



Computer Vision  
and Geometry Lab

# Informatik I for D-MAVT

## Exercise Session 12

# Organisatorisches

- Teaching Assistant
  - Alexander Schwing ([aschwing@inf.ethz.ch](mailto:aschwing@inf.ethz.ch))
  - CAB G 85.1
  - Website: <http://www.alexander-schwing.de/Info1DMaVt/>
- Übungsabgabe
  - Auf Papier, keine Emails
- Testatbedingungen
  - 75% aller Übungen
    - 12 Serien (d.h. 9 gelöste Serien)

# Themen

- Evaluierung
- Nachbesprechung Übung 10
- String Klasse
- File I/O
- Vorbereitung Übung 12

# Evaluierung

- Fragen
- Anregungen
- Anmerkungen

# Nachbesprechung: Übung 10

## ■ Konstruktor

```
cPolynom::cPolynom(double* a, int d) {  
    deg = d;  
    coef = a;  
}
```

Falsch

```
cPolynom::cPolynom(double* a, int d) {  
    deg = d;  
    coef = new double[d+1];  
    for(int i = 0; i <= d; i++){  
        coef[i] = a[i];  
    }  
}
```

Richtig

## ■ Destruktor

```
cPolynom::~~cPolynom() {  
    deg = 0;  
    coef = 0;  
}
```

Falsch

```
cPolynom::~~cPolynom() {  
    deg = -1;  
    delete[] coef;  
}
```

Richtig

# Nachbesprechung: Übung 10

## ■ operator+()

Konstanter Rückgabewert

Konstante Variable:

Objekt **pin** wird in der Methode nicht verändert

```
const cPolynom cPolynom::operator+(const cPolynom& pin) const{  
    int m = (deg > pin.deg)? deg : pin.deg;  
    cPolynom res(m);  
    for(int i = 0; i <= deg; i++){  
        res.coef[i] = coef[i];  
    }  
    for(int i = 0; i <= pin.deg; i++){  
        res.coef[i] += pin.coef[i];  
    }  
    return res;  
}
```

Konstante Methode:

**this** Objekt wird nicht verändert

# Nachbesprechung: Übung 10

## ■ Konstruktor 2

```
cPolynom::cPolynom(int degree) {  
    deg = degree;  
    coef = new double[deg+1];  
    for(int k=0;k<=deg;++k)  
        coef[k] = 0;  
}
```

## ■ Copy-C'tor

```
cPolynom::cPolynom(const cPolynom& r) {  
    deg = r.deg;  
    coef = new double[deg+1];  
    for(int k=0;k<=deg;++k)  
        coef[k] = r.coef[k];  
}
```

## ■ operator=()

```
cPolynom& cPolynom::operator =(const cPolynom& r) {  
    // Check for self-assignment!  
    if (this != &r) {  
        this->~cPolynom();  
        deg = r.deg;  
        coef = new double[deg+1];  
        for(int k=0;k<=deg;++k)  
            coef[k] = r.coef[k];  
    }  
    return *this;  
}
```

# Nachbesprechung: Übung 10

- operator-(): siehe operator+()
- Eval Funktion: z.B. Horner Schema
- Alte Prüfungsaufgabe:
  - operator\*()
  - Funktion: derive
- Weitere gute Übung:
  - Funktion: integrate

# String Klasse

- String Klasse repräsentiert Zeichenkette in cpp
- Einbinden der String-Klasse

```
#include<string>  
using namespace std;
```

- Wichtige Konstruktoren:

```
string() // Initialisiert leeren String  
string(const string& s) // Initialisierung mit Objekt s  
string(const char* s) // Initialisierung mit char array s,  
char array ist nullterminiert ('\0')
```

# String Klasse

## ■ Wichtige Funktionen:

```
size_t length()
```

```
size_t size()
```

```
void clear()
```

```
bool empty()
```

```
string substr(size_t pos, size_t n= npos)
```

```
const char* c_str() const
```

```
size_t find(const string& str, size_t pos = 0)
```

```
char& operator[](size_t pos)
```

Länge des Strings

Länge des Strings

String Inhalt wird geleert

Testet ob String leer ist

Erstellt einen Substring vom aktuellen Objekt

Erstellt einen c-String

Finde Substring **str** im aktuellen Objekt, starte an Position **pos**. Rückgabewerte Position **str** oder **npos** falls nicht gefunden

