Imperative Programmierung

Übung 5: Funktionen und Sichtbarkeit

Justin Kreikemeyer
Informatik, Uni Rostock

Fragen?

Leitfragen

- Was ist eine Prozedur/Funktion?
- Wie strukturiere ich meinen Code mit Funktionen?
- Was ist die Sichtbarkeit einer Variablen?
- Was ist die Lebensdauer einer Variablen?

Empfehlung

- Programmieren lernt man nur durch selber Programmieren!
- Löst auch nach der Übung weiter so viele Programmieraufgaben, wie irgendwie möglich ($\rightarrow$ Praktikum/Selbststudium)
- Hier: Überblick (sehr viel auf einmal!), Lösen einiger Aufgaben, Klärung von Fragen

Erinnerung: Grundstruktur

```c
#include <stdio.h>                    // Inklusion von externem Quellcode
int main(int argc, char* argv[]) {    // <- (Haupt-)Funktion!

  // Programmcode steht hier.         // Zwei Schrägstriche (//) kennzeichnen Kommentar

  return 0;                           // RÜckgabewert an das Betriebsystem (0 = Erfolg)
}
```

- Beobachtung: Farbliche Markierung von speziellen Wörtern in C (je nach Editor anders)
- `#include <stdio.h>`: Einbinden des Quelltextes aus der Datei `stdio.h` (beinhaltet Prozeduren zur Ein- und Ausgabe)
- `argc, argv` Anzahl und Werte der Kommandozeilenargumente (nächste Woche)
- Warum das alles so aussieht und **was genau** `int main(...)` **bedeutet klärt sich** ~~im Laufe des Semesters~~ **heute**

C Programme Bisher

```c
#include <stdio.h>
int main(int argc, char* argv[]) {
  int x = 0;
  scanf("%d", &x);
  if (x < 0) {
    for (int i = 0; i < 10; i++) {
      printf("%d\n", i);
    }
  } else {
    for (int i = -10; i < 0; i++) {
      printf("%d\n", i);
    }
  }
  return 0;
}
```

Was gibt das obige Programm aus? Was fällt auf?

C Programme Bisher

```c
#include <stdio.h>
int main(int argc, char* argv[]) {
  int x = 0;
  scanf("%d", &x);
  if (x < 0) {
    for (int i = 0; i < 10; i++) {       // Das ist ja fast dasselbe...
      printf("%d\n", i);
    }
  } else {
    for (int i = -10; i < 0; i++) {      // ...wie hier!
      printf("%d\n", i);
    }
  }
  return 0;
}
```

Irgendwie ist da was doppelt!

Funktion/Prozedur

- Zusammenfassung von (parametrisiertem) Codeblock für die Wiederverwendung
- Funktion: mit Rückgabe eines Wertes; Prozedur: Nur “Effekt”
- Parameter können beim Aufruf angepasst werden $\rightarrow$ Abstraktion!
- Vorteil: Dinge nur einmal hinschreiben
- Vorteil: Trennung der Aufgaben durch Zerlegung in Einzelteile
- Vorteil: Programm leichter zu lesen

Prozedur: am Beispiel

```c
#include <stdio.h>
void range_inc(int start, int end) {
    for (int i = start; i < end; i++) {
      printf("%d\n", i);
    }
}
int main(int argc, char* argv[]) {
  int x = 0;
  scanf("%d", &x);
  if (x < 0) {
    range_inc(0, 10);
  } else {
    range_inc(-10, 0);
  }
  return 0;
}
```

Definition einer Funktion in C

- `<Rückgabetyp> <name> ( <parameter> ) { <anweisungen> }`
- Rückgabetyp: Datentyp wie `int`, `float`, `...` oder `void` (keine Rückgabe)
- Name: Aussagekräftige knappe Beschreibung der Funktion
- Parameter: Liste von `<typ> <name>, <typ> <name>, ...`
- Rückgabe: mittels `return` wird das Ergebnis an den Punkt des Aufrufs zurück gegeben (bei Prozedur: (vorzeitiges) Beenden)
- `int main(...)` → spezielle Funktion, die Wert an Betriebssystem zurück gibt

```c
int add(int a, int b) { // "Funktionskopf"
  int summe = a + b;    // Berechnungen / Seiteneffekte
  return summe;         // Rückgabe des Ergebnisses
}
```

Funktionen müssen **vor** dem ersten Aufruf definiert werden!

Gemeinsames Beispiel: Gerade Zahlen (Theorie)

Woraus bestehen die Bestandteile einer Funktion, die für eine gegebene Zahl überprüft, ob diese gerade ist?

- Rückgabetyp?
- Name?
- Parameter?
- Rückgabe?

