

1. Semesters Auskunft

- **Besichtigung** der Klausur von Wintersemester
26.03.2009, Donnerstag 17:30 h.-18:15 h.
Computerraum IWZ 24-25
- **Nachklausur**
02.04.2009, Donnerstag 18:00 h.-19:30 h.
Computerraum IWZ 24-25

2. Semester

Organisatorisch

- 13 Vorlesungen + 1 Überblickvorlesung (jeweils 45 + 15 min.)
- 13 Übungsstunde 45 min.

2. Semester

Organisatorisch

- Einverständniserklärung: in 2 Wochen.
- Semester begleitende Prüfung in elektronischer Form. Klausur besteht aus 3 E-Tests jeweils 1/3 der ganzen Note.
 - 1. E-Test 30.04 - 01.05
 - 2. E-Test 04.06 - 05.06.
 - 3. E-Test 25.06.- 26.06
- Vorprüfung (04.07.08, Freitag 10:00 h.-11:30h.)
 - Nachprüfung am Anfang des nächsten Semesters
 - 3. Versuch

$$Gesamtnote = \frac{1}{3} \sum_{i=1}^3 Etest_i$$

2. Semester

Organisatorisch

- Weitere Möglichkeit. Gesamte elektronische Klausur (3 x 30 Minuten) im Prüfungszeitraum.
Fr. 03.07.2009 um 10:30
 - Insbesondere für Studenten aus dem höheren Semester

2. Semester

Organisatorisch

- Detaillierte Auskunft über Lehrveranstaltungen und E-Tests. Im Internet www.kramann.info unter dem Link [Informatik2Material](#) befindet sich [Termine_Purcalla](#).

Überblick des Faches

- Lernziele
 - Gewöhnlich Funktionen + Arrays verwenden / programmieren.
 - Selbst programmierte
 - Nicht selbst programmierte
 - Algorithmen entwickeln, bewerten und benutzen.
 - numerische Algorithmen (Integration, Ableitung)
 - Modularisierung
 - Vorprogrammierte Module
 - Programmierung eigener Module
 - Möglichkeiten anderer Bibliotheken
 - Ein- / Ausgabe durch Datei
 - Speicherplatz
 - Speicheradressen
 - Verwaltung

Überblick der heutigen Vorlesung

- Arrays
 - Numerisch
- Arrays mit Funktionen: Verwendung von Arrays.
 - Array als Parameter

Arrays

Kenntnisse aus dem ersten Semester

Eindimensionale Felder.

- Bedeutung des Begriffs
- Benutzung
 - Deklaration
 - Initialisierung
 - Verwendung im Programm
 - Von einer Position / allen Positionen
 - Wert abfragen
 - Wert speichern

Aufgabe1: von Problem zur Programm

Aufgabe 1: Es soll ein Programm entwickelt werden, das 10 ganze Zahlen einliest, und deren Summe ermittelt.

$$\begin{array}{c} a_1 \\ a_2 \\ \vdots \\ a_i \\ \vdots \\ a_{10} \end{array} \rightarrow a_1 + a_2 + \cdots + a_i + \cdots + a_{10}$$

Lösungsweg.

Schlüsselschritte.

- a) Eingabe: in einem **Array** gespeichert
Zahlen[0],.....,Zahlen[9].
- b) Summe durch **Schleife** durchführen
- c) Ausgabe: Summe

Detaillierter Lösungsweg.

a) Eingabe in einem **Arrayelement**, Position i

cin >> Zahlen[i] ;

Detaillierter Lösungsweg.

a) Eingabe in allen **Arrayelementen**

```
for ( i = 0; i < 10; i++ )  
    cin >> Zahlen[i] ;
```

Detaillierter Lösungsweg.

b) Addition eines **Arrayelements** in einer Variable.

`summe = summe + Zahlen[i] ;`

Detaillierter Lösungsweg.

b) Addition aller **Arrayelemente** in einer Variable.

```
for ( i = 0; i < 10; i++ )  
    summe = summe + Zahlen[i] ;
```

Entwicklung des Programms

Wir benötigen ,wieder die Werkzeuge aus dem ersten Semester.

