C/C++ Materialpaket (Level C)
01a_SMPL – Simple Example

Prof. Dr. Carsten Link

Zusammenfassung

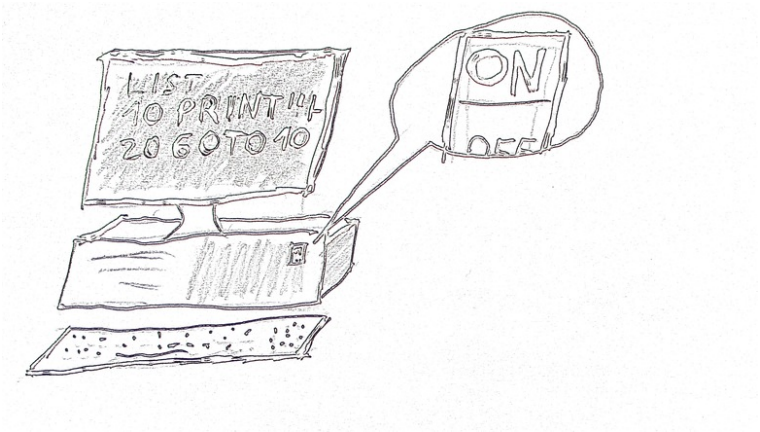

Abbildung 1: Ein erstes einfaches Programm

Inhaltsverzeichnis

1 Kompetenzen und Lernegebnisse

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

Sie können in der die für die C/C++-Programmierung üblichen Arbeitsumgebung Änderungen an einem einfachen Programm vornehmen.

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

- die zur Verfügung gestellte Linux-Programmierumgebung verwenden
- sich im Verzeichnisbaum mit der `bash`-shell bewegen
- Quelltexte editieren
- ein einfaches C++-Programm, welches aus einer Quelltextdatei besteht, erstellen
- dieses Programm modifizieren, so dass sich ein anderes Verhalten ergibt

2 Konzepte

2.1 Programmierumgebung

Installieren Sie VirtualBox auf ihrem Rechner und binden das zur Verfügung gestellt Image der virtuellen Maschine ein. Fahren Sie die Maschine hoch und melden sich als Nutzer `user` mit dem Passwort `user` an.

Starten Sie die `bash`-shell (bzw. ein `Terminal`) und erstellen ein neues Verzeichnis, in dem Sie dann arbeiten werden. Erstellen und editieren Sie die Quelldateien mit `pluma`. Achten Sie darauf, dass die Anzeige der Zeilennummern aktiviert ist.

2.2 Linux-System: Bedienung der Shell

Auf der Kommandozeile sind folgende Befehle hilfreich:

- Terminal starten: Rechtsklick auf Verzeichnis oder Desktop; dann “Terminal hier öffnen”
- anzeigen des aktuellen Verzeichnisses (working directory): `pwd`
- aktuelles Verzeichnis wechseln: `cd <name>`
- Textdatei editieren: `pluma <name>`
- Textdatei editieren: `pluma <name> &` (Terminal nicht blockiert)
- Programm im aktuellen Verzeichnis starten: `./<name>`
- Verzeichnis erzeugen: `mkdir <name>`

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

- Datei oder Verzeichnis verschieben/umbenennen: `mv <source> <target>`
- Datei kopieren: `cp <source> <target>`
- Verzeichnis kopieren: `cp -r <source> <target>`
- Datei(en) löschen: `rm <name> ... <name>`
- Unterschiede zwische Textdateien anzeigen: `diff <file1> <file2>` oder `meld <file1> <file2>`
- `.tar`-Datei in aktuelles Verzeichnis entpacken: `tar xf <name>`
- `.tar.gz`-Datei oder `.tgz`-Datei entpacken: `tar xzf <name>`
- Textdatei erzeugen: `touch <name.txt>`
- Verzeichnis löschen: `rmdir <name>`
- Programme installieren: `sudo apt install <name>` (z. B. `name = tree` oder `gedit`)
- Hex Dump einer Datei: `hexdump -vC <name>`
- die Datei `name` ausführbar machen: `chmod +x name`
- die Datei `name` starten: `./name`

2.3 Linux-System: Tastenkürzel

Die folgenden **besonders wichtigen** Tastenkürzel sind die Grundlage für flüssiges Arbeiten:

Kontext	Tastenkürzel	Wirkung
shell	CURSOR UP	last command
shell	TAB	completion
Terminal	CTRL-C	(kill process)
Editor	CTRL-S	save
Editor	CTRL-Q	quit
Editor	CTRL-C	copy selection to clipboard
Editor	CTRL-V	paste clipboard
Editor	SHIFT-ARROW	mark
Editor	POS1	cursor → beginning
Editor	END	cursor → end
pluma	CTRL-ALT-PAGEUP CTRL-ALT-PAGEDOWN	cycle tabs
Window Manager	ALT-TAB	cycle windows

