Einführung

Parameter	Kursinformationen
Veranstaltung:	`Vorlesung Prozedurale Programmierung / Einführung in die Informatik`
Semester	`Wintersemester 2022/23`
Hochschule:	`Technische Universität Freiberg`
Inhalte:	`Vorstellung des Arbeitsprozesses`
Link auf Repository:	https://github.com/TUBAF-IfI-LiaScript/VL_ProzeduraleProgrammierung/blob/master/00_Einfuehrung.md
Autoren	Sebastian Zug & André Dietrich & Galina Rudolf

Fragen an die heutige Veranstaltung ...

- Welche Aufgabe erfüllt eine Programmiersprache?
- Erklären Sie die Begriffe Compiler, Editor, Programm, Hochsprache!
- Was passiert beim Kompilieren eines Programmes?
- Warum sind Kommentare von zentraler Bedeutung?
- Worin unterscheiden sich ein konventionelles C++ Programm und eine Anwendung, die mit dem Arduino-Framework geschrieben wurde?

Umfrage

Hat Sie die letztwöchige Vorstellung der Ziele der Lehrveranstaltung überzeugt?

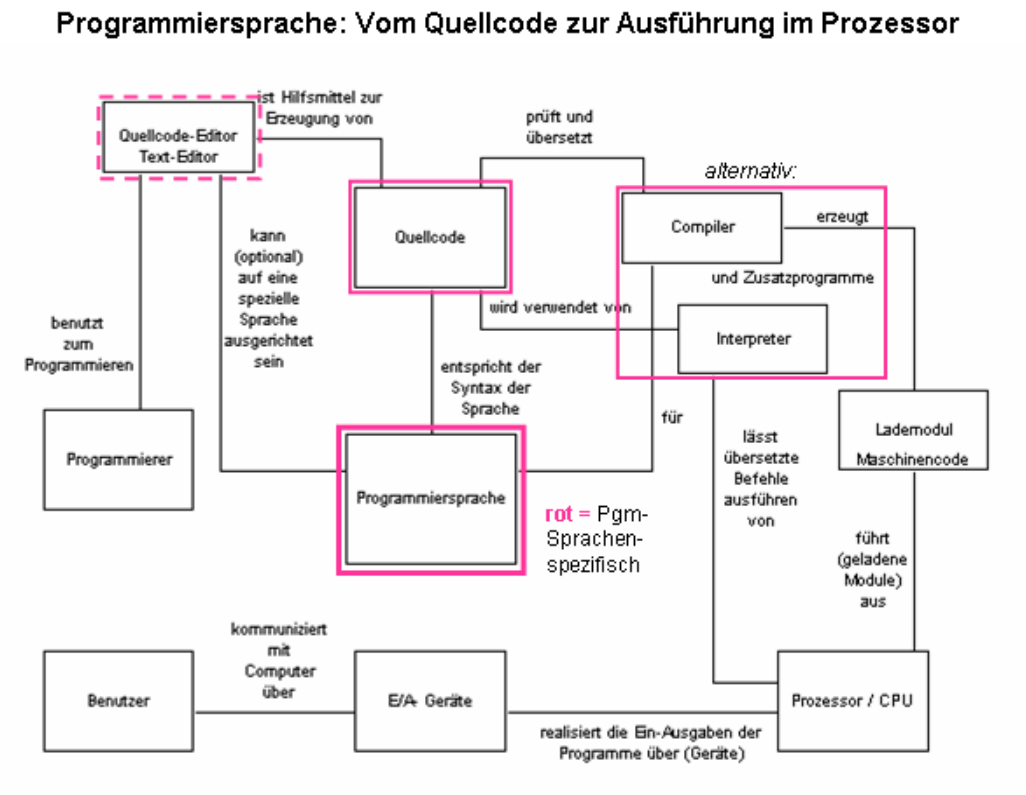
- ◯ Ja, ich gehe davon aus, viel nützliches zu erfahren.
- ◯ Ich bin noch nicht sicher. Fragen Sie in einigen Wochen noch mal.
- ◯ Nein, ich bin nur hier, weil ich muss.

Wie arbeitet ein Rechner eigentlich?

