Westfälische
Hochschule
Gelsenkirchen Bocholt Recklinghausen

Grundlagen der Informatik und Programmierung 2

Objektorientierte Programmierung

Motivation

Prof. Dr. Tom Vierjahn

Visual Computing (https://vc.w-hs.de)
Fachbereich Wirtschaft und Informationstechnik
Campus Bocholt

Sommersemester 2020

Ein Haufen Code

Westfälische Hochschule

Programmrahmen:

```c
void* data = NULL;
void* data_position = NULL;
unsigned int count = 0;

int main(int argc, char** argv) {
  data = malloc(100 * 96);
  data_position = data;

  /* ... */

  free(data);
}
```

Speicher

Westfälische Hochschule

0x	…0	…1	…2	…3	…4	…5	…6	…7	…8	…9	…a	…b	…c	…d	…e	…f
00…																
01…																
02…																
03…																
04…																
05…																
06…																

Ein Haufen Code

Westfälische Hochschule

```
21 *(unsigned int*)data_position = 199100001;
22 data_position += sizeof(unsigned int);
23
24 strcpy(data_position, "Jane");
25 data_position += 46;
26
27 strcpy(data_position, "Appleseed");
28 data_position += 46;
29
30 ++count;
```

Speicher

Westfälische Hochschule

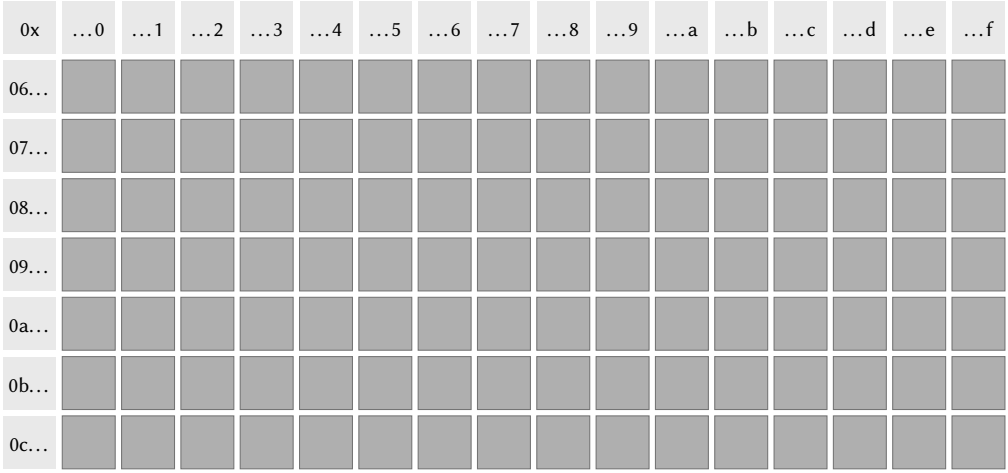
0x	…0	…1	…2	…3	…4	…5	…6	…7	…8	…9	…a	…b	…c	…d	…e	…f
00…																
01…																
02…																
03…																
04…																
05…																
06…																

Ein Haufen Code

Westfälische Hochschule

```
41 *(unsigned int*)data_position = 199100002;
42 data_position += 4;
43
44 const char* first_name = "John";
45 const char* last_name = "Doe";
46
47 strcpy(data_position, first_name);
48 strcpy(data_position + 46, last_name);
49
50 data_position += 92;
51 ++count;
```

Speicher

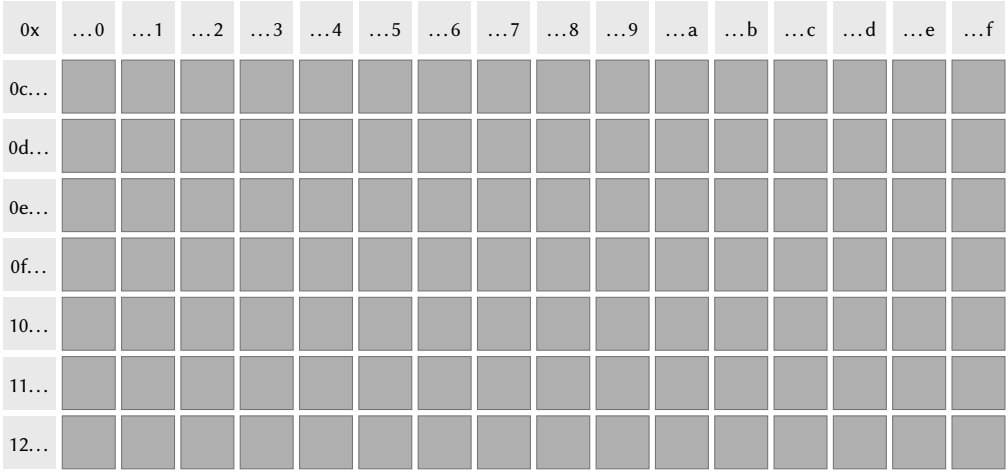


Ein Haufen Code

Westfälische Hochschule

```
61 *(unsigned int*)data_position = 199100003;
62 strcpy(data_position + 4, "J. Random");
63 strcpy(data_position + 50, "Student");
64
65 data_position += 96;
66 ++count;
```

