

Hardwarenahe Programmierung

Übungsaufgaben – 7. Dezember 2023

Diese Übung enthält Punkteangaben wie in einer Klausur. Um zu „bestehen“, müssen Sie innerhalb von 70 Minuten unter Verwendung ausschließlich zugelassener Hilfsmittel 12 Punkte (von insgesamt 24) erreichen.

Aufgabe 1: Länge von Strings

Strings werden in der Programmiersprache C durch Zeiger auf **char**-Variable realisiert.

Beispiel: **char** *hello_world = "Hello,_world!\n"

Die Systembibliothek stellt eine Funktion **strlen()** zur Ermittlung der Länge von Strings zur Verfügung (**#include <string.h>**).

- (a) Auf welche Weise ist die Länge eines Strings gekennzeichnet? (1 Punkt)
- (b) Wie lang ist die Beispiel-String-Konstante "Hello,_world!\n", und wieviel Speicherplatz belegt sie? (2 Punkte)
- (c) Schreiben Sie eine eigene Funktion **int strlen (char *s)**, die die Länge eines Strings zurückgibt. (3 Punkte)

Wir betrachten nun die folgenden Funktionen (Datei: **aufgabe-1.c**):

```
int fun_1 (char *s)
{
    int x = 0;
    for (int i = 0; i < strlen (s); i++)
        x += s[i];
    return x;
}
```

```
int fun_2 (char *s)
{
    int i = 0, x = 0;
    int len = strlen (s);
    while (i < len)
        x += s[i++];
    return x;
}
```

- (d) Was bewirken die beiden Funktionen? (2 Punkte)
- (e) Schreiben Sie eine eigene Funktion, die dieselbe Aufgabe erledigt wie **fun_2()**, nur effizienter. (4 Punkte)
- (f) Von welcher Ordnung (Landau-Symbol) sind die beiden Funktionen hinsichtlich der Anzahl ihrer Zugriffe auf die Zeichen im String? Begründen Sie Ihre Antwort. Sie dürfen für **strlen()** Ihre eigene Version der Funktion voraussetzen. (3 Punkte)
- (g) Von welcher Ordnung (Landau-Symbol) ist Ihre effizientere Funktion? Begründen Sie Ihre Antwort. (1 Punkt)

Aufgabe 2: Einfügen in Strings (Ergänzung)

Diese Aufgabe ist eine Ergänzung von Aufgabe 3 der Übung vom 2. November 2023 um die Teilaufgaben (e), (f) und (g). Für den „Klausur-Modus“ können Sie die Teilaufgaben (a) bis (d) als „bereits gelöst“ voraussetzen.

Wir betrachten das folgende Programm ([aufgabe-2.c](#)):

```
#include <stdio.h>
#include <string.h>

void insert_into_string (char src, char *target, int pos)
{
    int len = strlen (target);
    for (int i = pos; i < len; i++)
        target[i+1] = target[i];
    target[pos] = src;
}

int main (void)
{
    char test[100] = "Hochshule_Bochum";
    insert_into_string ('c', test, 5);
    printf ("%s\n", test);
    return 0;
}
```

Die Ausgabe des Programms lautet: `Hochschhhhhhhhhhhhh`

- (a) Erklären Sie, wie die Ausgabe zustandekommt.
- (b) Schreiben Sie die Funktion `insert_into_string()` so um, daß sie den Buchstaben `src` an der Stelle `pos` in den String `target` einfügt.
Die Ausgabe des Programms müßte dann `Hochschule Bochum` lauten.
- (c) Was kann passieren, wenn Sie die Zeile `char test[100] = "Hochshule_Bochum";` durch `char test[] = "Hochshule_Bochum";` ersetzen? Begründen Sie Ihre Antwort.
- (d) Was kann passieren, wenn Sie die Zeile `char test[100] = "Hochshule_Bochum";` durch `char *test = "Hochshule_Bochum";` ersetzen? Begründen Sie Ihre Antwort.
- (e) Schreiben Sie eine Funktion `void insert_into_string_sorted (char src, char *target)`, die voraussetzt, daß der String `target` alphabetisch sortiert ist und den Buchstaben `src` an der alphabetisch richtigen Stelle einfügt. Diese Funktion darf die bereits vorhandene Funktion `insert_into_string()` aufrufen.
(4 Punkte)

Zum Testen eignen sich die folgenden Zeilen im Hauptprogramm:

```
char test[100] = "";
insert_into_string_sorted ('c', test);
insert_into_string_sorted ('a', test);
insert_into_string_sorted ('d', test);
insert_into_string_sorted ('b', test);
```

Danach sollte `test[]` die Zeichenfolge `"abcd"` enthalten.

- (f) Wie schnell (Landau-Symbol in Abhängigkeit von der Länge n des Strings) arbeitet Ihre Funktion `void insert_into_string_sorted (char src, char *target)`? Begründen Sie Ihre Antwort. (1 Punkt)
- (g) Beschreiben Sie – in Worten oder als C-Quelltext –, wie man die Funktion `void insert_into_string_sorted (char src, char *target)` so gestalten kann, daß sie in $\mathcal{O}(\log n)$ arbeitet. (3 Punkte)

Viel Erfolg!