Haute école d'ingénierie et d'architecture Fribourg
Hochschule für Technik und Architektur Freiburg

Embedded Systeme 1 & 2

Klassen T-2/I-2 // 2018-2019

a.07 - C - Die Zeiger

Übung 1

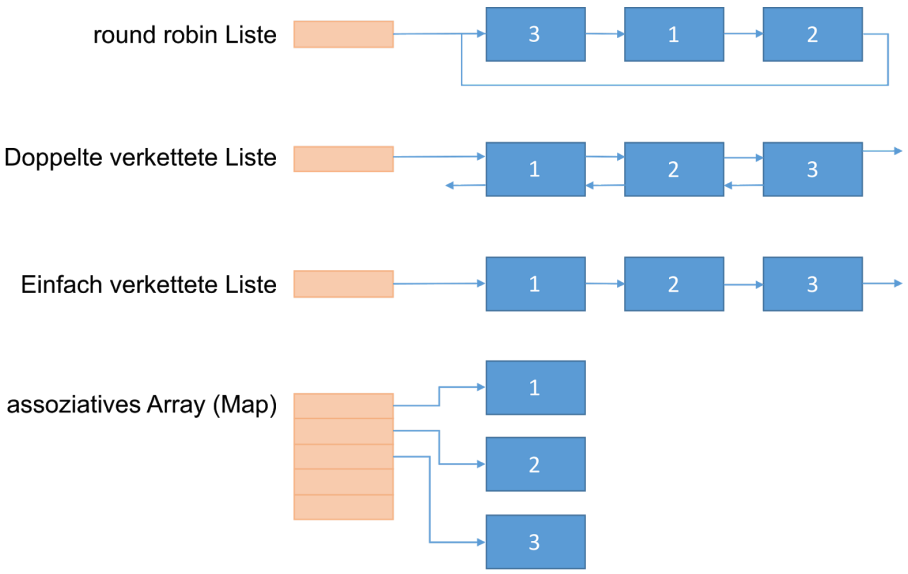
Geben Sie die Zeigerwerte für die nachstehenden Ausdrücke an:

```
struct a_struct {int a; int b; int c;}; // int is 32 bits long
/* 1 */ struct a_struct* pa = (struct a_struct*)1000;
/* 2 */ struct a_struct* pb = &pa[10]
/* 3 */ struct a_struct* pc = pa + 10;
/* 4 */ struct a_struct* pd = pc--;
/* 5 */ struct a_struct* pe = ++pb;
```

Übung 2

Entwickeln Sie ein Programm, das erlaubt, für jeden der Typen der nachstehenden verketteten Listen Objekte von folgendem Typ zu **erzeugen**, zu **löschen**, **einzufügen**, zu **suchen** und zu **extrahieren**:

```
struct element {
    int key;
    int value;
};
```



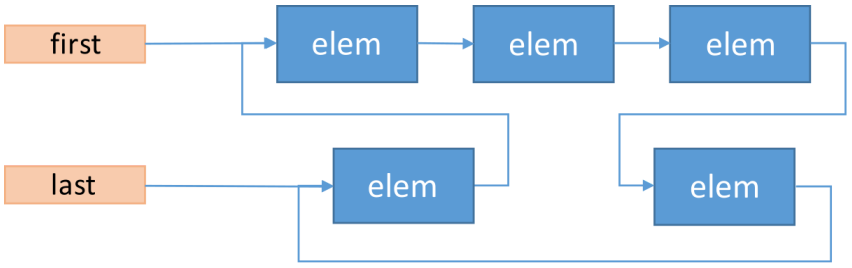
Nennen Sie die Vor- und Nachteile dieser Listentypen. Nennen Sie andere Darstellungen.

Übung 3

Gegeben seien die beiden folgenden Strukturen:

```
struct elem {
    char name;
    struct elem* next;
}

struct container {
    struct elem* first;
    struct elem* last;
}
```

und die verkettete Liste, die dem folgenden Schema entspricht:



codieren Sie die folgenden Funktionen in C:

a) Funktion für das Zählen der Anzahl Elemente:

```
/**
 * @param container points to the container structure
 * @return number of elements
 */

int get_nb_of_eles (const struct container* container);
```

b) Funktion, die ein Element entnehmen kann, ohne die Kette zu unterbrechen:

```
/**
 * @param container pointeur to the container structure
 * @param index position in the chain
 * @return pointer to the removed element, else 0
 */

struct elem* delete (struct container* container, int index);
```

Übung 4

Betrachten Sie die folgende Struktur:

```
struct node {
    unsigned char ident;
    long          size;
    struct node*  next;
};
```

Diese Datenstrukturen können einfach miteinander verkettet werden. Das Ende der Kette wird durch das Feld pnode auf **`null`** (0) identifiziert.