#define

```
#define ZEICHEN AUSDRÜCK
```

- Ersetzt überall im Programm wo ZEICHEN auftritt, durch AUSDRÜCK
- Vor der Ausführung (preprocessor Direktive)
- Besonders nützlich bei im Programm wiederholte Konstanten

#define bei Arrays

```
#define N Zahl_der_Arrayelemente
```

Lösung der Aufgabe 1:

Die Lösung liegt in der Datei *summe.cpp*

Weiterentwicklungen des Programms

- **Aufgabe 2:** Ein weitere Reihe von 10 Werten im Programm addieren:

summe2.cpp

$$\begin{array}{c} a_1 \\ a_2 \\ \vdots \\ a_i \\ \vdots \\ a_{10} \end{array} \rightarrow a_1 + a_2 + \cdots + a_i + \cdots + a_{10}$$

$$\begin{array}{c} b_1 \\ b_2 \\ \vdots \\ b_i \\ \vdots \\ b_{10} \end{array} \rightarrow b_1 + b_2 + \cdots + b_i + \cdots + b_{10}$$

Weiterentwicklungen des Programms

- **Aufgabe 3:** Die Summe der Vektoren als **Funktion** programmieren und ausführen

Aufgabe 3: Funktion Summe

- Ziel: Die Summe von 2 Vektoren mittels einer Funktion durchführen.
- Dafür sind benötigt Kenntnisse von:
 - Vektoren
 - Funktionen
 - der Zusammenwirkung zwischen Vektoren und Funktionen.
- In der Aufgabe 1 sind schon Vektoren bearbeitet.
- In den folgenden Folien wird die Lehre von Funktionen wiederholt.

Funktionen (Siehe letztes Semester)

Kenntnisse aus dem ersten Semester

- Bedeutung des Begriffs

- Kennzeichen
 - Parameter (Übergabe)
 - Namen
 - Datentypen
 - Name
 - Rückgabe
 - Datentyp

- Schreibweise im Programm
 - Deklaration (Kopf)
 - Aufruf(e): Verwendung mit bestimmten Parameter-Werten
 - Definition (Körper)

Funktionsbeispiel ohne Vektoren

Berechnung des Betrags von 3 Zahlen

- Ohne Funktion: *betrags.cpp*
- Als C Funktion *betragsfunc.cpp*
 - Parameterdatentyp: *int* (ganze Zahl)
 - Funktionsname: *betrags*
 - Rückgabedatentyp: *unsigned int* (positive ganze Zahl)

Aufgabe 3: Lösung

*Die Summe der Vektoren als **Funktion***

- Die Verwendung von Funktionen und Arrays zusammen kann man endlich im Programm *summe2func.cpp* beobachten.

- Parameterdatentyp: *int []* (Array ganzer Zahlen)
- Funktionsname: *betrag*
- Rückgabedatentyp: *int* (ganze Zahl)

Aufgabe 4: Ein praktischer Fall

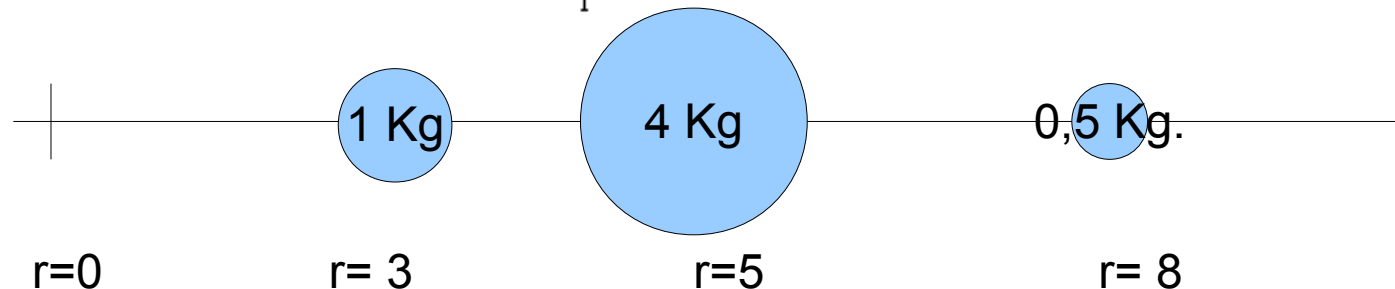
Der Schwerpunkt

- Berechnung des Schwerpunkts einer Menge von Massen auf einer 1-eindimensionalen Raum (einer Gerade)

Wir haben

- den Array der Positionen $r[i]$
- den Array der Massen $m[i]$
- die Formel

$$\vec{r}_{cm} = \frac{\sum_{i=1}^N m_i \vec{r}_i}{\sum_{i=1}^N m_i}$$



Aufgabe 4: Lösungsweg.

Schlüsselschritte.

a) Eingabe der Arrays:

b) Summe der Massen durch eine **Schleife**

$$\sum_{i=1}^N m_i$$

c) Summe der Produkt Massen mal Position durch eine weitere **Schleife**

$$\sum_{i=1}^N m_i r_i$$

d) Berechnung und Ausgabe des Schwerpunktes.
(Division beider Summen)

Aufgabe 4: Lösung

- Lösung: *Schwerpunkt.cpp*