Gemeinsames Beispiel: Gerade Zahlen (Praxis)

```
int istGerade(int zahl) {
  if (zahl % 2 == 0) {
    return 1;
  }
  return 0;
}
```

Geht das auch kürzer?

Sichtbarkeit

- **Sichtbarkeit**: Wo ist ein Variablenname bekannt?
- $\rightarrow$ innerhalb des Bereiches, in dem Sie definiert ist; Bereich = Block, i.e., $\{\dots\}$
- Frage: Habt ihr schonmal einen Block gesehen? Wo treten sie auf?
- **Lebensdauer**: Wie lange kann auf den Wert der Variable zugegriffen werden?
- $\rightarrow$ Solange sie sichtbar ist; (Genauer: solange der Speicher nicht freigegeben wurde; dazu später mehr).
- Vermeidung von Fehlern: Variablennamen nicht “nah beieinander” doppelt definieren!
- Jetzt: kurzes Quiz

Sichtbarkeit: Quiz I

```c
int main() {
  int x = 0;
  {
    x = 6;
    printf("%d\n", x);
  }
  printf("%d\n", x);
  return 0;
}
```

Was ist die Ausgabe des obigen Programms?

Sichtbarkeit: Quiz II

```c
int main() {
  int x = 0;
  {
    int x = 6;
    printf("%d\n", x);
  }
  printf("%d\n", x);
  return 0;
}
```

Was ist die Ausgabe des obigen Programms?

Sichtbarkeit: Quiz III

```c
int foo(int x) {
  x = 6;
  return x;
}
int main() {
  int x = 0;
  {
    int x = 6;
    printf("%d\n", x);
  }
  printf("%d\n", x);
  foo(x);
  printf("%d\n", x);
  return 0;
}
```

Was ist die Ausgabe des obigen Programms?

Sichtbarkeit: Quiz IV

```c
int foo(int x) {
  x = 6;
  return x;
}
int main() {
  int x = 0;
  {
    int x = 6;
    printf("%d\n", x);
  }
  printf("%d\n", x);
  x = foo(x);
  printf("%d\n", x);
  return 0;
}
```

Was ist die Ausgabe des obigen Programms?

Sichtbarkeit: Quiz V

```c
int foo(int x) {
  int y = 6;
  return x;
}
int main() {
  int z = 0;
  if (z < 0) {
    int u = 6;
  } else {
    int v = 7;
  }
  z = foo(z);          // ...ist u hier sichtbar?
  printf("%d\n", z);   // ...ist y hier sichtbar?
  return 0;            // ...lebt hier u noch?
}
```

Wo sind u, v, x, y und z sichtbar? Wo endet ihre Lebensdauer?

Fragen?

Aufgaben: Funktionen in C

Schreiben Sie die folgenden C-Programme und Funktionen[1]! Nutzen Sie dazu die Konzepte aus dieser Übung und das Cheat Sheet!
Bearbeitungszeit: bis 10 Minuten vor Schluss. Dann Besprechung von häufigen Problemen.

[1] Weitere Aufgaben können jederzeit beim Übungsleiter erfragt werden.

Programme I

EvenRange Schreiben Sie ein Programm, dass zwei ganze Zahlen einliest und alle **geraden** ganzen Zahlen dazwischen (inklusive Grenzen) ausgibt. Für das Überprüfen, ob eine Zahl eine gerade ist, nutzen Sie einen Aufruf einer für diesen Zweck definierten Funktion!

MultiMin Schreiben Sie ein Programm welches eine Zahl n einliest. Darauf hin soll n-mal nach einer Eingabe gefragt werden und zum Schluss soll das Maximum, Minimum und der Mittelwert der Zahlen ausgegeben werden. Erweitern Sie ihr Programm so, dass beim Einlesen ein Text zur Eingabe der Zahl auffordert und abstrahieren Sie das Einlesen in einer Funktion, welche die eingelesene Zahl zurückgibt!

Programme II

Factorial Schreiben Sie eine Funktion, die $n!$ (n Fakultät) berechnet! Für Eingaben $n \leq 1$ soll diese den Wert 1 zurückgeben.

Square Schreiben Sie eine Funktion `int countSquareNumbers(int a, int b)`, die die Anzahl der Quadratzahlen zwischen a und b bestimmt und zurückgibt. Der Funktion werden dafür zwei natürliche Zahlen $1 \leq a \leq b$) übergeben. Testen Sie Ihre Funktion für verschiedene Eingaben in der Main-Funktion.

Prime Schreiben Sie eine Funktion `int isPrime(int x)`, die 1 ausgibt, falls x eine Primzahl ist und 0 sonst.