ Online, 1.9.2017

(Lino Guzzella ist übrigens nicht Informatiker, sondern Maschineningenieur und Prof. für Thermotronik ☺)

Darum Programmieren!

- Jedes Verständnis moderner Technologie erfordert Wissen über die grundlegende Funktionsweise eines Computers.
- Programmieren (mit dem Werkzeug Computer) wird zu einer Kulturtechnik wie Lesen und Schreiben (mit den Werkzeugen Papier und Bleistift)

Darum Programmieren!

- Jedes Verständnis moderner Technologie erfordert Wissen über die grundlegende Funktionsweise eines Computers.
- Programmieren (mit dem Werkzeug Computer) wird zu einer Kulturtechnik wie Lesen und Schreiben (mit den Werkzeugen Papier und Bleistift)
- Programmieren ist *die* Schnittstelle zwischen Ingenieurwissenschaften und Informatik – der interdisziplinäre Grenzbereich wächst zusehends.

Darum Programmieren!

- Jedes Verständnis moderner Technologie erfordert Wissen über die grundlegende Funktionsweise eines Computers.
- Programmieren (mit dem Werkzeug Computer) wird zu einer Kulturtechnik wie Lesen und Schreiben (mit den Werkzeugen Papier und Bleistift)
- Programmieren ist *die* Schnittstelle zwischen Ingenieurwissenschaften und Informatik – der interdisziplinäre Grenzbereich wächst zusehends.
- Programmieren macht Spass (und ist nützlich)!

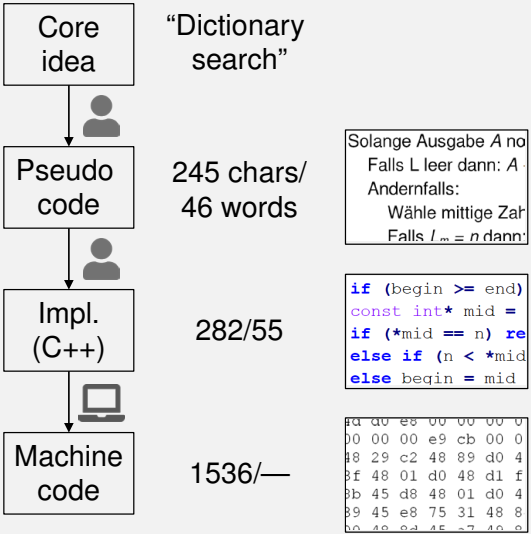
Programmieren

- Mit Hilfe einer *Programmiersprache* wird dem Computer eine Folge von Befehlen erteilt, damit er genau das macht, was wir wollen.
- Die Folge von Befehlen ist das *(Computer)-Programm*.

The Harvard Computers, Menschliche Berufsrechner, ca.1890

Programmiersprachen

- Sprache, die der Computer „versteht“, ist sehr primitiv (Maschinensprache).
- Einfache Operationen müssen in (extrem) viele Einzelschritte aufgeteilt werden.
- Sprache variiert von Computer zu Computer.



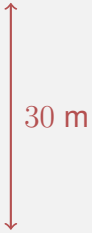
Rechengeschwindigkeit

In der mittleren Zeit, die der Schall von mir zu Ihnen unterwegs ist...

[1] Uniprozessor Computer bei 1GHz

Rechengeschwindigkeit

In der mittleren Zeit, die der Schall von mir zu Ihnen unterwegs ist...



arbeitet ein heutiger Desktop-PC mehr als 100

[1] Uniprozessor Computer bei 1GHz

Rechengeschwindigkeit

In der mittleren Zeit, die der Schall von mir zu Ihnen unterwegs ist...

30 m $\widehat{=}$ mehr als $100.000.000$ Instruktionen

arbeitet ein heutiger Desktop-PC mehr als 100 Millionen Instruktionen ab.[1]

[1] Uniprozessor Computer bei 1GHz

Höhere Programmiersprachen

Wir schreiben Programme (Implementierungen) in einer *höheren Programmiersprache*:

- Kann von Menschen *verstanden* werden kann
- Ist *hardwareunabhängig*
- Beinhaltet *wiederverwendbare* Funktionsbibliotheken

Warum C++?

