

# 1. Übung des Programmierpraktikums

Abgabetermin: 18. März 2024, 23:59 Uhr

---

Die Übungen sind grundsätzlich selbst zu lösen.

Abzugeben sind jeweils die sinnvoll dokumentierten Programmfiles (\*.cpp, \*.h) in einem separaten Ordner dessen Name `bsp-nummer` zu sein hat, d.h., der Ordner `bsp_2` gehört zu Aufgabe 2.

---

1. Legen Sie in der IDE (Integrated Development Environment) `Code::Blocks` ein neues Projekt (Typ: **Console Application**) `bsp_1` mit dem C++-Quelltextfile `main.cpp` (oder `bsp_1.cpp`) an. Der Ordner `bsp_1` wird dabei automatisch mit diesem Projekt neu angelegt.

- Über die Menüeinträge  $\rightarrow$  „Build“  $\rightarrow$  „Build and run“ wird dieses Programm übersetzt und ausgeführt. Testen Sie dies. (1 Pkt.)
- Ersetzen Sie den gesamten Quelltext in `main.cpp` durch den Quelltext des Files `demoVector.cpp`<sup>1</sup>, übersetzen und führen Sie das Programm aus.
- Fügen Sie die Zeilen

```
cout << exp(abs(bb[3])) << endl;

char ch;
cout << "Eingabe eines Zeichens: "
cin >> ch;

const double c5 = c.at(5);
cout << "c[5] : " << c5 << endl;
```

vor der `return`-Anweisung in Ihr geändertes Quelltextfile ein.

- Übersetzen (Compilieren+Linken: *Build*) Sie Ihr Projekt. Lesen Sie die ersten Zeile des Compilerfehlers und korrigieren Sie den Syntaxfehler. Wiederholen Sie diesen Vorgang solange bis der Code fehlerfrei übersetzt wird.
  - Führen Sie das Programm aus (*Run*). Suchen und beheben Sie den Laufzeitfehler. Hinweis: In C/C++ beginnt die Indizierung von Vektoren etc. mit 0.
2. Setzen Sie einen Breakpoint in Zeile 17 des Quellfiles und nutzen Sie die Debugging<sup>2</sup>-Möglichkeiten der IDE, um das Programm schrittweise abzuarbeiten und die Veränderung der Variablen `aa`, `cc`, `i` im weiteren Programmverlauf zu beobachten.
2. Legen Sie ein neues (Consolen-)Projekt `bsp_2` an und schreiben Sie ein Programm welches den Widerstand  $R = \rho \cdot \ell / A$  eines Leiters aus dem spezifischen Widerstand  $\rho$  [ $\frac{\Omega \text{mm}^2}{\text{m}}$ ], der Länge  $\ell$  [ $\text{m}$ ] und dem Querschnitt  $A$  [ $\text{mm}^2$ ] des Leiters berechnet.

Wie nehmen Kupfer als Leitermaterial an, siehe [www](#)<sup>3</sup> für den konstant zu wählenden Wert  $\rho$ . Die Werte für  $\ell$  und  $A$  sind via `cin` auf `double`-Variablen einzulesen.

Geben Sie das Ergebnis  $R$  via `cout` im Terminal aus.

Testdaten ( $\ell, A$ ): (10, 2.5), (2.5, 10)

Achtung: Eine Gleitkommazahl heißt im englischen „floating point number“, daher muß der Dezimalpunkt statt des Kommas bei der Zahleneingabe angegeben werden.

Wissenschaftliche Schreibweise:  $17.86 \cdot 10^{-3}$   $\rightarrow$  Computer: `17.86e-3` oder `1.786e-2`

---

<sup>1</sup><http://imsc.uni-graz.at/haasegu/Lectures/Kurs-C/Beispiele/demoVector.cpp>

<sup>2</sup><http://imsc.uni-graz.at/haasegu/Lectures/Kurs-C/Download/kfu.html>

<sup>3</sup><https://www.xplore-dna.net/mod/page/view.php?id=1291>

3. Legen Sie ein neues Projekt `bsp_3` an, Sie können den Vorlesungscode (`intro_function`) benutzen.

- Schreiben Sie vor die Funktion `int main()` in `main.cpp` eine **Funktion** welche den Widerstand eines kreisförmigen Leiters analog zu Bsp. 2 aus dessen Länge  $[m]$  und Durchmesser  $d$   $[mm]$  berechnet.

**Input:**  $\rho, \ell, d$

**Output:**  $R$

Der berechnete Wert für  $R$  ist nicht in der Funktion, sondern nur im Hauptprogramm auszugeben. Die Testdaten können Sie im Hauptprogramm als Konstanten definieren, das erspart das wiederholte Eingeben.

