OSTBAYERISCHE
TECHNISCHE HOCHSCHULE
REGENSBURG

INFORMATIK UND
MATHEMATIK

Programmieren 1 – Informatik
Übungsklausur

Prof. Dr.-Ing. Maike Stern | 13.12.2023

Rückblick

- Typedef
- Structs
- Enum
- Union

Typedef Arrays

bei Arrays wird die eckige Klammer an den neuen Bezeichner gehängt

Typedef ist ein Schlüsselwort, mit dem C-Schlüsselbegriffen ein Synonym zugewiesen werden kann

- Syntax
  typedef Datentyp neueBezeichnung;
  typedef Datentyp neueArrayBezeichnung[];

Datentyp, der eine zusätzliche Bezeichnung erhalten soll

eckige Arrayklammern, mit oder ohne Dimensionsangabe

Alias für den eigentlichen Datentyp

```c
int main(){
  typedef int myInt;
  typedef char string[100];

  myInt i = 7;
  string s1 = "Hello, World!",

  return 0;
}
```

Structs
Die Syntax

In der Regel werden Structs wie Funktionen vor der main-Funktion deklariert. Es gibt aber auch Anwendungsfälle, wo eine Deklaration innerhalb der main-Funktion sinnvoll ist

Die Deklaration der einzelnen structs erfolgt dann innerhalb der main-Funktion

```c
#include <stdio.h>

struct studentData {
    char firstName[20];
    char lastName[20];
    unsigned int studentID;
};

int main(){
    // Struktur vom Typ studentData mit
    // dem Strukturnamen student1
    struct studentData student1;

    strcpy(student1.firstName, "Maike");
    strcpy(student1.lastName, "Stern");
    stern.studentID = 1234;

    return 0;
}
```

Structs
Nested structures / verschachtelte Strukturen

Struct für die Uhrzeit, mit Integervariablen für Stunden und Minuten

Struct, die ein Character-Array sowie eine struct vom Typ time als Variablen hat

Deklaration einer Struct vom Typ meeting namens dataScienceMeetup. Die Struct wird direkt initialisiert mit den Variablenwerten. Die Variablenwerte der Struct time werden in geschweiften Klammern übergeben

Zugriff auf die Werte erfolgt über den Punktoperator

```c
typedef struct Time {
  unsigned int hour;
  unsigned int minutes;
} Time;

typedef struct Meeting {
  char name[10];
  Time time;
} Meeting;

int main() {

  Meeting dataScienceMeetup = {"Data Science MeetUp", {18, 30}};

  printf("The %s is at %d:%d", dataScienceMeetup.name,
         dataScienceMeetup.time.hour,
         dataScienceMeetup.time.minutes);

  return 0;
}
```

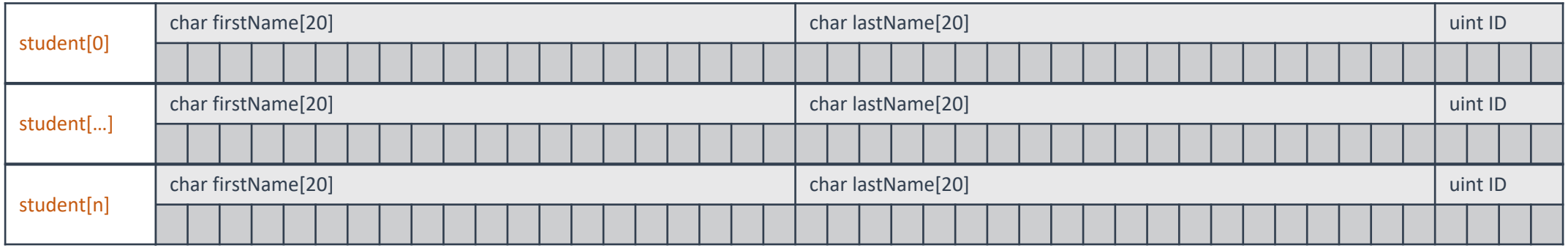
Structs
Arrays von Strukturen

Genau wie primitive Variablen können mehrere Structs in einem Array gesammelt werden

```
typedef struct Student{
    char firstName[20];
    char lastName[20];
    unsigned int studentID;
    } Student;
```

```
Student student[100];
```