Andere populäre höhere Programmiersprachen: Java, C#, Python, Javascript, Swift, Kotlin, Go, ...

Warum C++?

Andere populäre höhere Programmiersprachen: Java, C#, Python, Javascript, Swift, Kotlin, Go, ...

- C++ ist relevant in der Praxis, weit verbreitet und „läuft überall“
- C++ ist standardisiert, d.h. es gibt ein offizielles

Warum C++?

Andere populäre höhere Programmiersprachen: Java, C#, Python, Javascript, Swift, Kotlin, Go, ...

- C++ ist relevant in der Praxis, weit verbreitet und „läuft überall“
- C++ ist standardisiert, d.h. es gibt ein offizielles
- C++ ist eine der „schnellsten“ Programmiersprachen
- C++ eignet sich gut für Systemprogrammierung da es einen sorgfältigen Umgang mit Ressourcen (Speicher, ...) erlaubt/verlangt

Syntax und Semantik

- Programme müssen, wie unsere Sprache, nach gewissen Regeln geformt werden.
  - *Syntax*: Zusammenfügungsregeln für elementare Zeichen (Buchstaben).
  - *Semantik*: Interpretationsregeln für zusammengefügte Zeichen.

Syntax und Semantik

- Programme müssen, wie unsere Sprache, nach gewissen Regeln geformt werden.
  - *Syntax*: Zusammenfügungsregeln für elementare Zeichen (Buchstaben).
  - *Semantik*: Interpretationsregeln für zusammengefügte Zeichen.
- Entsprechende Regeln für ein Computerprogramm sind einfacher, aber auch strenger, denn Computer sind vergleichsweise dumm.

Deutsch vs. C++

Deutsch

Alleen sind nicht gefährlich, Rasen ist gefährlich!
(Wikipedia: Mehrdeutigkeit)

C++

```
// computation
int b = a * a; // b = a^2
b = b * b;     // b = a^4
```

C++: Fehlerarten illustriert an deutscher Sprache

- Das Auto fuhr zu schnell.

C++: Fehlerarten illustriert an deutscher Sprache

- Das Auto fuhr zu schnell.

Syntaktisch und semantisch korrekt.

C++: Fehlerarten illustriert an deutscher Sprache

- DasAuto fuh r zu sxhnell.

C++: Fehlerarten illustriert an deutscher Sprache

- DasAuto fuh r zu sxhnell.

Syntaxfehler: Wortbildung.

C++: Fehlerarten illustriert an deutscher Sprache

- Rot das Auto ist.

C++: Fehlerarten illustriert an deutscher Sprache

- Rot das Auto ist.

Syntaxfehler: Satzstellung.

C++: Fehlerarten illustriert an deutscher Sprache

- Man empfiehlt dem Dozenten
  nicht zu widersprechen

C++: Fehlerarten illustriert an deutscher Sprache

- Man empfiehlt dem Dozenten, nicht zu widersprechen.

Syntaxfehler: Satzzeichen fehlen .

C++: Fehlerarten illustriert an deutscher Sprache

- Sie ist nicht gross und rothaarig.

C++: Fehlerarten illustriert an deutscher Sprache

- Sie ist nicht gross und rothaarig.

Syntaktisch korrekt aber mehrdeutig. [kein Analogon]

C++: Fehlerarten illustriert an deutscher Sprache

- Die Auto ist rot.

C++: Fehlerarten illustriert an deutscher Sprache

- Die Auto ist rot.

Syntaktisch korrekt, doch semantisch fehlerhaft: Falscher Artikel. [Typfehler]

C++: Fehlerarten illustriert an deutscher Sprache

- Das Fahrrad galoppiert schnell.

C++: Fehlerarten illustriert an deutscher Sprache

- Das Fahrrad galoppiert schnell.

