Praktikumstermin Woche 49 (WiSe 2022)

(Praktika sind Pflichtleistungen, die bis zum genannten Termin erbracht werden müssen. Bei Verhinderung durch Krankheit ist eine ärztliche Bescheinigung der Arbeitsunfähigkeit vorzulegen!)

Name, Vorname, Matrikelnummer Unterschrift[1] Testat

Teilnehmer:

.....

Aufgabenstellung: Programmierung von Array-Operationen mit Funktionen unter Verwendung von Dateien

Das Programm soll mit Datensätzen unterschiedlicher Widerstände umgehen können, die über die Konsole eingegeben werden. **Unterteilen Sie** Ihr Programm in die Dateien main.cpp, widerstand.h und widerstand.cpp!

Die zentrale Datenstruktur in Ihrem Programm ist ein **dynamisches *Array*** *(`dataArray`) vom* global in der Header-Datei („`widerstand.h`“) definierten Datentyp `widerstand`:

```
struct widerstand
{
   int dim;              //mΩ --> 0, Ω --> 1, kΩ --> 2, MΩ -->3
   string design;        //Kohleschicht, Metalloxid etc.
   double value;         //positive Wert größer Null
};
```

Die Funktionsprototypen lauten wie folgt:

```
/*    Funktionskommentar (s. Hinweise) */
void printData(vector<widerstand> dataArray);
/*    Funktionskommentar (s. Hinweise) */
bool writeToFile(vector<widerstand> dataArray);
```

Das dynamische Array soll über den Template-Typ **vector<>** in der `main()`-Funktion angelegt werden:

```
vector<widerstand> dataArray(0);
```

Das Programm ist als **Konsolen-Anwendung** mit der IDE Visual Studio C++ zu realisieren.

Funktion **`main()`** (Steuerzentrale des Programms)

Die Main-Funktion ist wiederum die Steuerzentrale des Programms. Um sich auf die Funktionen konzentrieren zu können, entnehmen Sie Teile des Quellcode der main()-Funktion mit ‚*copy&paste*‘ dem Anhang dieser Aufgabe.

Ergänzen Sie die Main-Funktion an den Stellen, wo Sie den Kommentar „/*TODO*/“ finden.

Stellen Sie fest, ob der Benutzer vor Beendigung des Programms neu eingegebene Datensätze schon gespeichert hat und **warnen Sie** ihn vor dem möglichen Datenverlust (Einsatz einer booleschen Variablen)!

Funktion **`printData`**`(vector<widerstand> dataArray);`

Die Ausgabefunktion gibt die Bauform, den Widerstandswert und die Einheit für alle in der Feldvariablen **`dataArray`** gespeicherten Widerstände aus. In dieser einfachen Funktion führen Sie die folgenden Schritte aus:

- **Ermitteln Sie** die Feldgröße bspw. mit der eingebauten Elementfunktion **`size()`**:

  ```
  if(dataArray.size() > 0) …
  ```

- **Deklarieren Sie** die Variable **`dim`** vom Datentyp `string` !

[1] Dieses Deckblatt ist zum Abgabetermin ***ausgefüllt*** (!) mitzubringen.

- **Weisen Sie** dieser Variablen entsprechend der Widerstandsdimension in einer Switch-Anweisung innerhalb einer Zählschleife die korrekte Maßeinheit (Milli-Ohm, Ohm, Kilo-Ohm, Mega-Ohm) **zu**.
- **Geben Sie** die Datensätze, wie in der Abbildung dargestellt, **formatiert aus** (z.B. drei Nachkommastellen beim Widerstandswert)!

```
switch( /*...TODO*/ )
{
case 1: dim = "Milli-Ohm"; break;
case 2: dim = "Ohm"; break;
case 3: dim = "Kilo-Ohm"; break;
case 4: dim = "Mega-Ohm"; break;
}
```

Funktion **`writeToFile`**`(vector<widerstand> dataArray);`

Mit dieser Funktion werden die Elemente des Struktur-Arrays in einer Textdatei gespeichert. **Führen Sie** die folgenden Schritte **aus**:

- **Deklarieren Sie** die Datei-Variable (file handle) `ziel` vom Typ **ofstream**!
- **Öffnen Sie** die Datei zum Schreiben in die Datei → Elementfunktion: **`open()`**
- **Schreiben Sie** die Widerstands-Datensätze mit dem Output-Stream-Operator (**<<**) in die Datei, wenn die Datei zum Schreiben geöffnet werden konnte (**`is_open()`**)!
- **Trennen Sie** jeweils die einzelnen Strukturelemente voneinander, indem Sie mit der eingebauten Funktion `ziel.width()` oder mit setw() die Ausgabebreite der Daten steuern!
- **Schließen Sie** die Datei, wenn alle Datensätze geschrieben wurden!

Quellcodebeispiele des Speicherns in eine Textdatei und des Lesens aus einer Textdatei entnehmen Sie der Vorlesung, dem Seminar oder meinem Textskript!

