

Hardwarenahe Programmierung / Angewandte Informatik

Übungsaufgaben – 22. Januar 2018

Diese Übung enthält Punkteangaben wie in einer Klausur. Um zu „bestehen“, müssen Sie innerhalb von 65 Minuten unter Verwendung ausschließlich zugelassener Hilfsmittel 11 Punkte (von insgesamt 23) erreichen.

Aufgabe 1: Stack-Operationen

Wir betrachten das folgende Programm ([aufgabe-1.c](#)):

```
#include <stdio.h>

#define STACK_SIZE 10

int stack[STACK_SIZE];
int stack_pointer = 0;

void push (int x)
{
    stack[stack_pointer++] = x;
}

int pop (void)
{
    return stack[--stack_pointer];
}

void show (void)
{
    printf ("stack_content:");
    for (int i = 0; i < stack_pointer; i++)
        printf ("_%d", stack[i]);
    if (stack_pointer)
        printf ("\n");
    else
        printf ("_(empty)\n");
}

void insert (int x, int pos)
{
    for (int i = pos; i < stack_pointer; i++)
        stack[i + 1] = stack[i];
    stack[pos] = x;
    stack_pointer++;
}

void insert_sorted (int x)
{
    int i = 0;
    while (i < stack_pointer && x < stack[i])
        i++;
    insert (x, i);
}

int main (void)
{
    push (3);
    push (7);
    push (137);
    show ();
    insert (5, 1);
    show ();
    insert_sorted (42);
    show ();
    insert_sorted (2);
    show ();
    return 0;
}
```

- (a) Ändern Sie das Programm so, daß die Funktion `insert()` ihren Parameter `x` an der Stelle `pos` in den Stack einfügt, und den sonstigen Inhalt des Stacks verschiebt, aber nicht zerstört. (3 Punkte)
- (b) Ändern Sie das Programm so, daß die Funktion `insert_sorted()` ihren Parameter `x` an derjenigen Stelle einfügt, an die er von der Sortierung her gehört. (Der Stack wird hierbei vor dem Funktionsaufruf als sortiert vorausgesetzt.) (2 Punkte)
- (c) Schreiben Sie eine zusätzliche Funktion `int search (int x)`, die die Position (Index) des Elements `x` innerhalb des Stack zurückgibt – oder `-1`, wenn `x` nicht im Stack enthalten ist. Der Rechenaufwand darf höchstens $\mathcal{O}(n)$ betragen. (3 Punkte)
- (d) Wie (c), aber der Rechenaufwand darf höchstens $\mathcal{O}(\log n)$ betragen. (4 Punkte)

Aufgabe 2: Iterativer Floodfill

In Übungsaufgabe 3 vom 11. 12. 2017 (siehe hp-uebung-20171211.pdf) haben wir einen rekursiven Floodfill-Algorithmus implementiert.

- (a) Ergänzen Sie die Funktion `fill()` so, daß Sie verfolgen können („Animation“), in welcher Reihenfolge die Punkte auf dem Bildschirm „ausgemalt“ werden. (4 Punkte)

Hinweis: Eine Möglichkeit, den Bildschirm zu löschen und eine kurze Zeit lang zu warten, können Sie den Beispielaufgaben zum Sortieren (z. B. `sort-2.c`) entnehmen. (Um `usleep()` nutzen zu können, ist beim Compilieren mit `gcc` evtl. die Option `-std=gnu99` erforderlich.)

- (b) Unter Verwendung eines Stack ist es möglich, den Floodfill-Algorithmus iterativ (also mit Schleife) statt rekursiv zu implementieren.

Informieren Sie sich per Web-Suche über die Details und implementieren Sie einen iterativen Floodfill-Algorithmus mit Stack für die Text-Grafik-Bibliothek. Die Punkte sollen dabei in derselben Reihenfolge „ausgemalt“ werden wie bisher in der rekursiven Floodfill-Implementation. (4 Punkte)

- (c) Ersetzen Sie nun den Stack durch einen FIFO. Der Floodfill-Algorithmus sollte weiterhin funktionieren, nur daß die Punkte in einer anderen Reihenfolge „ausgemalt“ werden. Beschreiben Sie diesen Unterschied. (3 Punkte)

(Ein iterativer Floodfill mit FIFO ist nützlich für die Wegfindung und verwandte Aufgabenstellungen.)

Viel Erfolg!