Syntaktisch und grammatikalisch korrekt! Semantisch fehlerhaft. [Laufzeitfehler]

C++: Fehlerarten illustriert an deutscher Sprache

- Manche Tiere riechen gut.

C++: Fehlerarten illustriert an deutscher Sprache

- Manche Tiere riechen gut.

Syntaktisch und semantisch korrekt. Semantisch mehrdeutig. [kein Analogon]

Syntax und Semantik von C++

Syntax:

- Wann ist ein Text ein C++Programm?
- D.h. ist es *grammatikalisch* korrekt?
- $\rightarrow$ Kann vom Computer überprüft werden

Semantik:

- Was *bedeutet* ein Programm?
- Welchen Algorithmus *implementiert* ein Programm?
- $\rightarrow$ Braucht menschliches Verständnis

Was braucht es zum Programmieren?

- **Editor:** Programm zum Ändern, Erfassen und Speichern von C++-Programmtext
- **Compiler:** Programm zum Übersetzen des Programmtexts in Maschinensprache

Was braucht es zum Programmieren?

- **Editor:** Programm zum Ändern, Erfassen und Speichern von C++-Programmtext
- **Compiler:** Programm zum Übersetzen des Programmtexts in Maschinensprache
- **Computer:** Gerät zum Ausführen von Programmen in Maschinensprache
- **Betriebssystem:** Programm zur Organisation all dieser Abläufe (Dateiverwaltung, Editor-, Compiler- und Programmaufruf)

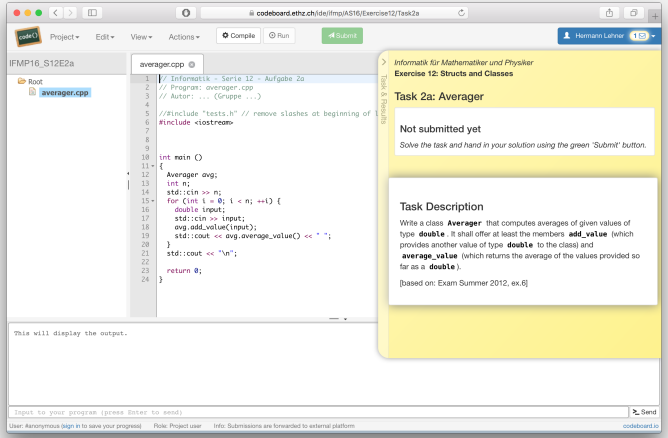
2. Demo: Projekt 1, Teilaufgabe 1

Die Vorlesung wird aufgezeichnet, Sie brauchen also nicht mitschreiben 🙂

3. Organisation der Programmierprojekte

Codeboard

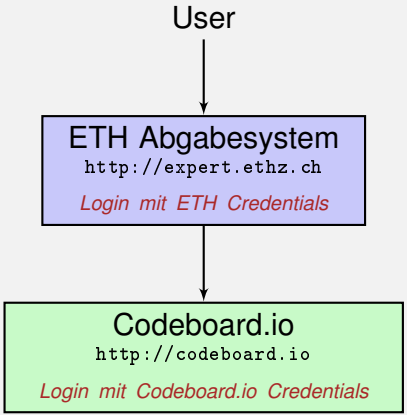
Codeboard ist eine Online-IDE: Programmieren im Browser!



Code Expert

Unser Übungssystem besteht aus zwei unabhängigen Systemen, die miteinander kommunizieren:

- **Das ETH Abgabesystem:** Ermöglicht es uns, Ihre Aufgaben zu bewerten
- **Die Online IDE:** Die Programmierumgebung



Projekte = Übungen

- Code Expert ist für einen „normalen“ Übungsbetrieb ausgelegt, daher die Terminologie „Übung“, „Übungsgruppe“, etc.
- Auf Code Expert sind unsere Programmierprojekte daher Übungen und es gibt genau eine Übungsgruppe

