

Vorname

Nachname

Matrikel-Nr

**Aufgabe 1 (15 Punkte):** Schreiben Sie (in C++) ein Unterprogramm namens **drehRum** entsprechend der folgenden Deklaration:

```
1 unsigned drehRum(unsigned n);
2 // Liefert eine Zahl, deren dezimale Ziffernfolge die gleichen Ziffern
3 // enthaelt wie (die dezimale Ziffernfolge von) n, aber in umgekehrter
4 // Reihenfolge. Beispiele:
5 //
6 // drehRum(123)          ist gleich 321
7 // drehRum(987654321)   ist gleich 123456789
8 // drehRum(5)           ist gleich 5
```

**Aufgabe 2 (15 Punkte):** Schreiben Sie (in C++) ein Unterprogramm namens **anzahlAnfaenge** entsprechend der folgenden Deklaration:

```
1 unsigned anzahlAnfaenge(vector<string> const & vs, string const & s);
2 // Wieviele der Strings in vs beginnen mit dem String s? Diese Funktion
3 // liefert die Antwort. Beispiele:
4 //
5 // Sei v ein Vektor, der die folgenden Strings enthaelt:
6 // "abruf", "aufruf", "ablauf", "a", "aber". Dann gilt:
7 //
8 // anzahlAnfaenge(v, "ab") ist gleich 3
9 // anzahlAnfaenge(v, "auf") ist gleich 1
10 // anzahlAnfaenge(v, "abc") ist gleich 0
11 // anzahlAnfaenge(v, "") ist gleich 5
```

**Aufgabe 3 (20 Punkte):** Betrachten Sie das folgende C++-Programm:

```
1 // Datei Prog01.cpp
2
3 struct K01 {
4     unsigned const nr;
5     unsigned alter;
6     unsigned getNr() {return nr;}
7     unsigned getAlter() {return alter;}
8
9     K01(unsigned nr, unsigned alter) : nr(nr), alter(alter) {anzahl++;}
10    static unsigned anzahl; // Deklaration
11    static unsigned getAnzahl() {return anzahl;}
12 }; // struct K01
13
14 unsigned K01::anzahl = 0; // Definition
15
16 struct K02 {
17     K02 * next;
18     K01 * zk01;
19     K02 * getNext() {return next;}
20     K01 * getZk01() {return zk01;}
21     K02(K01 * zk01, K02 * next) : zk01(zk01), next(next) {}
22 }; // struct K02
23
24 void main() {
25     K01 otto(100, 23);
26     K02 anna(&otto, NULL);
27     K02 * bert = new K02(new K01(200, 25), &anna);
28     anna.next = bert->next;
29     ...
30 } // main
```

Stellen Sie sich vor, dass der Ausführer dieses Programm bis zur Zeile 29 ausgeführt hat. Stellen Sie alle bis dahin erzeugten Variablen als Bojen dar.

**Aufgabe 4** (20 Punkte): Betrachten Sie noch einmal das Programm der vorigen Aufgabe und stellen Sie sich noch einmal vor, dass der Ausführer das Programm bis zur Zeile 29 ausgeführt hat.

4.1. Wieviel **Module** existieren in diesem Moment?

4.2. Geben Sie die Namen dieser Module an. Mit den von Ihnen angegebenen Namen sollte man (z.B. in Zeile 29 des Programms) auf die Module zugreifen können.

4.3. Geben Sie für jeden Modul M an, welche **Elemente** sich in M befinden. Geben Sie für jedes Element einen **zusammengesetzten Namen** an (der aus einem Modulnamen und dem Namen des betreffenden Elements besteht).

