Grundlagen wissenschaftlicher Programmierung

Sommersemester 2017

Technische Universität Berlin

1. Aufgabenblatt

Reichen Sie die Lösungen zu den Aufgaben 1 bis 3 bitte bis Montag, den **8. Mai 2017** um 23^{55} Uhr über ISIS ein. Als Abgabeformat für verwenden Sie bitte ein `zip`-komprimiertes `tar`-Archiv, in welchem Sie alle für die Lösung relevanten Dateien zusammenfassen (Aufgabe 3).

Erste Schritte

Bevor Sie mit der Lösung der ersten Aufgabe beginnen, machen Sie sich am besten zunächst mit der Linux-Installation im PC-Pool Physik oder auf Ihrem eigenen Rechner vertraut, indem Sie ein paar „alltägliche“ Befehle in der »Shell« ausführen. Öffnen Sie hierfür nach dem Login auf der graphischen Benutzeroberfläche ein Terminalfenster.

Alternativ können Sie sich natürlich auch per `ssh` mit einem geeigneten Rechner verbinden oder sich direkt an der Textkonsole eines Linux-Rechners anmelden.

- Nach dem Login oder dem Starten eines Terminalfensters befinden Sie sich normalerweise in Ihrem Heimatverzeichnis (*home*). Überprüfen Sie das, indem Sie sich den Namen des aktuellen Verzeichnisses inklusive Pfad mit dem Befehl `pwd` anzeigen lassen.
- Legen Sie mittels `mkdir` das Unterverzeichnis `GwP` für die Lehrveranstaltung an.
- Wechseln Sie mit `cd GwP` in dieses neue Verzeichnis und erzeugen Sie hier ein weiteres Unterverzeichnis namens `Uebung01`.
- Wenden Sie das Kommando `ls -Rl` im Verzeichnis `GwP` an und leiten Sie die Ausgabe des Befehls in eine Datei im Unterverzeichnis `Uebung01` um.
- Benutzen Sie ggf. `mv`, um die Dateien in die richtigen Verzeichnisse zu schieben.
- Mit `cd ~` kommen Sie immer wieder in Ihr Heimatverzeichnis zurück.

Die meisten Unix-Systeme verfügen über eine umfangreiche, aber mitunter gewöhnungsbedürftige Dokumentation: die sogenannten `man`-Pages, wobei `man` für *manual* steht. Die Eingabe von `man man` zeigt zum Beispiel die Dokumentation zum `man`-Befehl an.

Wenn Sie nicht wissen, was ein Befehl macht oder welche Optionen er bietet, ist die entsprechende `man`-Page eine gute Anlaufstelle. Informieren Sie sich über oben erwähnten `ls`-Befehl: Was bedeutet die Option `Rl`?

Technische Universität Berlin

Aufgabe 1

Programmieren Sie das sogenannte „Hallo,Welt!“-Programm, das beim Erlernen einer neuen Programmiersprache traditionellerweise als erste Übung implementiert wird.

- Erstellen Sie hierzu im Verzeichnis `Uebung01` (siehe »Erste Schritte«) eine Datei namens `hallowelt.cpp` mit folgendem Inhalt:

```
#include <iostream>

int main ()
{
  std::cout << "Hallo, Welt!" << std::endl;
  return 0;
}
```

- Kompilieren Sie die Datei mittels `c++ -Wall hallowelt.cpp -o hallowelt`.
- Führen Sie das Programm aus.
- Informieren Sie sich mittels `man` über das Programm `c++`. Lassen Sie sich anzeigen, welche Version von `c++` installiert ist und leiten Sie die Ausgabe in eine Datei namens `gcc-version` im Verzeichnis `Uebung01` um.

(3 Punkte)

Aufgabe 2

Schreiben (und kompilieren) Sie ein Programm, um den Speicherbedarf der fundamentalen Datentypen

- `float`
- `double`
- `long double`
- `char`
- `short`
- `int`
- `long int`
- `long long int`

auszugeben. Verwenden Sie hierfür den `sizeof`-Befehl. Definieren Sie außerdem eine Variable vom Typ `string` und ermitteln Sie dessen Größe:

```
string satz = "Dieser Satz kein Verb.";
cout << sizeof(satz) << endl;
```

Was können Sie beobachten? Kommentieren Sie das Ergebnis (ggf. mehrzeilig) im Quelltext.

(5 Punkte)

Technische Universität Berlin

Aufgabe 3

- Informieren Sie sich mittels `man tar` über den `tar`-Befehl.
- Erstellen Sie eine Textdatei mit den Antworten zu den folgenden Fragen und speichern Sie diese Textdatei ebenfalls im Verzeichnis `Uebung01` ab:
  1. Was bedeuten die Buchstaben `cvzf` im untenstehenden `tar`-Befehl?
  2. In welchem Verzeichnis müssen Sie sich befinden, damit die unten angegebene Befehlszeile das gewünschte Ergebnis liefert?
  3. Mit welchem Befehl bekommt man heraus, in welchem Verzeichnis man sich aktuell befindet?
- Fassen Sie den Inhalt des Verzeichnisses für die aktuelle Übung `Uebung01` mit dem Befehl `tar cvzf Uebung01-Name.tgz Uebung01` in einem zip-komprimierten `tar`-Archiv zusammen, wobei `Name` durch Ihren Nachnamen ersetzt wird.

