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
   Teamabgaben: OOP21_<Gruppe>_[AufgabenNr]_[Nachname1]_[Nachname2]
   zB: OOP21_A_05_Musterfrau.Petra

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

3. Schreiben Sie eine Klasse **TData** mit den Attributen **name (String)** und **data (char-Vector)**. Um Sicherheit bei den Aussagen zu gewährleisten können Sie Konstruktoren (Normal, Copy, Move) verwenden.

   a. Legen Sie eine **Variable a** vom Typ **TData** an. Weisen Sie einer weiteren neuen **Variable b** vom Typ **TData** die **Variable a** zu.
      - Wird bei der Zuweisung der Copy- oder der Move Konstruktor aufgerufen? **(1)**

   b. Schreiben Sie eine Methode, die **einen Parameter** vom Typ **TData** entgegennimmt. Rufen Sie diese Methode auf.
      - Wird bei der Übergabe des Parameters der Copy- oder der Move Konstruktor aufgerufen? **(1)**

   c. Schreiben Sie eine Methode, die **keinen Parameter** entgegennimmt. Sie legt aber eine **lokale Variable** vom **Typ TData** an, befüllt diese mit Daten und gibt dieses Objekt dann über **return** zurück. Rufen Sie die Methode auf und weisen das Ergebnis einer neuen **TData**-Variable **c** zu.
      - Wird bei der Übergabe durch **return** der Copy- oder der Move Konstruktor aufgerufen? **(1)**

d. **Unterdrücken** Sie nun jeweils die Erzeugung eines Default-Copy oder Move Constructors und testen Sie obige Methode.
   - Hat sich am Verhalten etwas geändert? **(1)**

e. Wenn ein **Move-Constructor** aufgerufen wird,…
   - Wie finden Sie nach dem Move die Datenelemente des Quellobjekts vor?
   - Wird trotzdem ein Destruktor aufgerufen? **(1)**

f. Füllen Sie mehrere **TData**-Elemente in einen **vector<…>** und ein **array<…>** und iterieren Sie durch diese Collections. Klären Sie die Fragen:
   - Werden bei deren Verwendung auch Move und Copy-Operatoren aufgerufen?
   - Was passiert, wenn Sie Move- oder Copy-Konstruktoren ausdrücklich unterbinden oder wechselseitig forcieren?
   Können diese Collections trotzdem verwendet werden? Kommentieren Sie nicht funktionierenden Code aus. **(1)**

g. Schreiben Sie nun mindestens einen **Konstruktor** und **Destruktor**.
   - Hat sich dadurch an den Antworten der obigen Fragen etwas geändert? **(1)**

h. Schreiben Sie eine eigenen **Copy-Constructor**, der das Objekt **in die Tiefe** kopiert. (Antwort als Code Snippet) **(1)**

i. Schreiben Sie einen eigenen **Move-Constructor**. (Antwort als Code Snippet) **(1)**

j. Wie kann bei einer Zuweisung die Anwendung eines **Move-Constructors erzwungen** werden? (Antwort als Code Snippet) **(1)**

k. Fügen Sie Ihrer Klasse TData einen **static int id** hinzu und stellen Sie einen **getter** und **setter** für dieses Attribut zur Verfügung.
   - Müssen Sie nun auch den Code ihrer Copy- und Move-Constructor anpassen? **(1 Bonuspunkt)**

Fügen Sie die Antworten all auf obige Fragen (und die etwaige Begründung dazu) unter Angabe der Nummerierung in Textform als **PDF** Ihrer Abgabe bei. Aufgabe bis spätestens vor der nächsten Laboreinheit bei Moodle Assignment 5. Code muss nicht abgegeben werden.