Programme sind Anweisungslisten, die vom Menschen erdacht, auf einem Rechner zur Ausführung kommen. Eine zentrale Hürde ist dabei die Kluft zwischen menschlicher Vorstellungskraft und Logik, die **implizite Annahmen und Erfahrungen** einschließt und der **"stupiden" Abarbeitung von Befehlsfolgen** in einem Rechner.

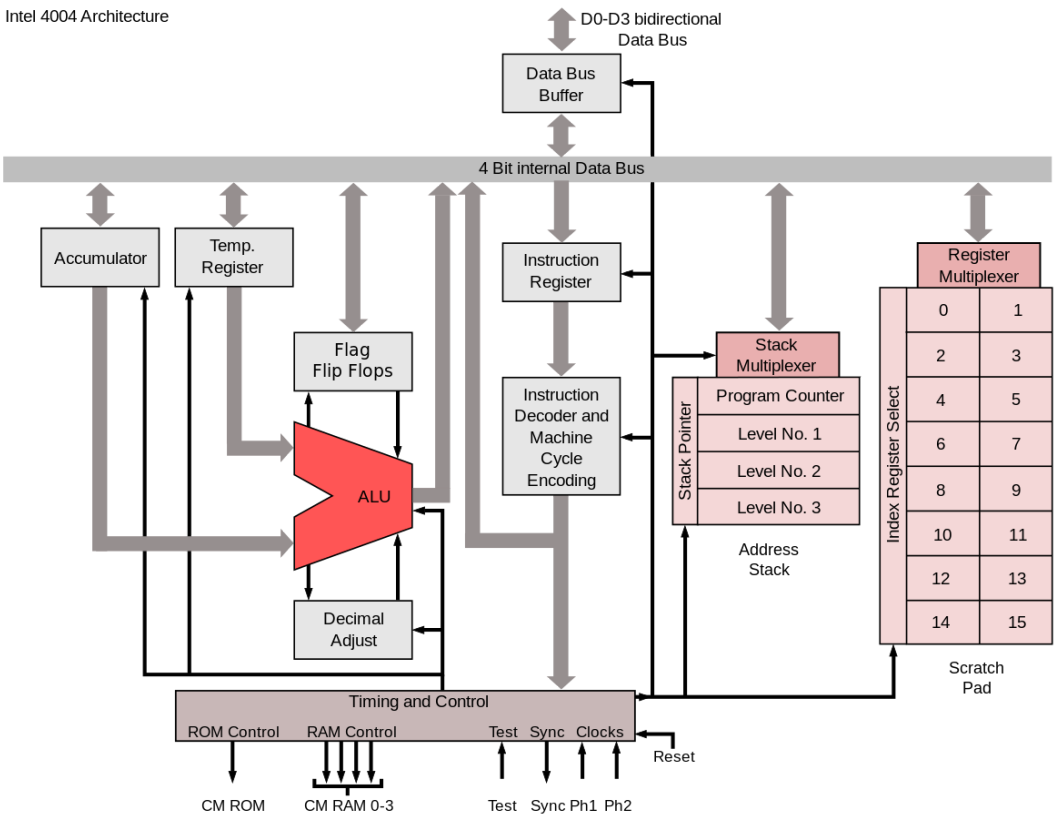
Programmiersprachen bemühen sich diese Lücke zu schließen und werden dabei von einer Vielzahl von Tools begleitet, diesen **Transformationsprozess** unterstützen sollen.

Um das Verständnis für diesen Vorgang zu entwickeln werden zunächst die Vorgänge in einem Rechner bei der Abarbeitung von Programmen beleuchtet, um dann die Realisierung eines Programmes mit C++ zu adressieren.



Erzeugung von Programmcode [Programmerstellung]

Beispiel: Intel 4004-Architektur (1971)



Intel 4004 Architekturdarstellung [Intel4004]

Jeder Rechner hat einen spezifischen Satz von Befehlen, die durch "0" und "1" ausgedrückt werden, die er überhaupt abarbeiten kann.