Speicher



Ein Haufen Code

Westfälische Hochschule

```
81 for (int i = 0; i < count; ++i) {
82   printf("%d ", *(unsigned int*)(data + i * 96));
83
84   for (int j = 0; j < 2; ++j) {
85     printf("%s ", (char*)(data + i * 96 + 4 + j * 46));
86   }
87   printf("\n");
88 }
```

Ausgabe:

Ein kleinerer Haufen Code

jetzt mit Datenstrukturen

Westfälische Hochschule

Globaler Scope:

```
1 struct Student {
2   unsigned int id;
3   char first_name[46];
4   char last_name[46];
5 };
6
7 Student* data = nullptr;
8 Student* data_position = nullptr;
9 unsigned int count = 0;
```

Ein kleinerer Haufen Code

jetzt mit Datenstrukturen

Westfälische Hochschule

main-Funktion:

```
10 int main(int argc, char** argv) {
11   data = (Student*)malloc(100 * sizeof(struct Student));
12   data_position = data;
13
14   /* ... */
15
16   free(data);
17 }
```

Speicher

Westfälische Hochschule

0x	…0	…1	…2	…3	…4	…5	…6	…7	…8	…9	…a	…b	…c	…d	…e	…f
00…																
01…																
02…																
03…																
04…																
05…																
06…																

Ein kleinerer Haufen Code

jetzt mit Datenstrukturen

Westfälische Hochschule

```
21 Student s1{199100001, "Jane", "Appleseed"};
22 *data_position = s1;
23
24 ++data_position;
25 ++count;
```

Speicher

Westfälische Hochschule

0x	…0	…1	…2	…3	…4	…5	…6	…7	…8	…9	…a	…b	…c	…d	…e	…f
00…																
01…																
02…																
03…																
04…																
05…																
06…																

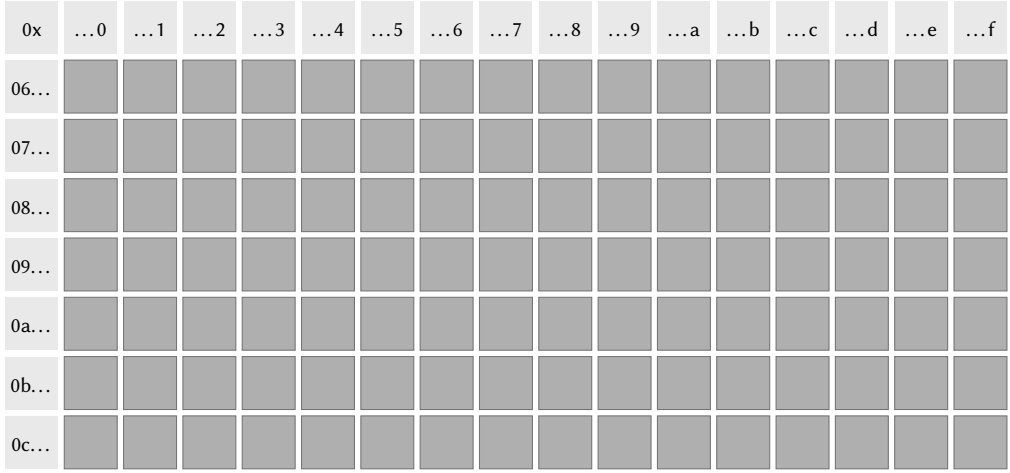
Ein kleinerer Haufen Code

jetzt mit Datenstrukturen

Westfälische Hochschule

```
41 *(unsigned int*)data_position = 199100002;
42 strcpy((char*)data_position + 4, "John");
43 strcpy((char*)data_position + 4 + 46, "Doe");
44
45 ++data_position;
46 count = data_position - data;
```

