Structs

Alexander Helbok, Raphael Riener, Anna Bozukov, Lorenz Ritsch

13. Juni 2022

Überblick

Einleitung

Was sind Structs und warum brauchen wir sie?

Definition

```c
struct person{
    char name[50];
    int age;
    float height;
};

int main(){
    struct person Alex, Anna;
}
```

```c
struct person{
    char name[50];
    int age;
    float height;
} Alex, Anna;
```

Initialisierung

```c
// Struct declaration

int main(){
    Alex.age = 19;
    Anna.height = 1.85;
    // Ausgabe
    printf("Alter = %d\n", Alex.age);
}
```

```c
// Struct declaration

int main(){
    struct person Alex = {.height = 1.7, .age = 19};
    struct person Anna = {"Anna", 20, 1.6};
    // Ausgabe
    printf("Alter = %d\n", Alex.age);
}
```

Nested Structs

```c
struct person{
   char name[50];
   int age;
   float height;
};

struct Family{
   struct person;
   int number_Sisters;
   int number_Brothers;
} myFamily;

int main(){
   struct person Alex = {.age = 19, .height = 1.7};
   struct Family myFamily = {Alex, .number_Sister = 2};
   printf("Height = %f\n", myFamily.Alex.height);
}
```

Structs und Funktionen

```c
// Struct declaration

struct person avgPerson(struct person p1, struct person p2);

int main(){
    struct person Alex = {.age = 19, .height = 1.7};
    struct person Anna = {.age = 20, .height = 1.6};
    struct person Avg = avgPerson(Alex, Anna);
}

struct person avgPerson(struct person p1, struct person p2){
    struct person temp;
    temp.age = (p1.age + p2.age)/2;
    temp.height = (p1.height + p2.height)/2;
    return temp;
}
```

Structs und Pointer

```c
// Struct declaration

void aging(struct person *p1);

int main(){
    struct person Alex = {.age = 19, .height = 1.7};
    aging(&Alex);
}

void aging(struct person *p1){
    p1->age += 50;
    (*p1).height -= 0.03;
}
```

Motivation

1. Datentypen mischen
2. Strukturierter als Arrays
3. Vereinfachte nachträgliche Änderung
4. Benennung der Felder macht Abfrage einfacher
5. Verschachteln

Bsp Rakete

Reichweite und Flugdauer einer Rakete

mit benutzerdefinierten Werten für

Treibstoffmasse, Masseverlustrate usw.

Bsp Rakete

```c
struct rakete {
   double m;      // Rakete Leermasse
   double v;      // Rakete Geschwindigkeit

   double mTank;   // Treibstoff Masse
   double vTank;   // Treibstoff Geschwindigkeit
   double dmTank;   // Treibstoff Masseverlustrate

   double s;      // Zurückgelegte Strecke
   double t;      // Vergangene Zeit
};
```

```c
void masseschritt(struct rakete * rocket, double delta_m) {
   rocket->v += rocket->vTank * delta_m/(rocket->m + rocket->mTank);
   rocket->mTank -= delta_m;
}
```

```c
void zeitschritt(struct rakete * rocket, double delta_t) {
   rocket->s += rocket->v * delta_t;
   rocket->t += delta_t;
}
```

Bsp Rakete

```c
struct rakete prototyp = {.v = 0, .s = 0, .t = 0,
   // restliche variablen via argv[] zuweisen
};
```

```c
struct rakete rocket = prototyp;
double delta = 1e-5;
```

```c
while (rocket.mTank > 0) {
   masseschritt(&rocket, delta);
   zeitschritt(&rocket, delta/rocket.dmTank);
}
```

```c
printf("Endgeschwindigkeit %f erreicht nach: %f\n", rocket.v, rocket.t);
```

Danke für Ihre Aufmerksamkeit!