Speicherauszug den Intel 4004:

Adresse	Speicherinhalt	OpCode	Mnemonik
0010	1101 0101	1101 DDDD	LD $5
0012	1111 0010	1111 0010	IAC

Unterstützung für die Interpretation aus dem Nutzerhandbuch, dass das *Instruction Set* beschreibt:

Quelle: Intel 4004 Assembler

[Intel4004] Autor Appaloosa, Intel 4004, https://upload.wikimedia.org/wikipedia/commons/thumb/8/87/4004_arch.svg/1190px-4004_arch.svg.png

[Programmerstellung] Programmiervorgang und Begriffe(Autor VÖRBY, https://commons.wikimedia.org/wiki/File:Programmiersprache_Umfeld.png

Programmierung

Möchte man so Programme schreiben?

Vorteil:

- ggf. sehr effizienter Code (Größe, Ausführungsdauer), der gut auf die Hardware abgestimmt ist

Nachteile:

- systemspezifische Realisierung
- geringer Abstraktionsgrad, bereits einfache Konstrukte benötigen viele Codezeilen
- weitgehende semantische Analysen möglich

Eine höhere Programmiersprache ist eine Programmiersprache zur Abfassung eines Computerprogramms, die in **Abstraktion und Komplexität** von der Ebene der Maschinensprachen deutlich entfernt ist. Die Befehle müssen durch **Interpreter oder Compiler** in Maschinensprache übersetzt werden.

Ein **Compiler** (auch Kompiler; von englisch für zusammentragen bzw. lateinisch compilare ‚aufhäufen') ist ein Computerprogramm, das Quellcodes einer bestimmten Programmiersprache in eine Form übersetzt, die von einem Computer (direkter) ausgeführt werden kann.

Stufen des Compile-Vorganges:

Stufen der Compilierung [Compiliation]

[Compiliation] Jimmy Thong, Oct 13, 2016, What happens when you type GCC main.c, https://medium.com/@vietkieutie/what-happens-when-you-type-gcc-main-c-2a136896ade3

Einordnung von C und C++

- Adressiert Hochsprachenaspekte und Hardwarenähe → Hohe Geschwindigkeit bei geringer Programmgröße
- Imperative Programmiersprache

  imperative (befehlsorientierte) Programmiersprachen: Ein Programm besteht aus einer Folge von Befehlen an den Computer. Das Programm beschreibt den Lösungsweg für ein Problem (C, Python, Java, LabView, Matlab, ...).

  deklarative Programiersprachen: Ein Programm beschreibt die allgemeinen Eigenschaften von Objekten und ihre Beziehungen untereinander. Das Programm beschreibt zunächst nur das Wissen zur Lösung des Problems (Prolog, Haskal, SQL, ...).

- Wenige Schlüsselwörter als Sprachumfang

  Schlüsselwort Reserviertes Wort, das der Compiler verwendet, um ein Programm zu parsen (z.B. if, def oder while). Schlüsselwörter dürfen nicht als Name für eine Variable gewählt werden

- Große Mächtigkeit

  Je "höher" und komfortabler die Sprache, desto mehr ist der Programmierer daran gebunden, die in ihr vorgesehenen Wege zu beschreiten.

Erstes C++ Programm

"Hello World"

HelloWorld.c

```
1  // That's my first C program
2  // Karl Klammer, Oct. 2022
3
4  #include <iostream>
5
6  int main() {
7      std::cout << "Hello World!";
8      return 0;
9  }
```

```
Failed to execute 'send' on 'WebSocket': Still in CONNECTING state.
```

Zeile	Bedeutung
1 - 2	Kommentar (wird vom Präprozessor entfernt)
4	Voraussetzung für das Einbinden von Befehlen der Standardbibliothek hier `std:cout()`
6	Einsprungstelle für den Beginn des Programmes
6 - 9	Ausführungsblock der `main`-Funktion
7	Anwendung eines Operators `<<` hier zur Ausgabe auf dem Bildschirm
8	Definition eines Rückgabewertes für das Betriebssystem

Halt! Unsere C++ Arduino Programme sahen doch ganz anders aus?

BuggyCode.cpp

```
1  void setup() {
2    pinMode(LED_BUILTIN, OUTPUT);
3  }
4
5  void loop() {
6    digitalWrite(LED_BUILTIN, HIGH);
7    delay(1000);
8    digitalWrite(LED_BUILTIN, LOW);
9    delay(1000);
10 }
```

```
Sketch uses 924 bytes (2%) of program storage space. Maximum is 32256
bytes.
Global variables use 9 bytes (0%) of dynamic memory, leaving 2039 bytes
for local variables. Maximum is 2048 bytes.
```

Noch mal Halt! Das klappt ja offenbar alles im Browserfenster, aber wenn ich ein Programm auf meinem Rechner kompilieren möchte, was ist dann zu tuen?