Speicher



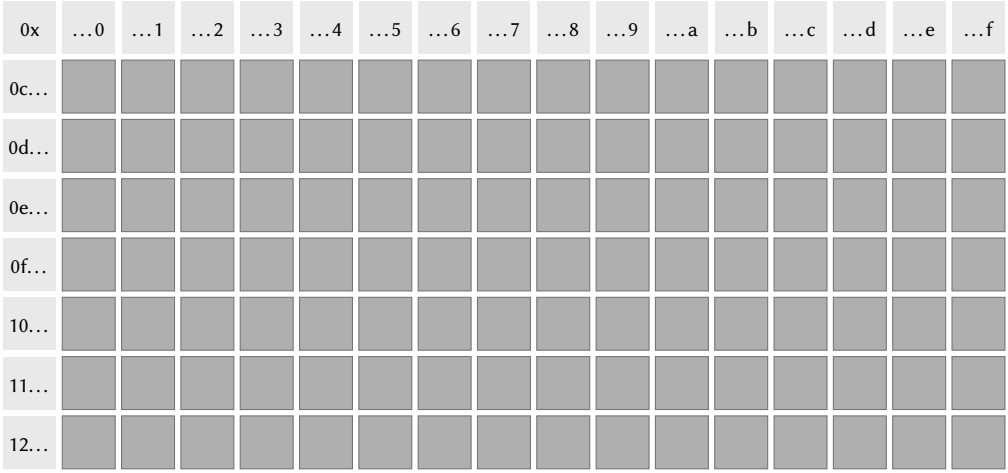
Ein kleinerer Haufen Code

jetzt mit Datenstrukturen

Westfälische Hochschule

```
61 data_position->id = 199100003;
62 strcpy(data_position->first_name, "J. Random");
63 strcpy(data_position->last_name, "Student");
64
65 ++count;
66 data_position = data + count;
```

Speicher



Ein kleinerer Haufen Code

jetzt mit Datenstrukturen

Westfälische Hochschule

```
81 for (int i = 0; i < count; ++i) {
82   printf("%d %s %s\n", data[i].id, data[i].first_name,
83          data[i].last_name);
84 }
```

Ausgabe:

Ein noch kleinerer Haufen Code

jetzt fast mit einem eigenen Container-Datentyp

Westfälische Hochschule

Global-Scope:

```
struct Student {
  unsigned int id;
  char first_name[46];
  char last_name[46];
};

Student* data = nullptr;
unsigned int count = 0;

bool InitializeData(unsigned int count) {
  data = (Student*)malloc(count * sizeof(Student));
  return data != nullptr;
}
```

Ein noch kleinerer Haufen Code

jetzt fast mit einem eigenen Container-Datentyp

Westfälische Hochschule

Global-Scope:

```
void DeleteData() {
  free(data);
  count = 0;
}

void Enroll(const Student& s) {
  data[count] = s;
  ++count;
}

unsigned int NumStudents() { return count; }

void PrintStudent(unsigned int i) {
  printf("%d %s %s\n", data[i].id, data[i].first_name, data[i].last_name);
}
```

Ein noch kleinerer Haufen Code

jetzt fast mit einem eigenen Container-Datentyp

Westfälische Hochschule

main-Funktion:

```
int main(int argc, char** argv) {
  InitializeData(100);

  /* ... */

  DeleteData();
}
```

Ein noch kleinerer Haufen Code

jetzt fast mit einem eigenen Container-Datentyp

Westfälische Hochschule

```
Enroll({199100001, "Jane", "Appleseed"});
```

```
Enroll({199100002, "John", "Doe"});
```

```
Enroll({199100003, "J. Random", "Student"});
```

Ein noch kleinerer Haufen Code

jetzt fast mit einem eigenen Container-Datentyp

Westfälische Hochschule

```
for (unsigned int i = 0; i < NumStudents(); ++i) {
  PrintStudent(i);
}
```

Ausgabe:

Ein übersichtlicherer Haufen Code

jetzt mit einem eigenen Container-Datentyp

Westfälische Hochschule

Global-Scope:

```
struct Student {
  unsigned int id;
  char first_name[46];
  char last_name[46];
};

struct StudentsList {
  Student* data = nullptr;
  unsigned int count = 0;
};
```

Ein übersichtlicherer Haufen Code

jetzt mit einem eigenen Container-Datentyp

Westfälische Hochschule

Global-Scope:

```
StudentsList ConstructStudentsList(unsigned int count) {
  StudentsList students_list;
  students_list.data = (Student*)malloc(count * sizeof(Student));
  students_list.count = 0;
  return students_list;
}

void DestructStudentsList(StudentsList* students_list) {
  free(students_list->data);
  students_list->data = nullptr;
  students_list->count = 0;
}
```

Ein übersichtlicherer Haufen Code

jetzt mit einem eigenen Container-Datentyp

Westfälische Hochschule

Global Scope:

```
void Enroll(StudentsList* students_list, const Student& s) {
  students_list->data[students_list->count] = s;
  ++(students_list->count);
}

unsigned int NumStudents(const StudentsList* students_list) {
  return students_list->count;
}

void PrintStudent(const StudentsList* students_list, unsigned int i) {
  const Student& student = students_list->data[i];
  printf("%d %s %s\n", student.id, student.first_name, student.last_name);
}
```

