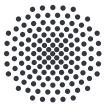
Universität Stuttgart
Institut für
Informationssicherheit

InformatikII - Programmierung in C++
Sommersemester 2023
Maria Unger-Zimmermann

Übungsblatt 4

08.05., 09.05. und 12.05

Problem 4.1: Strukturen (`struct`)

1.

Definieren Sie in einem neuen Projekt eine Struktur mit Namen `adresse` und Elementen

- `strasse` (Datentyp `string`) und
- `nummer` (`int` – dass man auch in Haus 1b wohnen kann, ignorieren wir).

Definieren Sie in der Funktion `main()` eine Variable vom Typ `adresse`, lesen Sie deren Elemente von `cin` ein und geben Sie sie auf `cout` wieder aus. (Zur Ein- und Ausgabe des Straßennamens ist ggf. wieder `#include <string>` nötig, vgl. Aufgabe **??**)

2. Definieren Sie im gleichen Programm eine weitere Struktur `brief` mit Elementen

   - `absender` (Datentyp `adresse`),
   - `empfaenger` (Datentyp `adresse`),
   - `gewicht` (Datentyp `double` – als Einheit stellen wir uns z.B. Gramm vor).

   Definieren Sie in der Funktion `main()` eine Variable vom Typ `brief`, besetzen Sie alle darin enthaltenen Elemente durch Zuweisungen mit geeigneten Werten und geben Sie die Elemente auf `cout` aus.

Problem 4.2: Vektoren, Teil 1: Grundlagen

In dieser Aufgabe üben wir grundlegende Techniken mit Vektoren:

- Deklaration und Herstellen eines Vektors
- Zugriff auf die Elemente des Vektors
- Schleifen über die Elemente des Vektors

Der Benutzer wird dazu Zahlen eingeben können, die wir ein einem Vektor speichern und von denen wir die Summe und den maximalen Wert bestimmen.

Schreiben Sie dazu ein Programm, bei dem in der `main`-Funktion folgendes passiert:

1. Zuerst soll der Benutzer gefragt werden, wie viele Zahlen er eingeben will; diese Anzahl n soll eingelesen werden (`cin`) – falls sie kleiner als 1 oder größer als 100 ist, geben wir eine Fehlermeldung aus und beenden das Programm mit Returncode 1.
2. Dann soll ein Vektor a von n `double`-Werten definiert und hergestellt werden.

   Aus der Vorlesung wissen wir schon folgendes über Vektoren (s. auch Breymann, Kap. 1.9.3 und 1.9.5):

   - Wenn die Definition des Datentyps `vector` mittels `#include <vector>` eingebunden wurde, haben wir z.B. einen Datentyp `vector<double>` zur Verfügung.
   - Wenn wir davon eine Variable definieren, können wir hinter dem Variablennamen in runden Klammern die (anfängliche) Zahl der Elemente spezifizieren.
   - Die Elemente eines Vektors `a` heißen `a[0]`, `a[1]`,…

3. Nun lesen wir in einer Schleife die n Elemente $a_0, \ldots, a_{n-1}$ ein (jeweils mit einer passenden Aufforderung an den Benutzer) und geben in einer weiteren Schleife alle eingelesenen Werte zur Kontrolle wieder aus.
4. Dann berechnen wir die Summe
$$s := \sum_{i=0}^{n-1} a_i$$
aller n gespeicherten Werte und geben sie aus.
5. Schließlich berechnen wir den maximalen Wert m und dessen Index i_m
$$m := \max_{0 \le i < n} a_i, \quad i_m := \operatorname*{argmax}_{0 \le i < n} a_i$$
(d.h. $m = a_{i_m}$) und geben beides aus (wenn der maximale Wert mehr als einmal im Feld vorkommt, soll i_m der Index irgendeines dieser maximalen Elemente sein).

Ein Dialog mit dem Programm könnte dann so aussehen (Eingaben des Benutzers unterstrichen):

```
Wieviele Zahlen: 3
a[0] eingeben: 2
a[1] eingeben: 5.3
a[2] eingeben: -3
a[0] = 2
a[1] = 5.3
a[2] = -3
Summe: 4.3
Maximales Element: a[1] = 5.3
```