**Aufgabe 5** (15 Punkte): Betrachten Sie das folgende C++-Programm:

```
1 // Datei prog02.cpp
2 #include <iostream>
3 #include <string>
4
5 void verziereUndGibAus(string & s) {
6     unsigned anzahl = 0;
7     for (unsigned i=0; i<s.size() && anzahl<3; i++) {
8         if (isdigit(s[i])) {
9             s.insert(i, "X");
10            anzahl++;
11        }
12    }
13    cout << s << endl;
14 } // verziereUndGibAus
15
16 void main() {
17     string s1("12AB");
18     string s2("AB12");
19     verziereUndGibAus(s1);
20     verziereUndGibAus(s2);
21 } // main
```

Was gibt dieses Programm (zur Standardausgabe, d.h. zum Bildschirm) aus? Die Funktion **isdigit** liefert **true**, wenn ihr Parameter eine Ziffer ist und sonst **false**.

**Aufgabe 6** (15 Punkte): Beantworten Sie die folgenden Fragen **kurz**, aber **genau** (möglichst so ähnlich, wie es in den Wiederholungen zu Beginn der Vorlesungen geschah):

6.1. Definition des Begriffs **binärer Baum**?

6.2. Wann ist ein binärer Baum **sortiert**?

6.3. Was ist ein **Modul**?

6.4. Was ist eine **Klasse**?

6.5. Was ist eine **Variable**?

6.6. Was ist ein **Typ**?

6.7. Eigentlich gibt es in allen Programmiersprachen nur 3 Arten von Befehlen. Wie heissen diese 3 Befehlsarten auf Deutsch und auf Englisch?

6.8. Woraus besteht ein C++-Programm (oder: Was sind die **größten sinnvollen Teile** eines C++-Programms?)

**Lösung 1:**

```

1 unsigned drehRum(unsigned n) {
2     // Liefert eine Zahl, deren dezimale Ziffernfolge die gleichen Ziffern
3     // enthaelt wie (die dezimale Ziffernfolge von) n, aber in umgekehrter
4     // Reihenfolge. Beispiele:
5     //
6     // drehRum(123)      ist gleich 321
7     // drehRum(987654321) ist gleich 123456789
8     // drehRum(5)       ist gleich 5
9
10    unsigned erg    = 0;
11    unsigned ziffer;
12
13    while(n != 0) {
14        ziffer = n % 10;
15        n     = n / 10;
16        erg   = erg * 10 + ziffer;
17    }
18
19    return erg;
20 } // drehRum

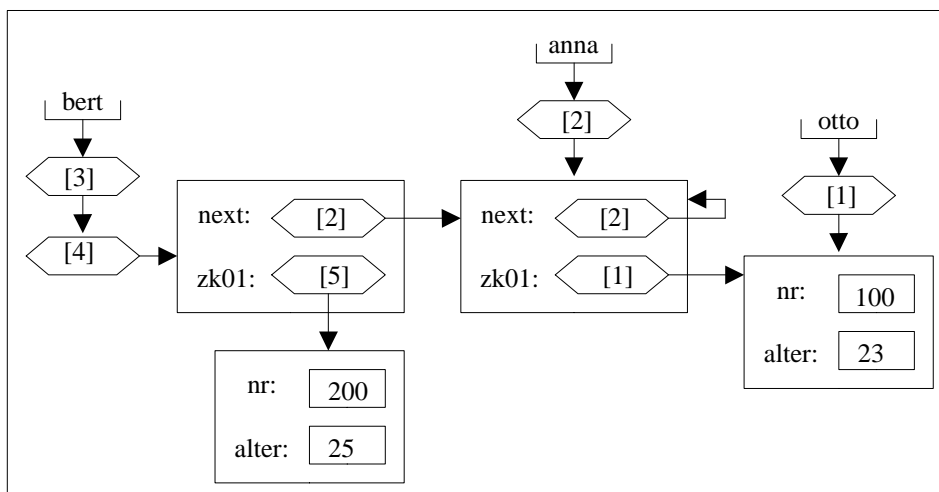
```

