

Computer Vision  
and Geometry Lab

# Informatik I for D-MAVT

Exercise Session 13

# Organisatorisches

- Teaching Assistant

- Alexander Schwing([aschwing@inf.ethz.ch](mailto:aschwing@inf.ethz.ch))

- CAB G 89

- Website:

- <http://www.alexander-schwing.de/Info1DMaVt/>

- Übungsabgabe

- Auf Papier, keine Emails

- Testatbedingungen

- 75% aller Übungen

- 12 Serien (d.h. 9 gelöste Serien)

# Themen

- Nachbesprechung
- Prüfungsvorbereitung
- Besprechung Übung 12
- Fragen
- Testat

# Nachbesprechung

- Aufgaben genau lesen

# Prüfungsvorbereitung

## ■ C++ Syntax:

Gegeben Code-Fragmente ist zu entscheiden ob es kompiliert, einen Laufzeitfehler ergibt in einer Endlosschleife mündet oder funktioniert (jeweils mit Begründung)

# Prüfungsvorbereitung

## ■ Beispiel:

```
1 double a = 3.0;  
2  
3 do {  
4     a = a / 2;  
5 } while (a < 2.0);
```

# Prüfungsvorbereitung

## ■ Casting

Gegeben Code-Fragmente, deklariere die Variablen so dass kein implizites casting ausgeführt wird.

Beachte: Modulodivision für integrale Variablen

# Prüfungsvorbereitung

## ■ Beispiel:

`a->c[a->b] = b[c] % 10;`

# Prüfungsvorbereitung

## ■ Schleifen

Gegeben eine Schleife (for, while, do), programmiere diese in einen anderen Typ um.

Beachte: Umprogrammieren in rekursive Funktion

# Prüfungsvorbereitung

## ■ Beispiel (umwandeln in do-while):

```
int t = 17, f = 0;
while(t > 3)
  t /= 2;
  f += t;
```

```
int t = 17, f = 0;
if( t > 3)
  do
    t /= 2;
    while(t > 3);
  f += t;
```

# Prüfungsvorbereitung

- Programmanalyse

Gegeben ein Programm, was macht es und wie sieht die Ausgabe aus?

# Prüfungsvorbereitung

## ■ Beispiel:

```
struct foo {  
    foo* a;  
    double b;  
};
```

```
foo *a;  
a = new foo;  
a->b = 3.9;  
a->a = new foo;  
a->a->b = 1.3;  
a->a->a = new foo;  
a->a->a->b = 8.4;  
a->a->a->a = 0;
```

# Prüfungsvorbereitung

## ■ Parameterübergabe

Gegeben Funktionsköpfe, wie können dies aufgerufen werden und welche Aufrufe verursachen Kompiler-Fehler?

Beachte: const, By-Value, By-Reference, Pointer, etc.

# Prüfungsvorbereitung

## ■ Beispiele:

Prefix: T1& operator ++(T1& a); //++a

Postfix: T1 operator ++(T1& a, int); //a++

<pre>void incr(int&amp; y) {     y++; }</pre>	<pre>incr(5); //Kompilerfehler</pre>	<pre>int i = 5; incr(++i); // i ist 7</pre>	<pre>int i = 5; incr(i++); //Kompilierfehler</pre>
<pre>void incr(int* y) {     (*y)++; }</pre>	<pre>int i = 5; incr(&amp;i++); //Kompilierfehler</pre>	<pre>int i = 5; incr(&amp;i); // i ist 6</pre>	<pre>int i = 5; incr(&amp;++i); // i ist 7</pre>
<pre>void incr(int y) {     y++; }</pre>	<pre>int i = 5; incr(i); // i ist 5</pre>		
<pre>int incr(int y) {     return y++; }</pre>	<pre>int i = 5; i = incr(i); // i ist 5</pre>		
<pre>int incr(int y) {     return ++y; }</pre>	<pre>int i = 5; i = incr(i); // i ist 6</pre>		

# Prüfungsvorbereitung

## ■ Funktionen

Gegeben eine mathematische Berechnungsvorschrift,

e.g. 
$$e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!}$$

Programmieren Sie eine Funktion, die obiges ohne Verwendung von `<cmath>` berechnet.

Beachte: Abbruchkriterium, keine Endlosschleife!!!