Kontext	Tastenkürzel	Wirkung
Window Manager	ALT-SHIFT	switch keyboard layout US / German

Die folgenden Tastenkürzel erleichtern flüssiges Arbeiten (Level C):

Kontext	Tastenkürzel	Wirkung
Terminal	SHIFT-CTRL-C	copy
Terminal	SHIFT-CTRL-V	paste
shell	CTRL-A	go to beginning
shell	CTRL-E	go to end
Terminal	CTRL-Z	suspend process bg, fg afterwards
Editor	CTRL-Z	undo
Editor	CTRL-N	new window
Editor	CTRL-T	new tab
Editor	CTRL-W	close tab
Editor	CTRL-ARROW	jump word-wise

2.4 Übersetzen eines einfachen C-Programmes

Das nachfolgende Programm verfügt über drei Funktionen: `function()`, `printValues()` und `main()`. Letzere wird beim späteren Programmstart automatisch ausgeführt.

```
1  // this is "src-cpp-student/01_SMPL/main.cpp"
2
3  #include "println.hpp" // declaration/definition of println()
4
5  // declare functions; will be defined below
6  void printValues(int first, int last);
7  int function(int);
8
9  int main(){           // program entry point
10   printValues(0,9);
11   return 0;           // no error occoured
12 }
13
14 int function(int x){
15   int y = 2 * x - 4;
16   return y;
17 }
18
```

```
19 void printValues(int first, int last){
20   println("first=", first, "  last=", last);
21   int k=first;
22   while(k<=last){
23     int y=function(k);
24     println("y(",k,")=",y);
25     k = k+1;
26   }
27 }
```

Oben ist der Inhalt der Datei `src-cpp-student/01_SMPL/main.cpp` zu sehen. Diese Datei kann nun mit dem Kommando `clang++` zu einem ausführbaren Programm übersetzt werden. Da der Compiler einige Parameter benötigt, wird der Aufruf des Compilers in dem Shellscript `compile.sh` getätigt, worin die Parameter gesetzt werden:

```
bash$ ls
main.cpp
bash$ ./compile.sh main.cpp
bash$ ls
a.out  main.cpp
```

Anhand der beiden `ls`-Ausgaben ist zu sehen, dass die Datei `a.out` neu entstanden ist. Sie wurde von `clang++`- erzeugt und ist ausführbar. Das Programm wird mit der Eingabe von `./a.out` gestartet.

Im obigen Compilier-Beispiel sind viele Schritte verborgen, da `clang++` als Compiler Driver fungiert und mehrere Komponenten der Toolchain angetrieben hat. Diese Komponenten werden in einem der nächsten Materialpaketen vorgestellt.

An obigen Programm wird folgendes illustriert:

- `#include` fügt andere Dateien ein
- Funktionen, deren Parameter und Aufruf
- automatischer Aufruf von `main()`
- lokale Variablen
- Iteration mit `while`
- `println()` für einfache Ausgaben in C++ aus dem Header `println.hpp`
- ein C++-Programm besteht aus Deklarationen und Definitionen
- damit Funktionen aufgerufen werden können, müssen sie vorher (d.h. weiter oben im Quelltext) deklariert oder definiert werden
- ein Shell-Skript `compile.sh` enthält die Kommandos, die zur Erstellung des Programms notwendig sind
- sofern kein anderer Name angegeben wird, heißt das erstellte Programm `a.out`

3 Material zum aktiven Lernen

Regelmäßiger Hinweis: Da eine Programmiersprache nur durch aktive Verwendung erlernt werden kann, werden im Folgenden Aufgaben zum praktischen Üben vorgestellt. Zunächst wird ein Grundgerüst (C/C++Programm) erstellt, welches dann auf mehrere Arten modifiziert wird. Insbesondere die Modifikationen ermöglichen es dem Lernenden (und auch dem Lehrenden), die Qualität des Kompetenzerwerbs bzgl. dieses Materialpakets bewerten zu können.

3.1 Aufgabe: Grundgerüst

Erstellen Sie ein Programm, das aus einer Datei `main_simple.cpp` besteht. Darin wird eine Funktion deklariert und definiert, die diese Signatur hat:

```
1 int polynom1(int x); // returns y = f(x) for value x
2                      // f(x) is a polynominal function
3                      // e.g. y = (x-1)*(x-2)*(x-3)
```

In der `main()`-Funktion wird die Funktion `polynom1()` für die Werte `0..10` aufgerufen und dort die Werte mit `println()` ausgegeben.

