Materialpaket 11_PUTT – Putting it all together

C/C++, Autor: Prof. Dr.-Ing. Carsten Link

Version 1.3.1 March 3, 2019

Contents

1 Kompetenzen und Lernegebnisse **1**

2 Konzepte **2**

3 Prüfungsvorbereitung **2**
- 3.1 Verständnisfragen zu 01_ENV . . . 2
- 3.2 Verständnisfragen zu 02_MENT . . . 2
- 3.3 Aufgaben zu 02_MENT . . . 3
- 3.4 Aufgaben zu 2_FLOW_a . . . 3
- 3.5 Verständnisfragen zu 03_FLOW_c . . . 3
- 3.6 Aufgaben zu 03_FLOW_c . . . 4
- 3.7 Aufgaben zu 05_OO_b . . . 4
- 3.8 Aufgaben zu 06_POLY . . . 5
- 3.9 Aufgaben zu 07_STD . . . 6
- 3.10 Aufgaben zu 10_PITF . . . 6
  - 3.10.1 Copy on Write . . . 6

4 Nützliche Links **7**

5 Literatur **7**

1 Kompetenzen und Lernegebnisse

Durch das Bearbeiten dieses Materialpaketes erwerben Sie diese Kompetenzen (Wissen, Fähigkeiten, Fertigkeiten zur Problemlösung):

Sie können die isoliert vorgestellten Inhalte der vorangegangenen Materialpakete in Kombination anwenden.

Die oben genannten Kompetenzen erwerben Sie, indem Sie Lernziele erreichen, welche sich prüfen lassen. Lernegebnisse: Sie können nachweislich[1]:

- Aufgaben, wie sie in einer Prüfung gestellt werden können und die alle vorangegangenen Themenbereiche betreffen, erfolgreich bearbeiten
- Fragen, wie sie in einer Prüfung gestellt werden können und die alle vorangegangenen Themenbereiche betreffen, richtig beantworten

2 Konzepte

Im Folgenden wird kein neuer Inhalt dargestellt. Vielmehr sollen Sie alles bereits Gelernte miteinander kombinieren, um les- und wartbare, effiziente C++-Programme entwickeln zu können.

3 Prüfungsvorbereitung

In diesem Materialpaket finden sich Aufgaben und Verständnisfragen, die über jene hinausgehen, die sich in den vorangegangenen Materialpaketen befinden, da sie Wissen und Fertigkeiten benötigen, die über das im jeweiligen Materialpaket Vorgestellte hinausgehen.

3.1 Verständnisfragen zu 01_ENV

1. Sie haben zwei Dateien `main.cpp` und `func1.cpp` mit entsprechenden Header-Dateien. In `func1.cpp` ist eine Funktion `func1` definiert. Beim Erzeugen eines ausführbaren Programms bekommen Sie eine Fehlermeldung `unresolved external: func1`. Was ist die Ursache?
2. Lückentext. Füllen Sie die Lücken in folgendem Text: Ein C++-Programm besteht aus ... -Dateien und ... -Dateien. Der ... fügt in erstere letztere ein und ersetzt ... . Vom ... wird ... -Code erzeugt, der vom ... in ... -Code übersetzt wird. Dabei entstehen ... -Dateien, die vom ... zu einem ausführbaren Programm zusammengesetzt werden.

3.2 Verständnisfragen zu 02_MENT

1. Gegeben sei die Zeichenkette `char * s = "0123"`. Wenn Sie das zugrundeliegende Bitmuster im Speicher uminterpretieren lassen (z. B. mit `int i = (*((int*)s));` bewusst das Bitmuster falsch interpretieren), welche Werte ergeben sich für die Typen `char` und `int`?