# Prüfungsvorbereitung

## ■ Beispiel (ohne Gewähr):

```
double expx(double x) {
    double tmpx = 1;
    double faculty = 1;
    double res = 0;
    double factor;
    int cnt = 1;
    do {
        factor = tmpx/faculty;
        res += factor;
        tmpx *= x;
        faculty *= cnt++;
    } while(factor>1e-12);
    return res;
}
```

# Prüfungsvorbereitung

- Strings und Zahlen  
Umwandlung

# Prüfungsvorbereitung

- Beispiel: siehe letzte Übung

# Prüfungsvorbereitung

## ■ Klassen

Programmieren, Memberfunktionen, Konstruktoren,  
Destruktoren, Operatoren, friend-Funktionen  
Beachte: public, private, protected

# Prüfungsvorbereitung

- Beispiel: siehe letzte Übung

# Besprechung Übung 12

## ■ Aufgabe 1)

### ■ A) struct Person

```
struct Person {  
    string Name;  
    Person* Vater;  
    Person* Mutter;  
}
```

### ■ B) Stammbaum

```
Person shmi = {"Shmi" , 0 , 0 };  
Person anakin = {"Anakin" , 0 , &shmi };  
Person padme = {"Padme" , 0 , 0 };
```

### ■ C) Großvater

```
ben.vater->vater->name
```

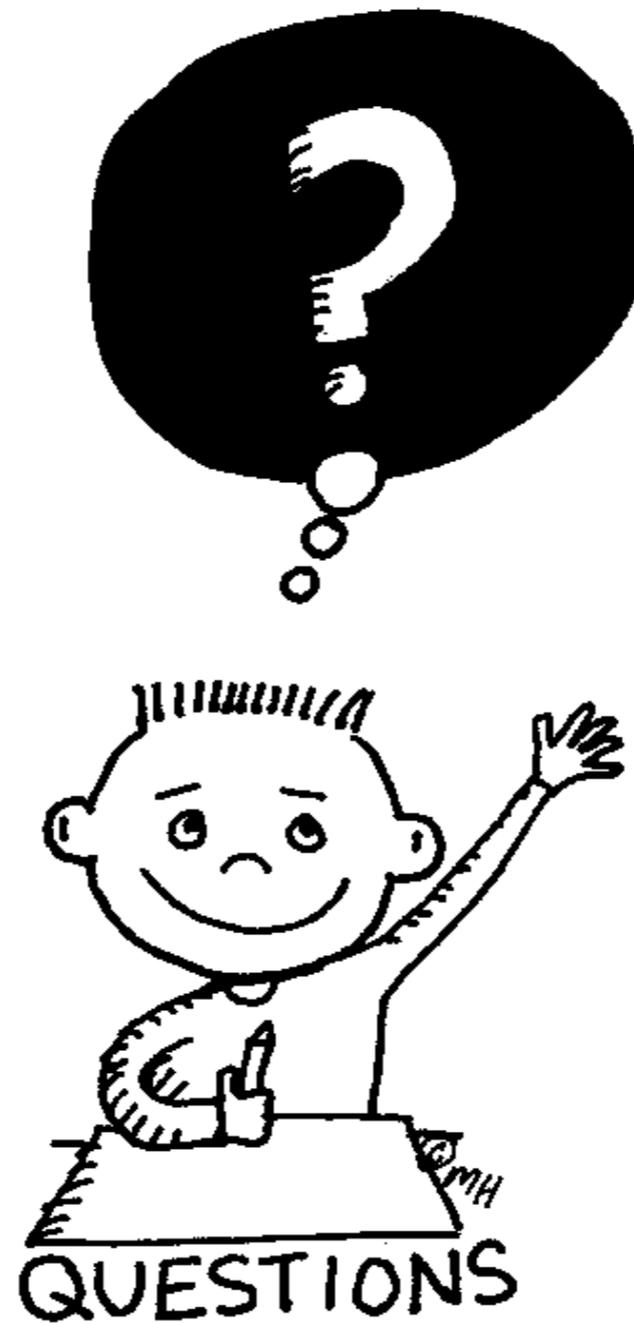
### ■ D) Kinder

```
struct Person {  
    string Name;  
    Person* Vater;  
    Person* Mutter;  
    list<Person*> Kinder;  
}
```

# Besprechung Übung 12

- Aufgabe 2) Generieren von Benutzernamen

# Fragen



# Testat

- Mittelwert: 8.33 Punkte
- Median: 9 Punkte
- Standardabweichung: 3.06 Punkte

# Viel Erfolg für die Prüfung

