Aufgaben

1. Gehen Sie die einzelnen Beispiele der Folien durch und probieren Sie das eine oder andere Programm in Visual Studio selbst aus. Setzen Sie Break Points, starten Sie die Programme und werfen Sie während der Laufzeit jeweils einen Blick auf den Typ und den Inhalt der Variablen.

   Experimentieren Sie etwas mit den Programmen, Visual Studio und dem Debugger.
   Versuchen Sie sich zu erklären, warum etwas so ist, wie es ist.
   Lesen Sie dazu jeweils auch in den angegebenen Quellen nach (färbiger Kasten jeweils rechts auf jeder Folie), um Ihre beim Testen gemachten Erfahrungen auch erklären zu können.
   Zudem bieten die Quellen weit mehr Anwendungsbeispiele und Informationen, als wir im Unterricht aus zeitlichen Gründen durchmachen könnten.

2. Erstellen Sie unter Visual Studio für jedes Aufgabenblatt (jede Abgabe) eine eigene Solution, in die Sie jeweils die Programm (Projekte) einfügen. Die Solution sollte die Bezeichnung für
   a. Einzelabgaben: OOP18_<Gruppe>_<AufgabenNr>_<Nachname>_<Vorname>
      zB: OOP19_01_Mustermann_Max
   b. Teamabgaben: OOP18_<Gruppe>_[AufgabenNr]_[Nachname1]_[Nachname2]
      zB: OOP19_07_Nordmann_Doe

   Die Namen der Projekte werden bei den einzelnen Unteraufgaben jeweils angegeben.

3. Programm **D1Array**

   a. Schreiben Sie ein Programm, das ein `std::array` für double-Werte anlegt. Die Elemente des Array sollten bei der Initialisierung jeweils mit 0-Werten initialisiert werden.

   b. Ergänzen Sie Ihr Programm aus a) und befüllen nun das Array aufsteigend in einer einfachen for-Schleife mit Zugriff auf die Elemente über den Index mit Werten beginnend ab 0, jeweils um die Schrittweite 0.3456789 erhöht.

   c. Ergänzen Sie Ihr Programm aus b) um eine Funktion **`coutArray`**, welche das Array entgegen nimmt und 4-spaltig auf `cout` ausgibt.

4. Programm **D1Vector**

   a. Schreiben Sie ein Programm, welches die dem Programm übergebenen Argumente (arg, argv) in einen `std::vector` überträgt.

   b. Ergänzen Sie Ihr Programm aus a) um eine Funktion, welche den Vektor und einen Dateinamen entgegennimmt. Die Funktion soll eine Textdatei erstellen, die den

Inhalt des Vektors wie folgt abbildet, wenn das Programm wie folgt aufgerufen wird:

```
D1Vector.exe Max Mustermann 1234
```

Die Ausgabe des Programms in die Textdatei wäre wie folgt:

```
Programmname=D1Vector.exe
Argumente=3
Argument_1=Max
Argument_2=Mustermann
Argument_3=1234
```

5. Programm **D2VectorDet**

   a. Schreiben Sie ein Programm, das eine Textdatei mit einer 2-dim Matrix der Größe 3x3 vom Typ `double` in einen 2-dimensionalen `std::vector` liest. Die Datei hat das einfache Format:

      a11, a12, a13
      a21, a22, a23
      a31, a32, a33

      Der Dateiname soll dem Programm als Argument übergeben. Etwaige Fehlermeldungen sollen auf cerr ausgegeben werden.

   b. Ergänzen Sie Ihr Programm aus a) um eine Funktion, welche die Determinante für den 2-dim.Vektor errechnet und als Ergebnis der Funktion zurückgibt (siehe https://de.wikipedia.org/wiki/Regel_von_Sarrus).

   c. Ergänzen Sie Ihr Programm aus b) um eine Ausgabefunktion, in der die Matrix als auch der Wert der Determinante ausgegeben wird. Die Ausgabe soll auf `cout` erfolgen und wie folgt aussehen:

      [a11, a12, a13]
      det [a21, a22, a23] = wert
      [a31, a32, a33]

Abgabe

- Klicken Sie unter Visual Studio mit der **rechten Maustaste im Solution Explorer** auf den Solutionname (erste Zeile). Wählen Sie den Menüpunkt **Clean Solution** aus. Damit sollen alle Compilate und temporären Dateien gelöscht sein.

- Die Abgabe erfolgt im Moodle unter Assignment 01
  Beachten Sie, dass es für jede Gruppe einen bestimmten Upload Link gibt.
- bis spätesten **vor Beginn der nächsten Laboreinheit**

Kontrollfragen

Die Kontrollfragen sind nicht abzugeben. Sie sollten aber nach der Wiederholung der Laborfolien und nach obigen Aufgaben/Übungen die Fragen beantworten können. Jeweils am Beginn der nächsten Laboreinheit, werden einzelne Studierende einige dieser Fragen zur Wiederholung gestellt bekommen (= LV-begleitende Beurteilung).

1. In welcher Codierung werden unter C++ Zeichenketten in std::strings gespeichert?
2. Worin unterscheiden sich die einzelnen Codierungen?

3. Welches Schlüsselwort muss für einen Typ verwendet werden, wenn der Datentyp automatisch gewählt werden soll?
4. Wann und auf Basis welcher Einflussfaktoren wird der Datentyp dabei gewählt: während der Runtime oder während der Compile time?

5. Mit welchem Operator kann unter C++ Speicher auf dem Heap allokiert werden?
   Und mit welchem Operator kann dieser Speicher wieder freigegeben werden?

6. Welche Vorteile bietet std::array<..> im Vergleich zu klassischen C-style Arrays?
   Welche Vorteile bietet std::vector<..> im Vergleich zu std::array<..>?

7. In welchem Speicher liegen die Daten von c-style Arrays, std::arrays und std::vector?

8. Wozu dienen Referenzen?
   Wie können Referenzen deklariert werden?

9. Welche Streams stehen unter C++ zum einfachen Lesen und Schreiben von (Text-)Dateien zur Verfügung?