

Hardwarenahe Programmierung

Übungsaufgaben – 5. Dezember 2022

Diese Übung enthält Punkteangaben wie in einer Klausur. Um zu „bestehen“, müssen Sie innerhalb von 85 Minuten unter Verwendung ausschließlich zugelassener Hilfsmittel 15 Punkte (von insgesamt 30) erreichen.

Aufgabe 1: Objektorientierte Tier-Datenbank (Ergänzung)

Diese Aufgabe ist eine Neuauflage der Übungsaufgabe 3 vom 28. November 2022, ergänzt um die Teilaufgaben (e) bis (g).

Das auf der nächsten Seite dargestellte Programm (Datei: [aufgabe-1a.c](#)) soll Daten von Tieren verwalten.

Beim Compilieren erscheinen die folgende Fehlermeldungen:

```
$ gcc -std=c99 -Wall -O aufgabe-1a.c -o aufgabe-1a
aufgabe-1a.c: In function 'main':
aufgabe-1a.c:31: error: 'animal' has no member named 'wings'
aufgabe-1a.c:37: error: 'animal' has no member named 'legs'
```

Der Programmierer nimmt die auf der nächsten Seite in Rot dargestellten Ersetzungen vor (Datei: [aufgabe-1b.c](#)). Daraufhin gelingt das Compilieren, und die Ausgabe des Programms lautet:

```
$ gcc -std=c99 -Wall -O aufgabe-1b.c -o aufgabe-1b
$ ./aufgabe-1b
A duck has 2 legs.
Error in animal: cow
```

- (a) Erklären Sie die o. a. Compiler-Fehlermeldungen.
- (b) Wieso verschwinden die Fehlermeldungen nach den o. a. Ersetzungen?
- (c) Erklären Sie die Ausgabe des Programms.
- (d) Beschreiben Sie – in Worten und/oder als C-Quelltext – einen Weg, das Programm so zu berichtigen, daß es die Eingabedaten ("A duck has 2 wings. A cow has 4 legs.") korrekt speichert und ausgibt.
- (e) Schreiben Sie das Programm so um, daß es keine expliziten Typumwandlungen mehr benötigt.
Hinweis: Verwenden Sie **union**. (4 Punkte)
- (f) Schreiben Sie das Programm weiter um, so daß es die Objektinstanzen **duck** und **cow** dynamisch erzeugt.
Hinweis: Verwenden Sie **malloc()** und schreiben Sie Konstruktoren. (4 Punkte)
- (g) Schreiben Sie das Programm weiter um, so daß die Ausgabe nicht mehr direkt im Hauptprogramm erfolgt, sondern stattdessen eine virtuelle Methode **print()** aufgerufen wird.
Hinweis: Verwenden Sie in den Objekten Zeiger auf Funktionen, und initialisieren Sie diese in den Konstruktoren. (4 Punkte)

```

#include <stdio.h>

#define ANIMAL 0
#define WITH_WINGS 1
#define WITH_LEGS 2

typedef struct animal
{
    int type;
    char *name;
} animal;

typedef struct with_wings
{
    int wings;
} with_wings;

typedef struct with_legs
{
    int legs;
} with_legs;

int main (void)
{
    animal *a[2];

    animal duck;
    a[0] = &duck;
    a[0]—>type = WITH_WINGS;
    a[0]—>name = "duck";
    a[0]—>wings = 2;  ← ((with_wings *) a[0])—>wings = 2;

    animal cow;
    a[1] = &cow;
    a[1]—>type = WITH_LEGS;
    a[1]—>name = "cow";
    a[1]—>legs = 4;  ← ((with_legs *) a[1])—>legs = 4;

    for (int i = 0; i < 2; i++)
        if (a[i]—>type == WITH_LEGS)
            printf ("A_%s_has_%d_legs.\n", a[i]—>name,
                    ((with_legs *) a[i])—> legs);
        else if (a[i]—>type == WITH_WINGS)
            printf ("A_%s_has_%d_wings.\n", a[i]—>name,
                    ((with_wings *) a[i])—> wings);
        else
            printf ("Error_in_animal:_%s\n", a[i]—>name);

    return 0;
}

```

Aufgabe 2: Iterationsfunktionen

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

```

#include <stdio.h>

void foreach (int *a, void (*fun) (int x))
{
    for (int *p = a; *p >= 0; p++)
        fun (*p);
}

void even_or_odd (int x)
{
    if (x % 2)
        printf ("%d_ist_ungerade.\n", x);
    else
        printf ("%d_ist_gerade.\n", x);
}

int main (void)
{
    int numbers[] = { 12, 17, 32, 1, 3, 16, 19, 18, -1 };
    foreach (numbers, even_or_odd);
    return 0;
}

```

- Was bedeutet **void (*fun) (int x)**, und welchen Sinn hat seine Verwendung in der Funktion **foreach()**? (2 Punkte)
- Schreiben Sie das Hauptprogramm **main()** so um, daß es unter Verwendung der Funktion **foreach()** die Summe aller positiven Zahlen in dem Array berechnet. Sie dürfen dabei weitere Funktionen sowie globale Variable einführen. (4 Punkte)

Aufgabe 3: Stack-Operationen

Das folgende Programm ([aufgabe-3.c](#)) implementiert einen Stapelspeicher (Stack). Dies ist ein Array, das nur bis zu einer variablen Obergrenze (Stack-Pointer) tatsächlich genutzt wird. An dieser Obergrenze kann man Elemente hinzufügen (push).

In dieser Aufgabe sollen zusätzlich Elemente in der Mitte eingefügt werden (insert). Die dafür bereits existierenden Funktionen `insert()` und `insert_sorted()` sind jedoch fehlerhaft.

```
#include <stdio.h>

#define STACK_SIZE 10

int stack[STACK_SIZE];
int stack_pointer = 0;

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

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) Korrigieren 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) Korrigieren 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 die Zahl `-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)

Viel Erfolg!