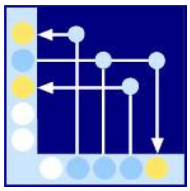
Hochschule Aalen
Fakultät Elektronik und Informatik
Studienbereich Informatik



Programmieren in C++

Vorlesung im Wintersemester 2024/2025
Prof. Dr. habil. Christian Heinlein

1. Übungsblatt (21. Oktober 2024)

Aufgabe 1: Referenzen

Lösen Sie die folgenden „Referenz-Puzzles“, d. h. sagen Sie die Ausgabe der Programme voraus, bevor Sie sie ausprobieren!

Teilaufgabe 1.a)

```
#include <iostream>
using namespace std;

int& f (int& x, int& y) {
    return x < y ? x : y;
}

int main () {
    int a = 1, b = 2;
    f(a, b) = 3;
    f(a, b) = 4;
    cout << a << " " << b << endl;
}
```

Teilaufgabe 1.b)

```
#include <iostream>
using namespace std;

int x = 1;

void f (int& y) {
    x += y;
    y += x;
}

int main () {
    f(x);
    cout << x << endl;
}
```

Teilaufgabe 1.c)

```
#include <iostream>
using namespace std;

int a [] = { 1, 2, 3 };
int* p = a + 3;

void f (int& x, int& y) {
    x = x + y;
    y = x - y;
    x = x - y;
}

int main () {
    f(*a, a[1]);
    f(a[2], p[-2]);
    cout << a[0] << " " << a[1] << " " << a[2] << endl;
}
```

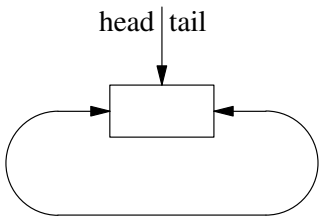
Teilaufgabe 1.d)

```
#include <iostream>
using namespace std;

int& swap (int& x, int& y) {
    int z = x; x = y; y = z;
    return x;
}

int main () {
    int a = 1, b = 2, c = 3, d = 4;
    swap(a, swap(b, swap(c, d)));
    cout << a << " " << b << " " << c << " " << d << endl;
}
```

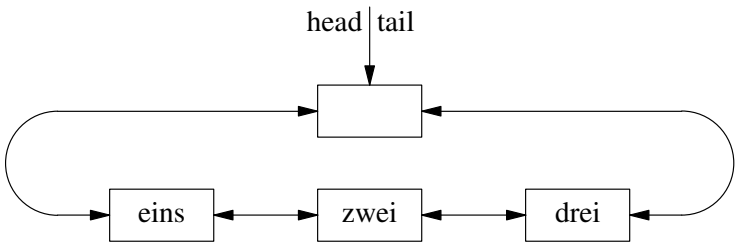
Aufgabe 2: Doppelt verkettete Listen

Gegeben sei die folgende unvollständige Definition einer Klasse/Struktur `List`:

```
#include <string>
#include <iostream>
using namespace std;

// Doppelt verkettete Ringliste mit Elementen des Typs string
// mit gemeinsamem Hilfsknoten am Anfang und am Ende der Liste.
struct List {
    // Knoten einer solchen Liste.
    struct Node {
        // Element.
        string elem;

        // Zeiger auf Vorgänger und Nachfolger.
        Node* prev;
        Node* next;

        // Initialisierung mit Element e
        // und möglicherweise undefinierten Verkettungszeigern.
        Node (const string& e) : elem{e} {}
    };

    // Zeiger auf den Hilfsknoten, der sich sowohl am Anfang als
    // auch am Ende der Liste befindet und immer zusätzlich zu den
    // "normalen" Knoten mit den Elementen der Liste vorhanden ist,
    // d. h. der Hilfsknoten ist einerseits Nachfolger des letzten
    // normalen Knotens und andererseits Vorgänger des ersten normalen
    // Knotens.
    // Eine leere Liste besteht nur aus dem Hilfsknoten, der dann
    // sein eigener Nachfolger und sein eigener Vorgänger ist.
    // (Da sich alle Daten eines union-Objekts an der selben Speicher-
    // adresse befinden, sind head und tail zwei verschiedene Namen
    // für denselben Zeiger.)
    union {
    Node* head;
    Node* tail;
    };
```



Leere Liste



Liste mit drei Elementen

```
    // Initialisierung als leere Liste.
    List ();

    // Element x am Anfang der Liste einfügen.
    void push_front (const string& x);

    // Element x am Ende der Liste einfügen.
    void push_back (const string& x);

    // Die Funktionen pop_front, pop_back, front und back
    // dürfen für eine leere Liste nicht aufgerufen werden.

    // Erstes Element der Liste entfernen
    // und den zugehörigen Knoten freigeben.
    void pop_front ();

    // Letztes Element der Liste entfernen
    // und den zugehörigen Knoten freigeben.
    void pop_back ();

    // Referenz auf das erste Element der Liste liefern.
    string& front ();

    // Referenz auf das letzte Element der Liste liefern.
    string& back ();

    // Alle Elemente der Liste in einer Zeile ausgeben.
    void print ();
};
```

Ergänzen Sie die Implementierungen des Konstruktors und der Elementfunktionen!

Alle diese Funktionen können entweder direkt in der Typdefinition implementiert werden, zum Beispiel:

```
struct List {
    ......
    string& front () {
        return head->next->elem;
    }
    ......
};
```

Oder sie können anschließend auf globaler Ebene implementiert werden, zum Beispiel:

```
string& List::front () {
    return head->next->elem;
}
```

Auf die Elementvariablen der Klasse kann in beiden Fällen gleichermaßen zugegriffen werden.