Extrahiere ein Zeichen des Strings an Position **pos**

# Umwandeln von Zahlen und Zeichenketten

- Zeichenkette zu Zahl umwandeln mit der `istringstream` Klasse
- Bsp:

```
double d;  
string s(34.56);  
istringstream inString(s);  
inString >> d;
```

# Umwandeln von Zahlen und Zeichenketten

- Zahl zu Zeichenkette umwandeln mit `ostringstream` Klasse
- Bsp :

```
double d = 45.32;  
string s;  
ostringstream outString;  
outString << d;  
s = outString.str();
```

# File I/O

## ■ Schreiben und lesen von Files

`ofstream`

Stream Klasse um Files zu schreiben

`ifstream`

Stream Klasse um Files zu lesen

`fstream`

Stream Klasse um Files zu lesen und schreiben

## ■ File öffnen

`open(const char *fileName, mode)`

`fileName:` Name des Files

`Mode:`

`ios::in`

Lesen

`ios::out`

Schreiben

`ios::binary`

Öffnen in Binärmodus

`ios::app`

Daten am Ende anfügen

`ios::ate`

Positionszeiger ans

Ende setzen

## ■ File lesen/schreiben wie Konsole

`>>`

Lesen

`<<`

Schreiben

## ■ File schliessen

`close()`

Schliessen

# File I/O: Beispiel

## ■ File einlesen:

```
string vorname, nachname;

// inFile.txt enthält pro Zeile zwei Strings

ifstream myfile("inFile.txt"); // Öffnen der Datei
if (myfile.is_open()){ // Datei wurde erfolgreich geöffnet

    myfile >> vorname >> nachname; // 2 Strings einlesen

    while(!myfile.eof()){
        cout << vorname << " " << nachname << endl;
        myfile >> vorname >> nachname;
    }

    myfile.close();
}
```

# File I/O: Beispiel

- File schreiben:

```
ofstream myfile("outName.txt"); // Öffnen der Datei

if (myfile.is_open()){ // Datei wurde erfolgreich geöffnet

    myfile << "First line" << endl;
    myfile << "Second line" << endl;
    myfile.close();
}
```

# Vorbesprechung: Übung 12

## ■ Aufgabe I: Stammbaum

### ■ a) Verkettete Liste

```
struct Person {  
    string Name;  
    Person* RechterNachbar;  
}
```

### ■ b) Liste füllen (auf Reihenfolge aufpassen)

```
Person Luca = {"Luca", NULL};  
Person Tim = {"Tim", &Luca};  
Person Patrick = {"Patrick", &Tim};
```

### ■ c) Auslesen

```
Patrick->RechterNachbar->RechterNachbar->Name;
```

### ■ d) Erweitern

# Vorbesprechung: Übung 12

- Aufgabe II: Generierung von Usernamen
  - Bedingung die der Username erfüllen muss:
    - Eindeutig und genau 6 Zeichen lang
    - Besteht ausschliesslich aus Kleinbuchstaben
    - Besteht aus Name und Vorname
    - Falls alle Usernamen bereits vergeben füge Zahl an
  - Mögliche Usernamen für Peter Mueller
    - pmuell, muelpe, petmue etc.

# Vorbesprechung: Übung 12

- Vorgehen:

- Einlesen der Vornamen/Nachnamen
- Konvertiere Vorname/Nachname zu Kleinbuchstaben

```
#include<cctype>
```

```
vorname[i] = tolower(vorname[i])
```

- Generiere Username

```
tmpuser = vorname.substr(0, i) + nachname.substr(0, 6-i);
```

- Es gibt 10 Kombinationen für die Länge 6

- Bsp: Peter Mueller

```
pmuell, pemuel, petmue, petemu, peterm
```

```
mpeter, mupete, muepet, muelpe, muellp
```

- Überprüfe ob `tmpuser` die länge 6 hat

# Vorbesprechung: Übung 12

- Vorgehen (cont):

- Überprüfe ob der Username bereits existiert

- Speichere all generierten Usernamen in einem String

```
usernames = usernames + " " + username;
```

- Suche Username

```
foundPos = usernames.find(username);
```

```
if (foundPos == string::npos) // Name nicht gefunden
```

- Falls alle Usernamen vergeben sind füge eine Zahl an

Bsp: **pmuell** wird zu **pmue13**

- Siehe Slides zum Thema: Zahl zu Zeichenketten umwandeln