Einführung in die Programmierung, WS 2016/2017
Blatt 7

Simon Hangl, Sebastian Stabinger, Alex Hirsch

2016–12–06

- Abgabe bis spätestens Montag 21:59 über OLAT (`https://lms.uibk.ac.at/olat/dmz/`).
- Bereiten Sie jede Aufgabe so vor, dass Sie Ihre Lösung im Proseminar präsentieren können!
- Benennen Sie Ihre Abgabe nach folgendem Schema: *Gruppennummer*-*Nachname*-`blatt`*Übungsblattnummer*`.tar.gz` Wenn Sie also Max Mustermann heißen und Gruppe 1 besuchen, heißt die Datei von Übung 7: `1-mustermann-blatt7.tar.gz`
- Compilieren Sie alle Programme mit den Optionen `-Wall -Werror -std=c99`

Feedback

Nutzen Sie die angebotenen Möglichkeiten, uns Feedback zu geben (eMail, Tutorium, Proseminar). Hier können Sie uns auf Probleme, notwendige Stoffwiederholungen, Unklarheiten, aber auch positive Dinge, die beibehalten werden sollten, hinweisen.

Testen und Dokumentation

Stellen Sie sicher, dass alle Lösungen fehlerfrei kompilieren. Testen Sie Ihre Lösungen ausführlich (z.B. auf falsche Eingaben, falsche Berechnungen, Sonderfälle) und dokumentieren Sie sie. Dies hilft Ihnen bei der Präsentation und uns beim Nachvollziehen Ihrer Entscheidungen. Lesen Sie die Aufgaben *vollständig* durch.

Aufgabe 1 (4 Punkte)

Gegeben ist ein Array (`list`) der Größe `100` (`capacity`), sowie eine Variable (`length`) welche beschreibt wie viele Elemente sich aktuell in der Liste befinden.

Folgende Funktionen sollen von Ihnen implementiert werden (verwenden Sie den unten angegebenen Quellcode als Vorlage und ändern sie die Signaturen der Funktionen nicht):

`list_get` Gibt den Wert an der Stelle `index` zurück, falls `index` außerhalb der Liste liegt, soll `0` zurückgegeben werden.

`list_push_back` Hängt eine neues Element mit dem Wert `value` an das Ende der Liste an (sofern noch die Liste noch nicht voll ist). Gibt `true` / `false` zurück je nachdem ob das neue Element in die Liste eingefügt werden konnte oder nicht.

`list_pop_back` Entfernt das letzte Element einer Liste und gibt dieses zurück. Falls die Liste leer ist soll `0` zurückgegeben werden.

`list_print` Gibt den Inhalt der Liste auf dem Bildschirm aus. Eine Liste welche die Zahlen 1, 2 und 3 enthält soll z.B. folgendermaßen ausgegeben werden: `[1,2,3]`

Testen Sie Ihre Implementierung ausführlich. Berücksichtigen Sie im speziellen Randfälle wie eine leere, oder eine volle Liste.

```c
#define CAPACITY 100

int list_get(int list[], int length, int index) { /* TODO */ }

bool list_push_back(int list[], int* length, int capacity, int value) {
    /* TODO */
}

int list_pop_back(int list[], int* length) { /* TODO */ }

int list_print(int list[], int length) { /* TODO */ }

int main(void) {
    int list[CAPACITY] = {0};
    int length = 0;
    /* TODO Testing */
    return EXIT_SUCCESS;
}
```

Hinweis: *Abgabe: 1-mustermann-a1.c*

Aufgabe 2 (4 Punkte)

Implementieren Sie eine Funktion, welche eine Array sowie seine Länge entgegen nimmt und es sortiert. Sie können einen beliebigen Sortieralgorithmus (z.B. Bubble sort[1])verwenden.

```
void sort(int array[], int length) { /* TODO */ }
```

- Können Sie diese Funktion verwenden um die Liste aus Aufgabe 1 zu sortieren?

Hinweis: *Abgabe: 1-mustermann-a2.c*

Aufgabe 3 (4 Punkte)

- Erklären Sie wie sich der Operator `sizeof` bei Pointern und bei Arrays verhält.
- Wieso könnte die folgende Funktion eine schlechte Idee sein?

```
void length(int array[]) {
  return sizeof(array) / sizeof(int);
}
```

- Ist das Ergebnis des folgenden Vergleichs `true` oder `false`? Erklären Sie ihre Antwort.

```
int* ptr1 = 0x00;
char* ptr2 = 0x08;
ptr1 + 3 == ptr2 + 4;
```

- Warum ist die folgende Funktion keine gute Idee?

```
int* new_int(void) {
  int my_new_int = 0;
  return &my_new_int;
}
```

Hinweis: *Abgabe: 1-mustermann-a3.txt*

[1] `https://en.wikipedia.org/wiki/Bubble_sort`