*Testdaten* ( $\rho, \ell, d$ ): (1.786e-2, 10, 1.784), (27.8e-3, 1000, 17.841)

- Schreiben Sie eine zweite **Funktion** welche den Widerstand in Abhängigkeit von der Temperatur  $T$   $[^{\circ}C]$  für einen kreisförmigen Kupferleiter berechnet. Die Formel

$$R(T) = R_{T_0} \cdot (1 + \alpha \cdot (T - T_0))$$

und die Materialdaten für  $\alpha, \rho$  sind im `www`<sup>4</sup> zu finden,  $T_0$  wird mit  $20^{\circ}C$  angenommen.

**Input:**  $\ell, d, T$

**Output:**  $R$

Verwenden Sie als  $R_{T_0}$  die erste Funktion dieser Aufgabe, d.h. rufen Sie diese erste Funktion in Ihrer zweiten Funktion auf. Die Werte von  $\alpha, \rho$  können als Konstanten innerhalb dieser zweiten Funktion definiert werden.

Der berechnete Wert für  $R$  ist nicht in der Funktion, sondern nur im Hauptprogramm auszugeben.

*Testdaten* ( $\ell, d, T$ ): (10, 0.8, 40), (1000, 7.98, -20)

Achtung: **Keine Leerzeichen oder Umlaute** in Variablennamen oder Filennamen benutzen.

4. Berechnungen des arithmetischen, des geometrischen und des harmonischen Mittels:

- (a) Sie können mit dem Code (`intro_function`) beginnen und ein **Funktion** hinzufügen, welche (2 Pkt.)
- drei ganze Zahlen als Inputparameter über Inputparameterliste erhält,
  - die drei Mittel berechnet, und
  - die drei Werte über die Parameterliste an das aufrufende Programm zurückgibt.
- (b) Rufen Sie die Funktion von Ihrem Hauptprogramm auf und geben Sie dort die Inputgrößen und die Resultate aus.
- (c) Inputdaten (1, 4, 16) ergeben die drei Mittel (7, 4, 2.28571).
- (d) Inputdaten (2, 3, 5) ergeben die drei Mittel (3.33333, 3.10723, 2.90323).
- (e) Überprüfen Sie die Korrektheit für (1000, 4000, 16000), abgesehen von der beschränkten Genauigkeit der Gleitkommazahlen.
- (f) Die zweite **Funktion** besitzt dieselbe Funktionalität wie 4a) aber mit einem Vektor beliebiger Länge als Inputparameter statt der drei einzelnen Zahlen, siehe §2.3.3, §7.4 des Skriptes<sup>5</sup>. Generieren Sie sich einen Vektor mit mind. 10 Elementen (des Datentyps `int` oder `double`) und überprüfen Sie die Ergebnisse.

Hints: `#include <cmath>`, `pow`, `#include <vector>`, `vector`<sup>6</sup>, `#include <string>`, `string`

<sup>4</sup><https://www.xplore-dna.net/mod/page/view.php?id=1291>

<sup>5</sup>[https://imsc.uni-graz.at/haasegu/Lectures/Kurs-C/Script/html/script\\_programmieren.pdf](https://imsc.uni-graz.at/haasegu/Lectures/Kurs-C/Script/html/script_programmieren.pdf)

<sup>6</sup><https://en.cppreference.com/w/cpp/container/vector>

5. Überprüfen Sie, ob die folgenden Rechenregeln der Mathematik auch im Computer gelten. Die verfügbaren math. Funktionen sind unter `cmath`<sup>7</sup> zu finden und werden über `#include <cmath>` in Ihren Quelltext eingebunden,  $\pi$  wird von den meisten Compilern in diesem Headerfile als `M_PI` bereitgestellt. Lesen Sie dazu §3.6 des Skriptes<sup>8</sup> und auf [cppreference.com](http://cppreference.com)<sup>9</sup>.

$$\frac{(e^a)^b}{e^2} = e^{a \cdot b - 2}$$

$$\frac{b}{\sqrt{a+b} - \sqrt{a}} = \sqrt{a+b} + \sqrt{a}$$

$$\pi/2 - \arcsin a = \arctan \frac{\sqrt{1-a^2}}{a}$$

$$\cosh(\operatorname{arsinh}(b)) = \sqrt{b^2 + 1}$$

- Schreiben Sie ein oder zwei kleine Funktionen welche den absoluten und den relativen Fehler<sup>10</sup> zweier einfach genauer<sup>11</sup> Zahlen (`float`)  $a$  und  $b$  (Input-Parameter) als Output-Parameter zurückgeben.
- Testen Sie obige Rechenregeln indem Sie die rechte und die linke Seite der Gleichungen jeweils einer Variablen zuweisen und neben dem Wert auch absolute und relative Genauigkeit ausgeben.
- Geben Sie die Anzahl von Byte zur Speicherung in einer Variablen vom gewählten Datentyp unter Nutzung von `sizeof`<sup>12</sup> an und geben Sie die relative Genauigkeit (Maschinen- $\varepsilon$ <sup>13</sup>) für Ihren Datentyp analog zum Beispiel `Ex390.cpp`<sup>14</sup> an.