Überprüfen Sie stets, ob die Datei zum Lesen bzw. Schreiben geöffnet werden konnte! Erzeugen Sie ggf. entsprechende Fehlermeldungen!
Kommentare im Quellcode (Mindestanforderung)

- Kommentarkopf der CPP- und der Header-Datei
- Kommentarkopf eines Funktionsprototyps in der Header-Datei

```
/*
 <Kurzbeschreibung der Funktion>
 @param1: <Was/Beschreibung>,<Datentyp>,<Parametertyp (c.b.r/v.)>
 @param2: …
  …
 Rückgabewert: <Was/Beschreibung>, <Datentyp>
*/
```

```
Kein Daten vorhanden!


Was wollen Sie tun?
Neuen Widerstand eingeben          (n/N)
Widerstands-Datensaetze ausgeben   (a/A)
Datensaetze suchen                 (s/S)
Datensaetze speichern              (m/M)
Programm beenden                   (b/B)


?
```

```
Eingabe eines neuen Datensatzes.
Widerstands-Bauform (Kohleschicht, Metallschicht, LDR etc.)
: Kohleschicht

Widerstands-Wert (Dezimaltrennzeichen ist der Punkt!): 4.7

Widerstandsdimension:
        MiliOhm --> 1
        Ohm     --> 2
        KiloOhm --> 3
        MegaOhm --> 4

        --> 3


        Widerstands-Bauform     Wid.-Wert      Einheit

               Kohleschicht         4.700      Kilo-Ohm



Was wollen Sie tun?
Neuen Widerstand eingeben         (n/N)
Widerstands-Datensaetze ausgeben  (a/A)
Datensaetze suchen                (s/S)
Datensaetze speichern             (m/M)
Programm beenden                  (b/B)
```

Abb. 1: Ausgaben des zu realisierenden Konsolen-Programms

Obligatorische Zusatzaufgabe (für „Nachholer“ und zum Selbststudium):

- **Implementieren Sie** eine Funktion **`suchen()`**, die über den **Algorithmus der linearen Suche** alle Widerstände eines bestimmten Typs sucht und auflistet!
- **Implementieren Sie** eine Funktion **`sortiere()`**, die die Datensätze nach dem Bauelementtyp sortiert (BubbleSort-Algorithmus)!

Informieren Sie sich selbständig, welche Bibliotheksfunktionen Sie über die `Include`-Anweisung in Ihr Programm einbinden müssen!

Hinweis:

Erstellen Sie im Vorfeld des Praktikums jeweils ein Quellcode-nahes Struktogramm (nach Symbolvorgaben von Nassi/Shneiderman !!!) für jede Funktion (außer main()), das den Lösungsweg für dieses Modul vorzeichnet!

Das Programm (C++-Projekt) wird während des Praktikums zum angegebenen Zeitpunkt im Informatik-Labor U340 abgenommen. Dazu erstellen Sie das Programm allein (!) auf der Basis des Struktogramms während des Praktikums. Vorlesungs- oder Seminarunterlagen können Sie verwenden!

Das Struktogramm soll den prinzipiellen Ablauf des Programms (Algorithmus) enthalten. Die angegebene Datenstruktur ist außerhalb des Struktogramms zu definieren!

Das Gerüst der Main-Funktion, die Prototypen und die Struktur-Definition können Sie den beigefügten Dateien entnehmen!

```
int main(void)
{
  //TODO…
  widerstand ds;      //ein Widerstandsdatensatz zum späteren Speichern in dataArray
  char chr, puffer[10];
  bool exit = false;


  while(true)
  {
        cout << "\n\n Was wollen Sie tun?";
        cout << "\n Neuen Widerstand eingeben        (n/N)";
        cout << "\n Widerstands-Datensaetze ausgeben (a/A)";
        cout << "\n Datensaetze speichern            (s/S)";
        cout << "\n Programm beenden                 (b/B)" << endl;
        cout << "\n\n ? ";
        cin >> chr;

        switch (tolower(chr))
        {
        case 'n':
               system("cls");

               /*TODO*/
               cout << "\n\n Eingabe eines neuen Datensatzes. ";
               //Widerstands-Bauform
               cout << "\n Widerstands-Bauform: ";
               cin >> /*TODO*/

               //Widerstands-Wert
               do
               {
                      cout << "\n Widerstands-Wert ";
                      cout << "(Dezimaltrennzeichen ist der Punkt!): ";
                      cin >> puffer;
                      /*TODO: support.h inkludieren und Funktion convertToDouble() verwenden*/
               }
               while( /*TODO*/ );

               //Dimension des Widerstandswertes
               do
               {
                      cout << "\n Widerstandsdimension: ";
                      cout << "\n\t MiliOhm --> 1";
                      cout << "\n\t Ohm     --> 2";
                      cout << "\n\t KiloOhm --> 3";
                      cout << "\n\t MegaOhm --> 4";
                      cout << "\n\n\t --> ";
                      cin >> puffer;
                      /*TODO: ggf. eigene support.h inkludieren und Funktion convertToInt()
                      verwenden*/
               } while ( /*TODO*/);
               /*TODO*/

               if( /*TODO*/) ausgabe(dataArrayFeld);
        break;

        case 'a':
               /*TODO*/
        break;
        case 's':
               system("cls");
               /*TODO*/
        break;
        case 'b': exit = true;
                  break;
        }//switch
        if(exit) break;
  }//while
  return EXIT_SUCCESS;
}//main
```