Human-Computer Interaction Group

Leibniz Universität Hannover

Programmieren 1 - WS 2022/23

Prof. Dr. Michael Rohs, Tim Dünte, M.Sc., Jan Feuchter, M.Sc.

Präsenzübung 9

Diese Aufgaben sind zur Lösung während der einstündigen Präsenzübung gedacht. Sie können die Aufgaben auf einem mitgebrachten Laptop oder auf Papier lösen.

Aufgabe 1: Zeigerarithmetik

Gegeben sei: `int a[] = { 10, 20, 30, 40, 50 };`
Das Array a beginne an Adresse 1000. Außerdem sei definiert:

```
int* p = &a[3];   int* q = &a[0];
char* c = (char*)p;   char* d = (char*)q;
```

Geben Sie die Werte der folgenden Ausdrücke an.

```
a) a
b) a + 1
c) a++
d) p - q
e) c – d
f) p > q
g) p > d
h) 1[a]
i) p[-1]
j) p[0]
k) p[1]
l) p[2]
m) *(q + 1)
n) (*q + 1)
```

Aufgabe 2: Wörter

Die Template-Datei für diese Aufgabe ist words.c. Sie enthält mehrere Funktionen zum Umgang mit in Zeichenketten enthaltenen Wörtern. Wörter sind voneinander durch Leerzeichen, Kommata bzw. Punkte getrennt.

a) Erläutern Sie die Funktionsweise von find_word_start und find_word_end. Funktionieren diese auch für den leeren String?
b) Warum wird in find_word_start die logische Und-Verknüpfung und in find_word_end die logische Oder-Verknüpfung verwendet? Ersetzen Sie in find_word_start die Und-Verknüpfungen durch geeignete Umformung durch Oder-Verknüpfungen.
c) Wie sind Wörter in words.c repräsentiert?
d) Erläutern Sie, wie mit print_word ein Wort, also ein Teil eines Strings, ausgegeben wird.

e) Implementieren Sie first_word und next_word unter Verwendung von find_word_start und find_word_end. In der main-Funktion ist die Verwendung dieser Funktionen in for-Schleifen gezeigt.

Aufgabe 3: Dynamisch allokierte Wörter

Die Template-Datei für diese Aufgabe ist wieder words.c. In dieser Aufgabe sollen Arrays von Wörtern allokiert werden.

a) Vervollständigen Sie die Implementierung von count_words zur Ermittlung der Anzahl der Wörter in einem String.
b) Implementieren Sie Word* get_words(/*in*/char* s, /*out*/int* n). Diese Funktion soll ein dynamisch allokiertes Array der Wörter einer Funktion zurückgeben. Zurückgegeben wird also ein Array mit Elementen vom Typ Word. Außerdem gibt die Funktion die Anzahl der Wörter im Parameter n zurück. Warum ist hierzu ein separater Parameter erforderlich?
c) Implementieren Sie nun als Variante void get_words2(/*in*/char* s, /*out*/Word** words, /*out*/int* n). get_words2 soll das gleiche leisten wie get_words, aber hat eine etwas andere Signatur. Erläutern Sie, warum hier Word** erforderlich ist. Welche Variante gefällt Ihnen besser?

Aufgabe 4 (optional): Dynamische Speicherverwaltung

Gegeben sei folgende Funktion.

```
int f(int a) {
    int* b = &a;
    int* c = xcalloc(4, sizeof(int));
    *(c + 1) = 2;
    *(c + 2) = 1;
    return *b + c[c[0]] + c[c[1]];
}
```

a) Wie lautet das Ergebnis des Funktionsaufrufs f(3)?
b) Wie viel Speicher wird von der Funktion dynamisch allokiert?
c) Behandelt die Funktion die Speicherverwaltung korrekt?
d) Befindet sich die Variable b auf dem Stack oder auf dem Heap?
e) Zeigt die Variable b auf einen Wert auf dem Stack oder auf dem Heap?
f) Befindet sich die Variable c auf dem Stack oder auf dem Heap?
g) Zeigt die Variable c auf einen Wert auf dem Stack oder auf dem Heap?
h) Vereinfachen Sie die Funktion so weit wie möglich.