(1 Pkt.)

Hinweise: `exp`, `log`, `sqrt`, `pow`, `asin`, `atan`, `asinh`, `cosh`, `M_PI`, `sizeof`, `numeric_limits`

Testdaten (a,b): (0.9876, 1.762e-6) bzw. (0.678, 98),

Für Fortgeschrittene: Packen Sie obige Aufgabe in eine Template-Funktion welche Sie sowohl für `float` als auch für `double` und `long double` ausprobieren, siehe §10.1.1 im Skript<sup>15</sup> oder auf [Lern-Cpp.com](http://Lern-Cpp.com)<sup>16</sup>.

<sup>7</sup><https://en.cppreference.com/w/cpp/header/cmath>

<sup>8</sup>[http://imsc.uni-graz.at/haasegu/Lectures/Kurs-C/Script/html/script\\_programmieren.pdf](http://imsc.uni-graz.at/haasegu/Lectures/Kurs-C/Script/html/script_programmieren.pdf)

<sup>9</sup><https://en.cppreference.com/w/cpp/header/cmath>

<sup>10</sup>[https://de.wikipedia.org/wiki/Fehlerschranke#Absoluter\\_Fehler](https://de.wikipedia.org/wiki/Fehlerschranke#Absoluter_Fehler)

<sup>11</sup>[https://de.wikipedia.org/wiki/Einfache\\_Genauigkeit](https://de.wikipedia.org/wiki/Einfache_Genauigkeit)

<sup>12</sup><http://imsc.uni-graz.at/haasegu/Lectures/Kurs-C/Beispiele/DataTypes.cpp>

<sup>13</sup><https://de.wikipedia.org/wiki/Maschinengenauigkeit>

<sup>14</sup><http://imsc.uni-graz.at/haasegu/Lectures/Kurs-C/Beispiele/Ex390.cpp>

<sup>15</sup>[https://imsc.uni-graz.at/haasegu/Lectures/Kurs-C/Script/html/script\\_programmieren.pdf](https://imsc.uni-graz.at/haasegu/Lectures/Kurs-C/Script/html/script_programmieren.pdf)

<sup>16</sup><http://www.learncpp.com/cpp-tutorial/131-function-templates/>

6. Berechnen Sie die folgenden Ausdrücke für gegebene komplexe Zahlen  $a$ , und  $b$ . Die verfügbaren math. Funktionen sind unter *complex*<sup>17</sup> zu finden und werden über `#include <complex>` wie in §2.3.2 des Skriptes<sup>18</sup> in Ihren Quelltext eingebunden. (1 Pkt.)

Benutzen Sie doppelt genaue komplexe Zahlen, sodaß  $-3.5 + 2i \implies \text{complex<double> } b(-3.5, 2)$ . Geben Sie für die gegebenen Zahlen  $a$  und  $b$  die Polarkoordinaten, d.h., den Winkel  $\varphi$  (`arg()`) und den Betrag  $r$  (`abs()`), in der Gaußschen Zahlenebene aus.

$$\begin{aligned} e^{\pi \cdot a} &= \\ a^3 &= \\ \sqrt{b} &= \\ b + \bar{b} &= \\ e^{i\varphi_b} - (\cos \varphi_b + i \cdot \sin \varphi_b) &= \\ 3 \cdot a - b &= \end{aligned}$$

*Testdaten* (a,b):  $(0 + i, -3.5 + 2 \cdot i)$ , bzw.  $(0 + \frac{2}{3} \cdot i, -3 - 4 \cdot i)$ ,

Hinweise: `pow`, `log`, `log10`, `acos`, `atan`, `exp`, `sqrt`, `arg`, `abs`, `conj`

Die Abgabe der Lösungen (`*.cpp`-Files, `*.h`, (`Makefile*`), ...) erfolgt über UNI Graz-Moodle, siehe dazu die Hinweise auf der LV-Homepage<sup>19</sup>.

Die Verzeichnisnamen mit den Files für die jeweilige Aufgabe **müssen** dem Schema `bsp_nummer` folgen, z.B. enthält das Verzeichnis (der Ordner) `bsp_1` alle Files für Beispiel 1. Durch die automatische Abgabekontrolle zählen andere Verzeichnisnamen als nicht abgegeben.

Keine Leerzeichen, Sonderzeichen oder Umlaute in File- und Verzeichnisnamen benutzen (Portabilität).

<sup>17</sup><https://en.cppreference.com/w/cpp/numeric/complex>

<sup>18</sup>[http://imsc.uni-graz.at/haasegu/Lectures/Kurs-C/Script/html/script\\_programmieren.pdf](http://imsc.uni-graz.at/haasegu/Lectures/Kurs-C/Script/html/script_programmieren.pdf)

<sup>19</sup><http://imsc.uni-graz.at/haasegu/Lectures/Kurs-C/Download/kfu.html>