Aufgabenblatt 1

Themen

- Einarbeitung in die Programmiersprache C++
- Debuggen von Programmcode
- Erstellen und Verwenden einer einfachen Klasse

Antestat

1. **Debuggen mit IDE**
   Übertragen Sie den Code aus Listing 1. Setzen Sie einen Haltepunkt in die Zeile 5. Starten Sie den Debugger. Das Programm wird an der Stelle unterbrochen, bevor die Zeile ausgeführt wird. Verändern Sie den Wert der Variable a auf 12 und führen Sie das Programm zu Ende aus. Die Ausgabe sollte lauten:

   ```
   42
   12
   ```

   Erklären Sie die Ausgabe.[1]

```
1  #include <iostream>
2  int main(){
3    int a {42};
4    std::cout << a << std::endl;
5    std::cout << a << std::endl;
6  }
```

Listing 1: Debuggen üben

2. **Kompilieren & Ausführen mit der Konsole**
   Zeigen Sie, dass Sie das Startprojekt ohne IDE mit Hilfe der Konsole kompilieren und ausführen können.

[1] Verwenden Sie dabei die Begriffe *Laufzeit*, *Editierzeit* und *Speicher*.

Aufgaben

1. **Klasse**
   - Skizzieren Sie auf „Papier“ eine Klasse `Widerstand`, die einen elektrischen Widerstand repräsentiert.
   - *Eigenschaften*:
     - `widerstandswert` (Ohm)
     - `name` (String)
   - *Methoden*:
     - `ausgabe()`: Gibt die Eigenschaften formatiert auf der Konsole aus.
2. **Implementierung**
   - Implementieren Sie die Klasse `Widerstand` in Ihrer `main.cpp`.
   - *Optional*: Lagern Sie den Code für die Klasse Widerstand in eine separate Datei aus und binden Sie diese mittels einer *Header-Datei* in die `main.cpp` ein.
3. **Vektorausgabe**
   - Implementieren Sie eine Funktion, die einen `std::vector` der Standardbibliothek vom Typ `Widerstand` als Parameter aufnimmt und jedes Element des Vektors auf der Konsole ausgibt.
   - Verwenden Sie hierfür die `ausgabe()`-Methode der Klasse `Widerstand`.
4. **Gesamtwiderstand berechnen**
   - Implementieren Sie die Methode `double get_widerstand_reihe(const std::vector<Widerstand>& widerstaende)` in `main.cpp`, die den Gesamtwiderstand bei Reihenschaltung der Widerstände berechnet und zurückgibt:

$$R_{\text{ges}} = R_1 + R_2 + R_3 + \dots$$

5. **Testfälle**
   - Entwickeln Sie in der `main()`-Methode verschiedene Testfälle, um die Funktionalität der Klasse und Methoden zu überprüfen. Siehe Beispiel in Listing 2.

```
Widerstand w1 {"R1", 80};
Widerstand w2 {"R2", 120};
Widerstand w3 {"R3", 250};
vector<Widerstand> w_all{};
w_all.push_back(w1);
w_all.push_back(w2);
w_all.push_back(w3);
print_widerstaende(w_all);
std::cout << "Widerstand in Reihe - Erwartet: 450, Ergebnis: " << get_widerstand_reihe(w_all) << std::endl;
```

Listing 2: Beispiel für einen Testfall

Abtestat

- Zeigen Sie, dass Sie Ihr Programm *kompilieren* und ausführen können. Wenn Sie eine IDE verwenden, zeigen Sie, mit welchen Argumenten compiliert und ausgeführt wird und wie Sie dies konfigurieren können.
- Erläutern Sie Ihre Modellierung.
- Erläutern Sie Ihre Testfälle.

Hinweise zu std::vector

- `std::vector<Typ>`[2] ist ein sogenannter Daten-Container aus der C++ Standardbibliothek.
- Die Einbindung erfolgt über `#include<vector>`
- Ein Vektor kann genauso genutzt werden wie ein Daten-Feld, schützt aber vor Speicherüberschreitungen und verfügt über viele nützliche Funktionen.
- Listing 4 zeigt einige Beispiele zum Umgang.

```
std::vector<double> vec1; // Deklariert einen leeren Vektor-Container fuer double Elemente
std::vector<int> vec2 {1, -2}; // Ein Vektor-Container laesst sich wie ein Feld initialisieren
std::cout << vec2[1] << std::endl; // Zugriff auf das zweite Element in vec2
// Ausgabe: -2
std::cout << vec2.size() << std::endl; // Gibt die Laenge des Vektors zurueck
// Ausgabe: 2
vec2.push_back(10); // fuegt ein weiteres Element mit dem Wert 10 hinzu
```

Listing 3: Beispiel-Code std::vector

Benötigte Ausstattung

- Kommandozeile (MinGW/Shell/Terminal)
- GCC-Compiler mit C++23[3]
- C++ Startprojekt `https://git.fh-muenster.de/vclab/cpp/cpp23_test` auf dem GitLab-Server[4] der FH Münster
- IDE, z. B. CLion, Visual Studio Code
- (optional) Anwendung zum Zeichnen von Diagrammen, z. B. draw.io[5], Mermaid[6]

Vorbereitung

- Richten Sie Ihre C++ Entwicklungsumgebung ein. Ein Ausgangspunkt kann unser Startprojekt zur Veranstaltung sein.
- Testen Sie, mit Hilfe der gegebenen `main.cpp`, ob Ihre Umgebung C++23 unterstützt. Folgen Sie dabei den Anweisungen in der gegebenen `README.md`.
- Lesen Sie die Aufgaben sorgfältig.
- Finden Sie eine/n Gruppenpartner/in.
- Bereiten Sie sich auf das Antestat vor.
- Besprechen Sie in Ihrer Gruppe die Aufgaben und überlegen Sie sich geeignete Testfälle.

Hinweis: Nutzen Sie das Tutorium für die Vorbereitung des Praktikums. Sie können dort Hilfestellungen zu allen Themen bekommen.

[2] `https://en.cppreference.com/w/cpp/container/vector`
[3] `https://gcc.gnu.org/`
[4] `https://git.fh-muenster.de/`
[5] `https://app.diagrams.net/`
[6] `https://mermaid.js.org/syntax/classDiagram.html`