Ein Wort zu den Formalien

HelloWorld.cpp

```
1  // Karl Klammer
2  // Print Hello World drei mal
3
4  #include <iostream>
5
6  int main() {
7    int zahl;
8    for (zahl=0; zahl<3;  zahl++){
9        std::cout << "Hello World! ";
10   }
11   return 0;
12 }
```

```
Failed to execute 'send' on 'WebSocket': Still in CONNECTING state.
```

BadHelloWorld.cpp

```
1  #include <iostream>
2  int main() {int zahl; for (zahl=0; zahl<3;  zahl++){ std::cout << "Hel
     World! ";} return 0;}
```

```
Failed to execute 'send' on 'WebSocket': Still in CONNECTING state.
```

- Das *systematische Einrücken* verbessert die Lesbarkeit und senkt damit die Fehleranfälligkeit Ihres Codes!
- Wählen sie *selbsterklärende Variablen- und Funktionsnamen*!
- Nutzen Sie ein *Versionsmanagmentsystem*, wenn Sie ihren Code entwickeln!
- Kommentieren Sie Ihr Vorgehen trotz "Good code is self-documenting"

Gute Kommentare

1. Kommentare als Pseudocode

```
/* loop backwards through all elements returned by the server
(they should be processed chronologically)*/
for (i = (numElementsReturned - 1); i >= 0; i--){
    /* process each element's data */
    updatePattern(i, returnedElements[i]);
}
```

2. Kommentare zur Datei

```
// This is the mars rover control application
//
// Karl Klammer, Oct. 2018
// Version 109.1.12

int main(){...}
```

3. Beschreibung eines Algorithmus

```
/* Function:  approx_pi
 * --------------------
 * computes an approximation of pi using:
 *    pi/6 = 1/2 + (1/2 x 3/4) 1/5 (1/2)^3  + (1/2 x 3/4 x 5/6) 1/7 (1/2)^5
 *
 *  n: number of terms in the series to sum
 *
 *  returns: the approximate value of pi obtained by suming the first n ter
```

```
 *           in the above series
 *           returns zero on error (if n is non-positive)
 */

 double approx_pi(int n);
```

In realen Projekten werden Sie für diese Aufgaben Dokumentationstools verwenden, die die Generierung von Webseite, Handbüchern auf der Basis eigener Schlüsselworte in den Kommentaren unterstützen → doxygen.

4. Debugging

```
int main(){
  ...
  preProcessedData = filter1(rawData);
  // printf('Filter1 finished ... \n');
  // printf('Output %d \n', preProcessedData);
  result=complexCalculation(preProcessedData);
  ...
}
```

Schlechte Kommentare

1. Überkommentierung von Code

```
x = x + 1;  /* increment the value of x */
std::cout << "Hello World! "; // displays Hello world
```

"... over-commenting your code can be as bad as under-commenting it"

Quelle: C Code Style Guidelines

2. "Merkwürdige Kommentare"

```
//When I wrote this, only God and I understood what I was doing
//Now, God only knows

// sometimes I believe compiler ignores all my comments

// Magic. Do not touch.
Hello World !Hello World !Hello World !Hello World !Hello World !Hello Wor
  !Hello World !Hello World !Hello Wor
// I am not responsible of this code.

try {

} catch(e) {
```

```
} finally { // should never happen }
```

Sammlung von Kommentaren

Was tun, wenn es schief geht?

ErroneousHelloWorld.cpp

```
1  // Karl Klammer
2  // Print Hello World drei mal
3
4  #include <iostream>
5
6  int main() {
7    for (zahl=0; zahl<3;  zahl++){
8        std::cout << "Hello World! "
9    }
10   return 0;
11
```

```
Failed to execute 'send' on 'WebSocket': Still in CONNECTING state.
```

Methodisches Vorgehen:

- **RUHE BEWAHREN**
- Lesen der Fehlermeldung
- Verstehen der Fehlermeldung / Aufstellen von Hypothesen
- Systematische Evaluation der Thesen
- Seien Sie im Austausch mit anderen (Kommilitonen, Forenbesucher, usw.) konkret