Schreiben Sie eine C Funktion, die die Aufgabe hat, die Struktur zu finden, deren Feld "ident" (das eindeutig ist) gleich dem Parameter "id" ist. Diese Funktion muss die folgende Schnittstelle entsprechen:

```
/**
 * @param list point to the first element of the chain
 * @param id identificator of the element to search
 *           (ident field of the structure)
 * @return reference to the found structure if successful,
 *         null (0) if no element could be found.
 */

struct node* search (struct node *list, unsigned char id);
```

Anmerkung: Wenn die Kette leer ist, soll der Parameter `liste` **`null`** (0) sein.

Übung 5

Gegeben seien die folgenden Variablendeklarationen und Initialisierungen:

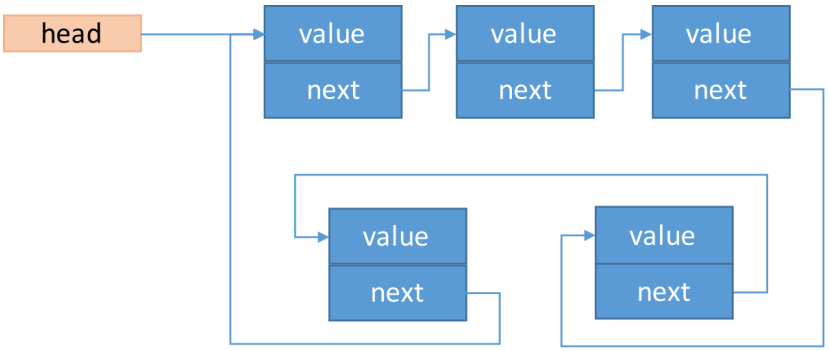
```
#define ARRAY_SIZE(x) (sizeof(x)/sizeof(x[0]))

char str1[]="Programmierung";
char str2[]="Algorithmik";
char str3[]="Mikroprozessoren";
char str4[]="Physik";

char* tab[] = {str1, str2, str3, str4}; char** ptab = tab;
```

```c
int main() {
    char* max = max_str(ptab, ARRAY_SIZE(tab));
    printf ("longest string=%s\n", max);
    return 0;
}
```

Codieren Sie die Funktion "`max_str`" in C, die in der Lage ist, die längste in der Tabelle "`tab`" enthaltene Zeichenkette (ihre Adresse) zurückzugeben.

Übung 6

Gegeben sei folgende Kette:



Sowie die folgenden Deklarationen:

```c
struct elem {
    short  value;
    struct elem* next;
};

struct result {
    short  max;
    short  min;
};
```

Schreiben Sie eine C-Funktion, die den Maximal- und Minimalwert (`short`) der unten dargestellten verketteten Liste definieren kann. Die Rückgabe soll durch Referenz (Zeiger) auf eine Struktur vom Typ "`result`" in den Feldern "`max`" und "`min`" erfolgen.

Bedingungen: Wenn die Kette leer ist, enthält der Zeiger "`first`" einen Wert `null` (0).

Prototyp der Funktion:

```c
void compute_min_max (const struct elem* first,
                      struct result* result);
```

Übung 7

Das ist die Deklaration einer `union`

```
struct str {
    uint16_t    val;
    struct str* next;
};

union example {
    uint32_t    word;
    uint16_t    hword;
    uint8_t     byte;
    struct str  elem;
} u;
```

Wenn sich "u" ab Adresse `0x2000` befindet, stellen Sie für jede Zuordnung die Bytes (in hexadezimaler Form) im Speicher dar

```
1) u.byte = 1536;
2) u.var = -3;
3) u.lg = 96;
4) u.elem.val = 127;
5) u.elem.pt = &u;
```

address	1)	2)	3)	4)	5)
0x2000					
0x2001					
0x2002					
0x2003					
0x2004					
0x2005					
0x2006					
0x2007					
0x2008					
0x2009					
0x200a					
0x200b					
0x200c					
0x200d					
0x200e					
0x200f					
0x2010					
0x2011					
0x2012					

Bemerkung: Die Speicherelemente, von denen der Inhalt nicht bekannt ist, müssen mit einem Kreuz markiert werden