Einschreibung

Codeboard.io Registrierung

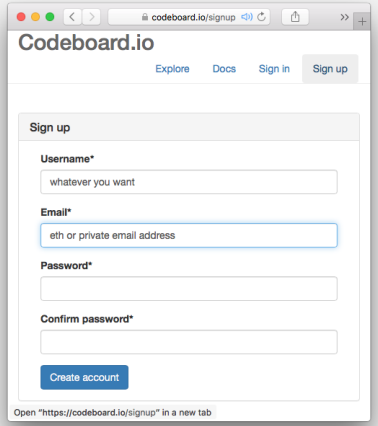
Gehen Sie auf `http://codeboard.io` und erstellen Sie dort ein Konto, bleiben Sie am besten eingeloggt.

Einschreibung auf

Gehen Sie auf `http://expert.ethz.ch/inf0itet18` und schreiben Sie sich in die Übungsgruppe `Students` ein.

Codeboard.io Registrierung

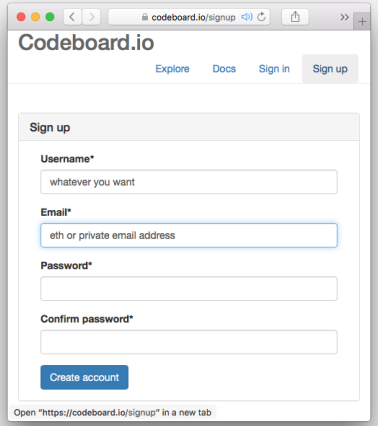
Falls Sie noch keinen **Codeboard.io** Account haben ...



- Wir verwenden die Online IDE **Codeboard.io**

Codeboard.io Registrierung

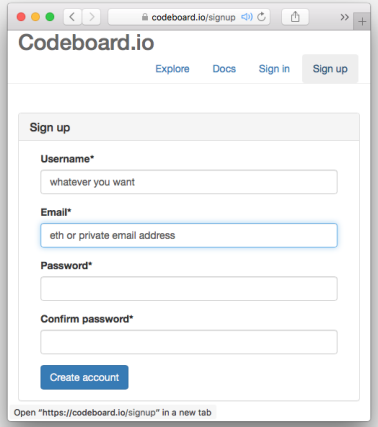
Falls Sie noch keinen **Codeboard.io** Account haben ...



- Wir verwenden die Online IDE **Codeboard.io**
- Erstellen Sie dort einen Account, um Ihren Fortschritt abzuspeichern und später Submissions anzuschauen

Codeboard.io Registrierung

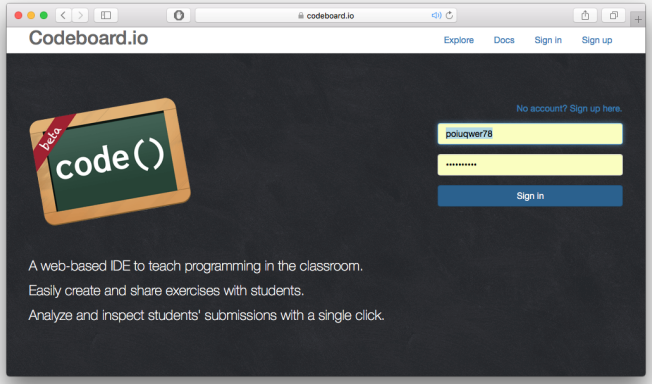
Falls Sie noch keinen **Codeboard.io** Account haben ...