**Lösung 2:**

```

1 unsigned anzahlAnfaenge(vector<string> const & vs, string const & s) {
2     // Wieviele der Strings in vs beginnen mit dem String s? Diese Funktion
3     // liefert die Antwort. Beispiele:
4     //
5     // Sei v ein Vektor, der die folgenden Strings enthaelt:
6     // "abruf", "aufruf", "ablauf", "a", "aber". Dann gilt:
7     //
8     // anzahlAnfaenge(v, "ab") ist gleich 3
9     // anzahlAnfaenge(v, "auf") ist gleich 1
10    // anzahlAnfaenge(v, "abc") ist gleich 0
11    // anzahlAnfaenge(v, "") ist gleich 5
12
13    unsigned anzahl = 0;
14    for (unsigned i=0; i<vs.size(); i++) {
15        if (vs[i].substr(0, s.size()) == s) anzahl++;
16    }
17    return anzahl;
18 } // anzahlAnfaenge

```

**Lösung 3:**

**Lösung 4:**

4.1. Es existieren **6 Module**

4.2. Die Namen der Module: K01, K02, anna, otto, \*bert, \*(bert->zk01),

4.3. Die Namen der Elemente:

K01::K01 (Konstruktor), K01::anzahl (Attribut), K01::getAnzahl (Methode)

K02::K02 (Konstruktor)

anna.next, anna.zk01, anna.getNext, anna.getZk01

otto.nr, otto.alter, otto.getNr, otto.getAlter

bert->next, bert->zk01, bert->getNext, bert->getZk01

bert->zk01->nr, bert->zk01->alter, bert->zk01->getNr, bert->zk01->getAlter

**Lösung 5:** Ausgabe des Programms:

```
1 XXX12AB
2 ABXXX12
```

**Lösung 6** (15 Punkte):

6.1. Definition des Begriffs **binärer Baum**?

Ein **binärer Baum** ist entweder ein **leerer Baum** oder er besteht aus einem **Knoten K**, an dem zwei Bäume hängen. Diese Bäume bezeichnet man auch als den **linken** und **rechten Unterbaum** des Knotens K.

6.2. Wann ist ein **binärer Baum sortiert**?

Wenn für jeden Knoten **K** des Baumes gilt: Im **linken** Unterbaum von K sind alle Schlüssel **kleiner** als der Schlüssel des Knotens K und im **rechten** Unterbaum von K sind alle Schlüssel **größer** als der Schlüssel von K.

6.3. Was ist ein **Modul**?

Ein **Behälter** für Variablen, Konstanten, Unterprogramme, Typen etc., der aus mindestens 2 Teilen besteht, einem **sichtbaren** und einem **unsichtbaren** Teil.

6.4. Was ist eine **Klasse**?

Ein **Modul** und ein **Bauplan** für Module.

6.5. Was ist eine **Variable**?

Ein **Behälter** für Werte (dessen Inhalt man jederzeit verändern kann, z.B. mit Hilfe einer Zuweisung).

6.6. Was ist ein **Typ**?

**Antwort1:** Ein **Bauplan** für Variablen.

**Antwort2:** Eine Menge von **Werten** und eine Menge von **Operationen**, die man auf die Werte anwenden kann. (Das Wort **Operation** ist hier in seinem weiteren Sinne gemeint).

6.7. Eigentlich gibt es in allen Programmiersprachen nur 3 Arten von **Befehlen**. Wie heissen diese 3 Befehlsarten auf **Deutsch** und auf **Englisch**?

**Vereinbarungen** (declarations), **Ausdrücke** (expressions), **Anweisungen** (statements)

6.8. Woraus besteht ein **C++-Programm** (oder: Was sind die größten sinnvollen Teile eines C++-Programms?)

**Antwort1:** Ein C++-Programm besteht aus **Dateien** (besser: aus **Compilationen**).

**Antwort 2:** Ein C++-Programm besteht aus **Vereinbarungen**, die in **Dateien** stehen.