(2 Punkte)

Aufgabe 4

Um die Zusammenarbeit mit anderen Programmierern zu erleichtern, existieren in jeder Programmiersprache ein paar allgemein anerkannte Regeln zur Standardisierung (Zeichenabstände, Einrückungen, Benennung der Variablen, etc.) des Quelltextes – so auch bei C++.

- Beachten Sie den Anhang zu den Regeln für C/C++-Quelltext.
- Wenden Sie die Richtlinien möglichst umfassend in jedem Ihrer Programme an.
- Überprüfen Sie Ihren Quelltext immer wieder anhand dieser Richtlinien, da es sonst zu Punktabzug kommt.
- Machen Sie es am besten von Anfang an „richtig“.

Hinweis

Das einzureichende Archiv `Uebung01-Name.tgz` sollte die folgenden Dateien enthalten:

- Eine Textdatei mit den Antworten zu Aufgabe 3
- eine Datei mit der Ausgabe des `ls`-Befehls
- die Datei `gcc-version`
- die Quelltextdatei `hallowelt.cpp`
- die ausführbare Datei `hallowelt`
- die Quelltextdatei `aufgabe02.cpp`
- die ausführbare Datei `aufgabe02`

Sie können den Inhalt mit `tar tzf Uebung01.tgz` überprüfen.

Technische Universität Berlin

Kleiner Styleguide für C++

Dieser sehr kurz gehaltene Styleguide enthält nur ein paar wesentliche Vorgaben, die für die Abgabe der Lösungen in C++ eingehalten werden sollten. Gewöhnen Sie sich diese elementaren Vorgaben bitte von Anfang an an.

Blockklammerung

- Die öffnende Klammer soll in der gleichen Zeile stehen, wie die Anweisung, die den Block einleitet.
- Die schließende Klammer muss so weit eingerückt sein wie der einleitende Befehl.
- Die Anweisungen in einem Block werden um zwei Zeichen eingerückt.

```
for (i = 0; i < 10; i++) {
  if ((i & 1) == 0) {
    j += i;
  }
}
```

- Die Klammern um einen Funktionsblock stehen in einer eigenen Zeile und sind so weit eingerückt wie der Funktionskopf.

```
int foobar (int x)
{
  return ++x;
}
```

Leerzeichen

- Die meisten binären Operatoren (+, <, >, ==, . . .) sollen durch Leerzeichen abgetrennt werden (siehe Beispiele zur Blockklammerung).
- Ausnahmen sind die Operatoren zur Memberauswahl, die Strukturname und Strukturelement miteinander verbinden sollen.

```
struktur.element;
strukturZeiger->element;
```

Namensgebung

- Es sollen aussagekräftige Bezeichner verwendet werden.
- Unterschiedliche Sprachen sollten nicht gemischt werden.
- Namen von Variablen und Funktionen sollen in Kleinbuchstaben geschrieben werden. Großbuchstaben dienen der Trennung von Wortbestandteilen.

```
int wichtigeZahl;
void liesDatei(string dateiname);
```

- Typnamen und Klassennamen sollen in Kleinbuchstaben geschrieben werden und mit einem Großbuchstaben beginnen.

```
typedef ULong usigned long;
class MyClass {
  ULong memberFunction();
};
```

- Konstanten sollen in Großbuchstaben geschrieben werden, wobei ein Unterstrich (_) die Namensbestandteile voneinander trennt.

```
#define MAX_COLORS 256
```

Deklarationen

- Alle Variablen sollen am Anfang des Blocks deklariert werden.
- Konstanten sollen nicht per `#define`-Anweisung angelegt werden (keine Typinformation), sondern als konstante Variable.
- Jede Variable wird in einer eigenen Deklaration angelegt.

```
int index;
int count;
```

- Bei der Deklaration von Zeigern wird * direkt an den Typnamen angehängt, entsprechend beim & für Referenzen.

```
char* ch;
void getValue(int& data) { ... }
```

- Größere Strukture sollten auch dann per Referenz an Prozeduren übergeben werden, wenn sie nicht verändert werden sollen. In diesem Fall sind sie zur Sicherheit als konstant zu deklarieren.

```
int printData (const bigStructure& data) { ... }
```

Kommentare

- Nichttriviale Stellen im Quelltext sollten kommentiert werden.
- Es sollten vor allem Zeilenkommentare (//) verwendet werden.
- Hinter Variablendeklarationen sollten Zeilenkommentare die Bedeutung der Variable erläutern.
- Vor jeder Funktion sollte ein Kommentar die Parameter und die Bedeutung der Funktion beschreiben.