- Wir verwenden die Online IDE **Codeboard.io**
- Erstellen Sie dort einen Account, um Ihren Fortschritt abzuspeichern und später Submissions anzuschauen
- Anmeldedaten können beliebig gewählt werden! *Verwenden Sie nicht das ETH Passwort.*

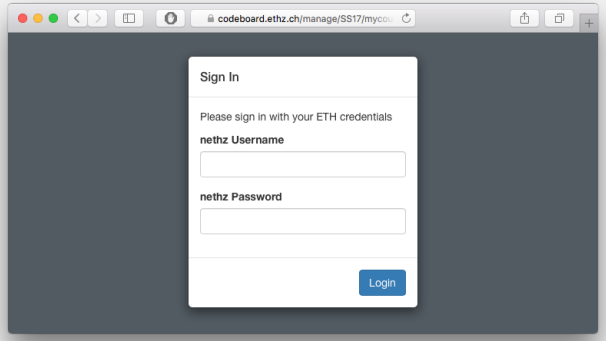
Codeboard.io Login

Falls Sie schon einen Account haben, loggen Sie sich ein:



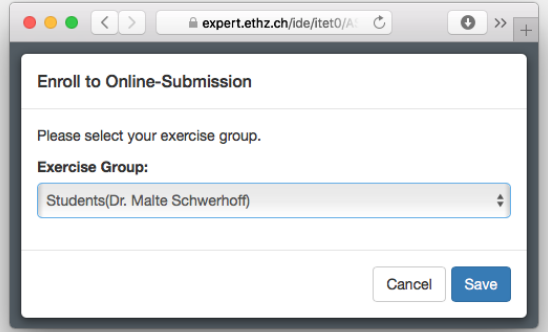
Einschreibung in Übungsgruppen - I

- Besuchen Sie `http://expert.ethz.ch/inf0itet18`
- Loggen Sie sich mit Ihrem nethz-Account ein.



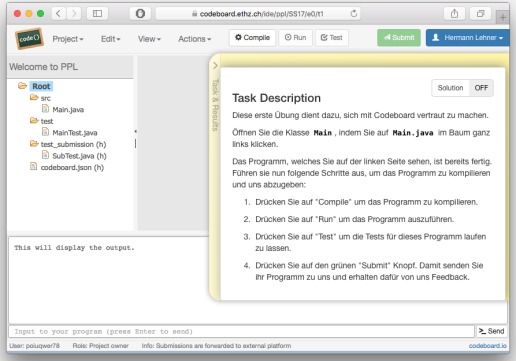
Einschreibung in Übungsgruppen - II

Schreiben Sie sich in diesem Dialog in die Übungsgruppe `Students` ein.



Die erste Übung

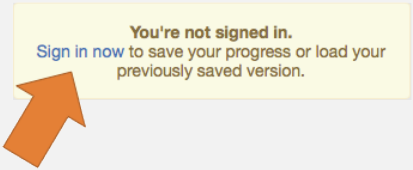
Sie sind nun eingeschrieben und die erste Übung ist geladen.
Folgen Sie den Anweisungen in der gelben Box.



(Screenshot nicht aktuell)

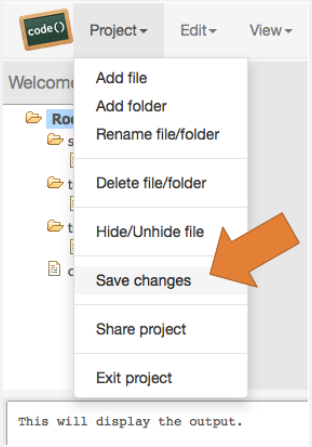
Achtung: Codeboard.io-Login

Achtung! Falls Sie diese Nachricht sehen, klicken Sie auf Sign in now und melden Sie sich dort mit Ihrem **Codeboard.io**-Account ein.



Achtung: Fortschritt speichern

Achtung! Speichern Sie Ihren Fortschritt regelmässig ab. So können Sie jederzeit an einem anderen Ort weiterarbeiten.



Akademische Lauterkeit

Es gilt die Leistungskontrollenverordnung der ETH Zürich

Regel: Sie geben nur eigene Lösungen ab, welche Sie selbst verfasst und verstanden haben.

Wir prüfen das (zum Teil automatisiert) nach und behalten uns disziplinarische Massnahmen vor.

So ist's OK (siehe Algorithmen-Folien):

- Kernidee: gerne in der Gruppe diskutieren
- Pseudocode: gerne in der Gruppe diskutieren
- Implementierung: *Einzelarbeit!*