C/C++ Programmierung – Aufgabenblatt „Zeiger“

Geschätzter Bearbeitungsaufwand außerhalb der Übungsstunden: 45 Minuten.

1. Destruktor von Matrix

Erweitern Sie die Klasse `Matrix` um einen Destruktor, der „Matrix (…) wird zerstört“ ausgibt, wobei für „…“ die Elemente hintereinander angezeigt werden sollen (Methode `ausgabe()`).

- Wie viele Ausgaben dieser Art erwarten Sie, wenn Sie das Hauptprogramm der vorigen Abgabe starten? Wie viele sind es tatsächlich und wann erfolgen sie?
  Bitte schreiben Sie die Antworten zu diesen Fragen in die Datei mit Ihrem Hauptprogramm.

2. Vektor2D-Objekte auf dem Heap

Erweitern Sie nun das Hauptprogramm aus dem Aufgabenblatt „Klassen 1“:

- Legen Sie *auf dem Heap* zwei weitere Vektoren vom Typ `Vektor2D` an: `z` und `w`. Initialisieren Sie `z` mit (-3,1), während für `w` der Standard-Konstruktor verwendet werden soll.
- Addieren Sie am Ende des Hauptprogramms zu `a` (siehe Aufgabenblatt „Klassen 1“) den Vektor `z` durch Aufruf der `addiere`-Methode und lassen Sie sich das Ergebnis (also `a`) ausgeben.
- Addieren Sie nun zu `w` den Vektor `z` und lassen Sie sich das Ergebnis (also `w`) ausgeben.
- Addieren Sie nun zu `w` den Vektor `b` und lassen Sie sich das Ergebnis (also `w`) ausgeben.
- Überprüfen Sie anhand der Ausgabe des Destruktors von Matrix (siehe die Aufgabe 1 oben), ob auch alle Ihre angelegten Vektoren wieder freigegeben werden.

3. Testprogramm zu Vektor2D

Ergänzen Sie `Vektor2D` um eine Methode `void kopiereIn(„DatenTyp“ zielvar)`, die die Koordinaten des Objekts, auf dem die Methode aufgerufen wird, *in* einen `Vektor2D` *kopiert*, der über den Parameter `zielvar` erreichbar sein soll (für „Datentyp“ etwas Sinnvolles[1] einsetzen!).

- Benutzen Sie `kopiereIn`, um den Inhalt von `u(1,2)` (also `u` ein Vektor auf dem Stack mit den Koordinaten (1,2))
  - in `w` (Vektor auf dem Heap, siehe oben) und
  - in `a` (Vektor auf dem Stack, siehe letztes Aufgabenblatt) zu kopieren.

  Überprüfen Sie, ob das funktioniert hat, indem Sie sich `w` und `a` mit der Methode `ausgabe()` ausgeben lassen (beidesmal sollte `(1,2)` ausgegeben werden).

4. Globale Funktion

Implementieren Sie in `Vektor2D.cpp` eine globale Funktion `tausche(a,b)`[2] (ähnlich zu der Funktion `swap` in „CPR_05_Zeiger.pdf“, Folie 21[3], aber bitte `tausche` nennen, weil `swap` schon existiert!), die den Inhalt von zwei Instanzen von `Vektor2D` vertauscht. Benutzen Sie dazu die Methode `kopiereIn`!

- Testen Sie Ihre Funktion am Ende des Hauptprogramms mit den Variablen `z` und `a`.
- Die cpp-Datei soll dabei nicht per `include` in das Hauptprogramm eingebunden werden! Tipp dazu: schauen Sie nochmal Folie 56[3] in „CPR_02_Kompiliervorgang.pdf“ an!

---

[1] Typ von `zielvar` soll aber *keine* sog. Referenz sein. Referenzen hatten wir noch nicht in der Vorlesung. (Referenzen werden mit & angelegt – wenn Sie davon noch nichts gehört haben, umso besser: dann diese Fußnote komplett ignorieren!). Insbesondere soll die Signatur *nicht* `void kopiereIn(Vektor2D&)` sein.

[2] Auch hier: *keine* sog. Referenz verwenden sein. Also *nicht* `void tausche(Vektor2D&,Vektor2D&)`.

[3] Die Foliennummer steht oben links auf den Folien, diese ist i.A. nicht identisch mit der PDF-Seitennummer!