Datenstrukturen und Algorithmen

Labor

Organisatorisches

- Kontakt: E-Mail vsolea[at]cs.ubbcluj.ro oder Teams

- Laborregeln bitte **aufmerksam** durchlesen!
  Files / Class Materials / Labor
- Interfaces und Teste für Hausaufgaben:
  Files / Class Materials / Labor / Interfaces
- Andere Ressourcen:
  - Videotutorial Interface und Struktur der Projekte
  - Videotutorial Debugging
  - C++ Cheat Sheet
  - Dynamische Allokation in C++ (in diesem Dokument)

Statische Allokation in C++

```
int a[20];
```

- Statische Allokation, wird bei Kompilierung bestimmt
- Länge des Vektors muss konstant sein, d.h. bei Kompilation bekannt.
- Kann zu Speicherverschwendung oder Mangel führen wenn man die benötigte Kapazität nicht gut erraten kann
- Lebensdauer der Variablen: von Deklaration bis zum Ende der Funktion

Dynamische Allokation in C++

```
//...
int *a;
int n = someResult();
a = new int[n];
//do stuff with a
delete[] a;
//...
```

- `int *a` definiert ein **Pointer** zu einem Integer. `a` ist die Adresse eines Speicherplatzes, dessen Inhalt als int interpretiert werden soll.
- Man kann Pointer zu jedwelchen Datentypen definieren
- Dynamische Allokation: Größe wird bei Runtime bestimmt
- Manipulation wie bei statischen Arrays (mehreres später bei OOP)
- Lebensdauer der Variablen: von `new` bis `delete`

Dynamische Allokation in C++

- `new` und `delete` sind Operatoren, nicht Funktionen.
- Beim Aufrufen von `new` wird **zuerst** genügend Speicherplatz reserviert und **danach** der Constructor aufgerufen
- Beim Aufrufen von `delete` wird **zuerst** der Destructor aufgerufen und **danach** der Speicherplatz befreit
- Das Benutzen der Funktionen `malloc()` und `free()` ist in C++ deshalb unerwünscht
- Zum Unterschied von anderen Programmiersprachen (z.B. Python, Java) hat C++ kein garbage collector: **allokierte Objekte immer selbst deallokieren!**

Dynamische Allokation in C++

`delete myptr;` - löscht ein einziges Element

`delete[] myptr;` - löscht ein Array

Dynamische Allokation in C++

Derefernzierungsoperator −> für Mitglieder

```
struct point{
    double x, y, z;
};
//...
point *A;
//...
A->x = 5;
```