Compilerfehlermeldungen

Beispiel 1

Error.cpp

```
1  #include <iostream>
2
3  int mani() {
4      std::cout << "Hello World!";
5      return 0;
6  }
```

```
Failed to execute 'send' on 'WebSocket': Still in CONNECTING state.
```

Beispiel 2

Error.cpp

```cpp
#include <iostream>

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

```
Failed to execute 'send' on 'WebSocket': Still in CONNECTING state.
```

Manchmal muss man sehr genau hinschauen, um zu verstehen, warum ein Programm nicht funktioniert. Versuchen Sie es!

ErroneousHelloWorld.cpp

```cpp
#include <iostream>

imt main() {
  std::cout << "Hello World!'';
  std::cout << "Wo liegt der Fehler?";
  return 0;
}
```

```
Failed to execute 'send' on 'WebSocket': Still in CONNECTING state.
```

Und wenn das Kompilieren gut geht?

... dann bedeutet es noch immer nicht, dass Ihr Programm wie erwartet funktioniert.

ErroneousHelloWorld.cpp

```cpp
#include <iostream>

int main() {
  char zahl;
  for (zahl=250; zahl<256; zahl++){
      std::cout << "Hello World!";
  }
  return 0;
}
```

```
Failed to execute 'send' on 'WebSocket': Still in CONNECTING state.
```

> **Hinweis:** Die Datentypen werden wir in der nächsten Woche besprechen.

Warum dann C++?

Zwei Varianten der Umsetzung ... C++ vs. Python

HelloWorld.cpp

```cpp
#include <iostream>

int main() {
  char zahl;
  for (int zahl=0; zahl<3; zahl++){
      std::cout << "Hello World! " << zahl << "\n";
  }
  return 0;
}
```

```
Failed to execute 'send' on 'WebSocket': Still in CONNECTING state.
```

```python
for i in range(3):
  print("Hallo World ", i)
```

```
Failed to execute 'send' on 'WebSocket': Still in CONNECTING state.
```

Beispiele der Woche

Gültiger C++ Code

Output.cpp

```
#include <iostream>

imt main() {
  int i = 5;
  int j = 4;
  i = i + j + 2;
  std::cout << "Hello World ";
  std::cout << i << "!";
  return 0;
}
```

Umfrage: Welche Ausgabe erwarten Sie für folgendes Code-Schnippselchen?

- ◯ `Hello World7`
- ◯ `Hello World11`
- ◯ `Hello World 11!`
- ◯ `Hello World 11 !`
- ◯ `Hello World 5 !`

Algorithmisches Denken

Aufgabe: Ändern Sie den Code so ab, dass das die LED zwei mal mit 1 Hz blinkt und dann ausgeschaltet bleibt.

BuggyCode.cpp

```cpp
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
  digitalWrite(LED_BUILTIN, HIGH);
  delay(1000);
  digitalWrite(LED_BUILTIN, LOW);
  delay(1000);
}
```

```
Sketch uses 924 bytes (2%) of program storage space. Maximum is 32256
bytes.
Global variables use 9 bytes (0%) of dynamic memory, leaving 2039 bytes
for local variables. Maximum is 2048 bytes.
```

<!--START_SKIP_IN_PDF-->

Quiz

Einbinden von Bibliotheken

Wie müssen Bibliotheken in C++ eingebunden werden?

- using iostream
- import iostream
- #include <iostream>

Kommentare

Sind diese Kommentare angebracht?

```cpp
#include <iostream>                    // I always forget this xD

int main() {                           // main-function
```

```
  char zahl;                          // I have homework due tomorrow :(
  for (zahl=250; zahl<256; zahl++){ // I really have no idea what this ever
    but whatever... LOL
      std::cout << "Hello World!";  // Prints "Hello World!" in console
  }
  return 0;
}
```

- ◯ Ja
- ◯ Nein

Konventionen und Formalien

Wählen Sie alle Programme aus, die den üblichen Formalien entsprechen.

1.cpp

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

2.cpp

```
#include <iostream>

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

3.cpp

```
#include                    <iostream>

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

- [ ] 1.cpp
- [ ] 2.cpp
- [ ] 3.cpp

<!--END*SKIP*IN_PDF⟶