Im Speicher wird Platz für 100 Strukturen vom Typ students reserviert



Structpointer

Zugriff auf Structvariablen, wenn die Struct als Variable deklariert wurde:

- student.firstName;

Zugriff auf Structvariable per Pointer:

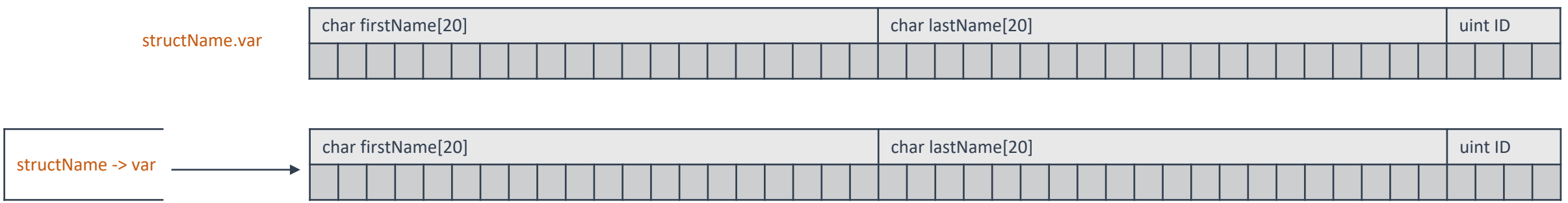
- student -> firstName;

```c
typedef struct studentData{
    char firstName[20];
    char lastName[20];
    unsigned int studentID;
} studentData;

int main(){

    studentData studentVar;
    studentData *studentPointer;

    return 0;
}
```



Enumeration

Enumerationen dienen dazu, Konstanten einen Namen zuzuweisen

- Syntax:
  typedef enum name {const1, const2};

- Beispiel
  typedef enum Week {monday = 1, tuesday, wednesday, thursday, friday, saturday, sunday};

- Deklaration im Code
  Week day = wednesday;

Unions (Varianten)

Unions sind eine Sonderform der Structs und eine speicherplatzsparende Möglichkeit, Daten zu strukturieren

- Von der Syntax her sind Structs und Unions gleich
- Unions reservieren jedoch immer nur Speicherplatz vom Umfang der größten Variable (im Beispiel also Speicherplatz für eine Double-Variable)
- Das heißt, in einer Union teilen sich die Variablen einen Speicherplatz
- Daher kann auch immer nur eine Union-Variable verwendet werden, im Beispiel entweder die Integer-Variable oder die Double-Variable. Die Union-Variablen sind also exklusiv verwendbar
- Das ist praktisch, wenn der Speicherplatz knapp ist oder die exklusive Verwendung sinnvoll ist (Bsp.: verschachtelte Union mit Structs)

```
typedef union typeName {
  int var1;
  double var2;
  ...
} alias;



int main(){
 alias  unionName;
...}
```

In dieser Vorlesung

Übungsklausur

- Unter "Realbedingungen": 1,5 Stunden & auf Papier
- Wer fertig ist, bitte einen Kaffee / Tee / Glühwein trinken gehen
- Wir treffen uns wieder um: _____ und besprechen die Lösung
- Ich lade die Lösung auch auf ELO hoch

Nächste Vorlesung

- Vortrag von
  - Dr. Daniel Grünbaum (Gruppenleiter Data Science ams OSRAM)
  - Alexander Luce (Doktorand Simulation & Modelling ams OSRAM, Max-Planck-Institut für die Physik des Lichts)
- Themen:
  - Wie schreibt man besseren Code
  - Software Engineering & Tools, die einem das Leben einfacher machen / die man kennen muss
  - Bißchen künstliche Intelligenz