Lassen Sie die Quelldatei mit `./compile.sh main_simple.cpp` oder `clang++ -I../90_aux_src/helpers -std=c++17 main_simple.cpp` übersetzen. Die Datei `a.out` ist das Ergebnis des Übersetzungsvorgangs.

Achten Sie darauf, dass Sie ihren Texteditor wie folgt konfiguriert haben:

- Anzeige der Zeilennummern aktiv
- vier Spaces statt Tabs
- ggf. Anzeige der Fläche, die die geöffneten Dateien anzeigt (Sidebar, File Browser pane)

3.2 Aufgabe: Modifikationen

Regelmäßiger Hinweis: Weiter unten ist eine Liste mit Modifikationen gegeben, die zwei Zwecken dienen: 1) Sie dienen als Richtschnur für das Praktizieren und Üben der Inhalte dieses Materialpakets. 2) Die Modifikationen können im Rahmen eines Testats als Aufgabe verwendet werden, durch deren Lösung Studierende nachweisen können, dass sie den Stoff dieses Materialpakets beherrschen. Stellen Sie sicher, dass Sie jede einzelne der nachfolgenden Modifikationen innerhalb weniger Minuten (ca. 5 - 10) vor Zuschauern (Testatsituation) umsetzen können. Konkret sollen Sie im Testat in der Lage sein, das gegebene Grundgerüst um mindestens eine zufällig ausgewählte Modifikation zu erweitern. Bereiten Sie dazu auf ihrer Arbeitsumgebung ein Verzeichnis vor, welches ausschließlich das Grundgerüst enthält. **Arbeiten Sie also auf einer Kopie des Verzeichnisses `src-cpp-student/`!** Achten Sie darauf, dass der Text auf Ihrem Bildschirm in heller Umgebung aus einem Meter Abstand heraus gut lesbar ist (light mode, große Schrift).

Hinweis: Im Testat sollen Sie die `bash` und `pluma` verwenden.

Modifikationen:

1. Provozieren Sie einen Syntaxfehler. Aktivieren Sie ggf. im Texteditor die Anzeige der Zeilennummern, um die vom Compiler gemeldete Codestelle finden zu können
2. Modifizieren Sie den Quellcode in ihrem Programm geringfügig, indem Sie eine Zeile verdoppeln und zeigen den kompletten Rundlauf editieren-compilieren-starten-editieren unter Einsatz der Tastenkürzel. Verwenden Sie nicht die Maus oder das Touchpad!
3. Verschieben Sie die Funktion `main()` nach oben, so dass es die erste Funktion im Quelltext ist. Verwenden Sie zum Markieren sowie Ausschneiden und Einsetzen die Tastatur. Lässt sich das Programm noch übersetzen?
4. Bauen Sie eine zweite C-Funktion `polynom2()` mit einer anderen Polynomfunktion ein und verwenden diese
5. Innerhalb der Shell: erstellen Sie ein Verzeichnis `mods01a/` (auf der selben Ebene wie `01_SIMPL/`; also mit `mkdir ../mods01a` im Arbeitsverzeichnis `01_SIMPL/`) kopieren alle benötigten Dateien dort hin und arbeiten auf dieser neuen Kopie weiter
6. Bauen Sie `println("1")` in `main()` ein. Unter ausschließlicher Benutzung der Tastatur: führen Sie alle zehn Durchläufe (roundtrip edit, compile, run) durch, um nacheinander die Zahlen bis `10` auszugeben. Verwenden Sie zehn einzelne Anweisungen; keine Iteration (Schleife)
7. Level C: Ersetzen Sie die `for`-Schleife durch `while`-Schleife (oder umgekehrt)

3.3 Verständnisfragen

Nach Bearbeitung des Kapitels "Konzepte", der Erstellung des Grundgerüsts sowie dem Üben der Modifikationen sollten Sie in der Lage sein, die folgenden Fragen zu beantworten.

1. Warum ist es wichtig, die Tastenkürzel zu beherrschen?
2. Warum wird eine Deklaration einer Funktion benötigt, die im Quelltext unterhalb einer Aufrufstelle definiert ist?
3. Freiwillig: Zeigen Sie auf, inwiefern das Piktogram auf der Titelseite dieses Materialpaketes den Inhalt zusammengefasst darstellt.

4 Nützliche Links

- Sektion 3 der Linux Manual Pages (z. B. `man 3 printf`)

- Clang Man Page online[2] oder per `man clang`
- Bjarne Stroustrup's FAQ http://www.stroustrup.com/bs_faq.html

5 Literatur

- [PPP] Stroustrup, Bjarne: Programming - Principles and Practice using C++
- [TCPL] Stroustrup, Bjarne: The C++ Programming Language, Fourth Edition

[2] http://clang.llvm.org/docs/CommandGuide/clang.html