[1] Sie können das Erzielen der einzelnen Lernergebnisse beispielsweise bei einem Testat im Praktikum oder einer Aufgabe in der Modulprüfung nachweisen

2. Eine Funktion verfügt über eine lokale Variable `local_int` (ist darin deklariert); das Programm, in dem sich die Funktion befindet, hat eine globale Variable `global_int`. Worin unterscheiden sich `local_int` und `global_int`?
3. Wie oft existiert `local_int` während des Programmablaufs?
4. Wie oft kann `local_int` während des Programmablaufs *gleichzeitig* existieren?
5. Stellen Sie sich eine Tabelle vor (z. B. aus dem Spiel "Stadt, Land, Fluss" oder eine Tabelle, welche nur Zahlenwerte enthält wie beispielsweise Umfrageergebnisse "Alter in Jahren, Gewicht in kg, Körpergröße in cm"). Ordnen Sie nun die beiden Aggregatstypen struct und array ein. Welche Möglichkeiten Ergeben sich, die Tabelle im Speicher eines C++-Programmes darzustellen?
6. Welchen Typ haben die unten angegebenen Ausdrücke (bei mehreren: der letzte)?

```
1 char *s = "1.0"; *s
2 s[2]
3 4 / 3
```

3.3 Aufgaben zu 02_MENT

1. Laden sie das virtuelle LC-Display[2] herunter und experimentieren Sie damit. Erstellen Sie ein Programm, welches einen `int` hochzählt und diesen mittig auf dem Display darstellt.

3.4 Aufgaben zu 2_FLOW_a

- Implementieren Sie den FloodFill-Algorithmus rekursiv. Erstellen Sie hierzu

z. B. ein zweidimensionales `char`-Array (ggf. `std::array<>`), welches mit Leerzeichen, einem Rahmen und Hindernissen gefüllt ist.

3.5 Verständnisfragen zu 03_FLOW_c

1. Welche Vorteile hat es, einen Algorithmus rekursiv statt iterativ umzusetzen?

[2] http://www.technik-emden.de/~clink/projects/2016w-ProjGrp/05_Website_ohne_Quellcodes/11_Dokumente/13_Doxygen/doc/html/index.html

3.6 Aufgaben zu 03_FLOW_c

1. Rechner für geklammerte Ausdrücke ohne Rekursion: Implementieren Sie den Rechner für geklammerte boolsche Ausdrücke ohne Rekursion. Benutzen Sie stattdessen `std::verctor<>`, um zwei Stacks zu realisieren.

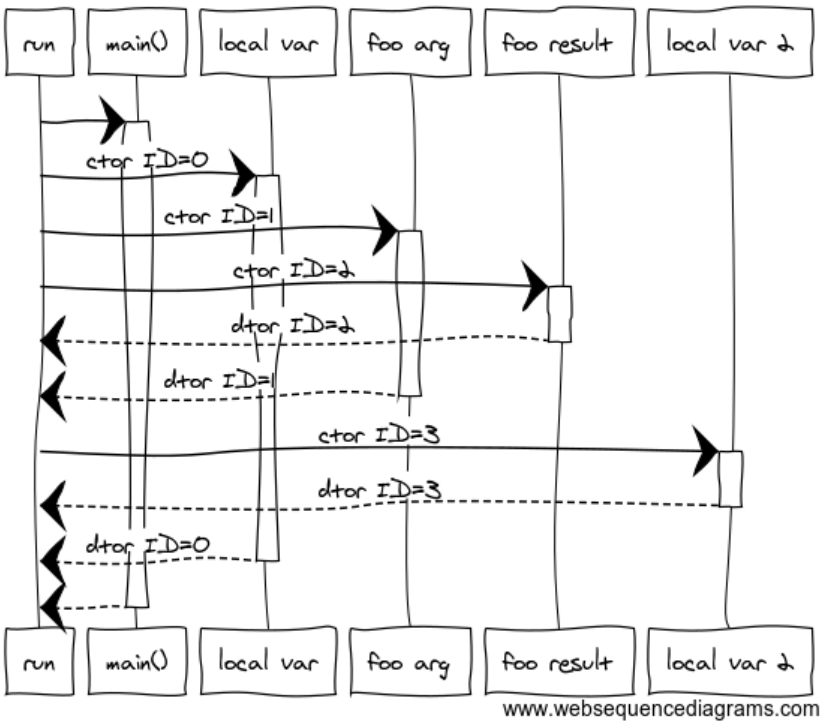
3.7 Aufgaben zu 05_OO_b

1. Betrachten Sie folgenden Code:

```
1  struct A{
2  };
3
4  struct B{
5  };
6
7
8  class K {
9   A a;
10 };
11
12 class M : public K {
13  B b;
14 };
15
16 M * m =  new M();
17 delete m;
```

In welcher Reihenfolge werden welche special member functions aufgerufen?