Ein übersichtlicherer Haufen Code

jetzt mit einem eigenen Container-Datentyp

Westfälische Hochschule

Programmrahmen:

```
int main(int argc, char** argv) {
  StudentsList enrolled_students = ConstructStudentsList(100);

  /* ... */

  DestructStudentsList(&enrolled_students);
}
```

Ein übersichtlicherer Haufen Code

jetzt mit einem eigenen Container-Datentyp

Westfälische Hochschule

```
Enroll(&enrolled_students, {199100001, "Jane", "Appleseed"});
```

```
Enroll(&enrolled_students, {199100002, "John", "Doe"});
```

```
Enroll(&enrolled_students, {199100003, "J. Random", "Student"});
```

Ein übersichtlicherer Haufen Code

jetzt mit einem eigenen Container-Datentyp

Westfälische Hochschule

```
for (unsigned int i = 0; i < NumStudents(&enrolled_students); ++i) {
  PrintStudent(&enrolled_students, i);
}
```

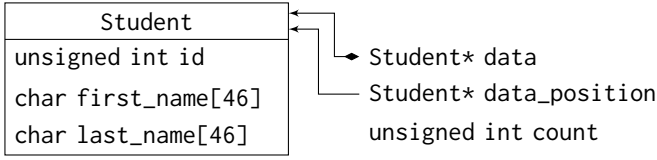
Ausgabe:

Übersichtlicher gezeichnet

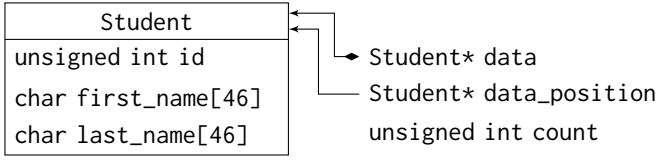
Westfälische Hochschule

```
void* data
void* data_position
unsigned int count
```

Übersichtlicher gezeichnet

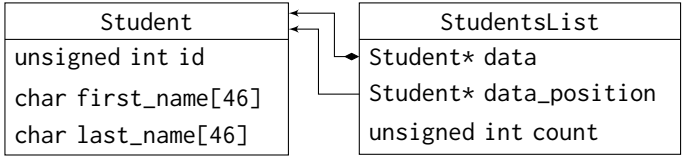


Übersichtlicher gezeichnet



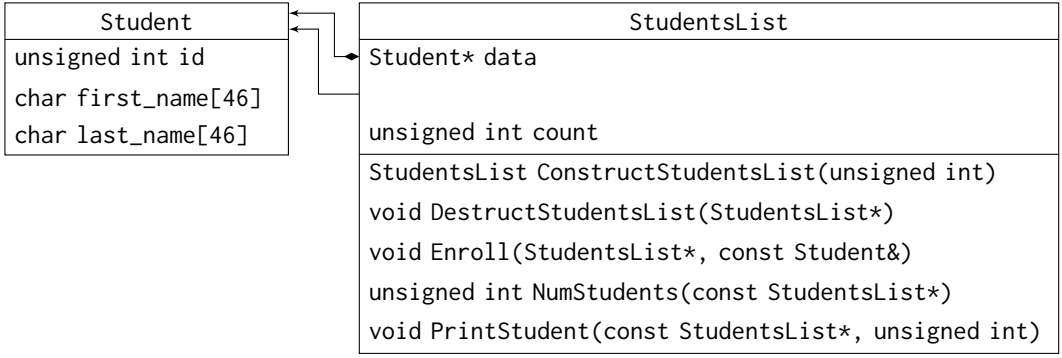

```
int InitializeData(unsigned int)
void DeleteData()
void Enroll(const Student&)
unsigned int NumStudents()
void PrintStudent(unsigned int)
```

Übersichtlicher gezeichnet




```
StudentsList ConstructStudentsList(unsigned int)
void DestructStudentsList(StudentsList*)
void Enroll(StudentsList*, const Student&)
unsigned int NumStudents(const StudentsList*)
void PrintStudent(const StudentsList*, unsigned int)
```

Übersichtlicher gezeichnet



Prof. Dr. Tom Vierjahn
- E-Mail: tom.vierjahn@w-hs.de

Visual Computing
- Web: `https://vc.w-hs.de`
- YouTube: Visual Computing WH
- Twitter: @VisComputingWH

Westfälische Hochschule
Fachbereich Wirtschaft und Informationstechnik
Campus Bocholt