2. Auf der Web-Seite https://www.websequencediagrams.com lassen sich UML-Sequenzdiagramme erstellen, welche in Textform definiert sind. Diese lassen sich zweckentfremden, um die Lebensdauer von Objekten zu visualisieren (siehe nachfolgende Abbildung Websequence-Diagrams). Erstellen Sie eine Klasse `SequenceDiagramCreator`, welche im Konstruktor und Destruktor dafür sorgt, dass Text ausgegeben wird, der von `websequencediagrams.com` interpretiert werden kann.



Experimentieren Sie anschließend mit globalen, lokalen und statischen Variablen. Ebenso sollten Sie die Parameterübergabe erkunden.

3.8 Aufgaben zu 06_POLY

1. Erstellen Sie jeweils ein kurzes einfaches Beispielprogramm für
   a. ad-hoc polymorphism
   b. subtyping polymorhism
   c. parametric polymorphism
2. Betrachten Sie folgende Klassendeklarationen:

```
1  class TimePiece {
2  public:
3    virtual ~TimePiece();
4    void foo();
5    virtual std::string toString();
6  };
7
8  class AnalogWatch : public TimePiece {
9  public:
10   void foo();
```

```
11    void bar();
12    virtual std::string toString();
13  };
14
15  class DigitalWatch : TimePiece {
16  public:
17    void foo();
18    void bar();
19    virtual std::string toString();
20  };
```

Was geschieht bei den folgenden Anweisungen?

```
1  TimePiece * ta = new AnalogWatch();
2  TimePiece * td = new DigitalWatch();
3  TimePiece * tp = ta;
4
5  tp->toString();
6  tp->foo();
7  ta->foo();
8  td->foo();
9  td->bar();
10 tp->bar();
```

3. Entwickeln Sie ein kleines Programm, welches Funktionszeiger verwendet, um die Häufigkeit von Buchstaben, Ziffern, und sonstigen Zeichen in einem `std::string` zu ermitteln. Sie können beispielsweise ein Array anlegen, in dem für jedes Zeichen ein Funktionszeiger hinterlegt ist (die Funktionen hinter den Zeigern könnten Variablen hochzählen).

3.9 Aufgaben zu 07_STD

1. Verwenden Sie `IntStack` oder `std::vector<>`, um den Callstack nachzubilden. Das heißt: erstellen Sei einige `void/void`-Funktionen (ohne Parameter und Rückgabewert), welche sich gegenseitig aufrufen (gern auch rekursiv, ggf. indirekt) und die Parameter und Rückgabewerte über eine Instanz von `IntStack` oder `std::vector<>` austauschen.

3.10 Aufgaben zu 10_PITF

3.10.1 Copy on Write

Um Werte in einem Programm zu Speichern, welche sehr viel Speicherplatz benötigen, bietet sich das Idiom bzw. Design Pattern *copy on write* an. Hierbei verwenden mehrere Kopien ein und denselben Speicherplatz; allerdings nur

so lange, bis eine Änderung (*write*) durchgeführt wird: dann wird für das Modifizierte Objekt eine eigene Kopie erstellt, welche dann verändert wird.

Aufgabe: nehme Sie die Dateien `OneByOneMatrix.h`, `OneByOneMatrix.cpp` und `main.cpp` im Verzeichnis `11_PUTT/CopyOnWrite/` als Vorlage.

Compilieren Sie nun mit den `typedef`s `int` und `OneByOneMatrix` und lassen das Programm jeweils laufen.

```
1 typedef int NumberType;
2 //typedef OneByOneMatrix NumberType;
3 //typedef LargeCowMatrix NumberType;
```

Compilieren Sie nun mit dem `typedef LargeCowMatrix`. Verändern Sie den Code, so dass keine Fehler beim Übersetzen auftreten und keine Fehler zur Laufzeit auftreten.

4 Nützliche Links

- Stack Overflow: http://stackoverflow.com
- C++ reference: http://en.cppreference.com/w/
- C++ Referenz: http://de.cppreference.com/w/

5 Literatur

- [PPP] Stroustrup, Bjarne: Programming - Principles and Practice using C++
- [TCPL] Stroustrup, Bjarne: The C++ Programming Language, Fourth Edition
- [CUEB] U. Kirch, P. Prinz: C++ das Übungsbuch, Testfragen und Aufgaben mit Lösungen