Fakultät IV
NI & CV

Probeklausur – C/C++

Einführung in die Informatik

Wintersemester 2017/2018

Hinweis: Diese Probeklausur ist eine kleine Aufgabensammlung, die etwa dem Schwierigkeitsgrad der schriftlichen Prüfung des Moduls Einführung in die Informatik entspricht. Die hier zur Verfügung gestellten Aufgaben decken jedoch nicht alle behandelten Themenbereiche ab. Darüber hinaus wird es in der Klausur Wissensaufgaben geben, welche aus allen Themen und sämtlichen Teilen der Veranstaltung (Vorlesung, Tutorium und Hausaufgaben) kommen können. In den Klausuren für dieses Semester wird die Gewichtung ca. 60 % Programmieren und 40 % Rechneraufbau sein.

Aufgabe 1 (Allgemeine Fragen (C)).

In dieser Aufgabe ist jeweils genau eine Antwort richtig, welche Sie ankreuzen sollen. Kreuzen Sie pro Teilaufgabe nur ein Kästchen an. Eine richtige Antwort ergibt einen Punkt, eine falsche 0 Punkte. Es gibt keine Minuspunkte. Um ein versehentlich gesetztes Kreuz wieder zu löschen, füllen Sie das jeweilige Kästchen aus und zeichnen ein leeres daneben. Ein Beispiel:

Fragetext		*Fragetext*
☒ Antwort 1	Korrektur →	☐ ■ Antwort 1
☐ Antwort 2		☐ Antwort 2
☐ Antwort 3		☒ Antwort 3
☐ Antwort 4		☐ Antwort 4

1. Welcher Datentyp steht im C-Standard **nicht** zur Verfügung?

   ☐ `float`
   ☐ `octa`
   ☐ `double`
   ☐ `int`

2. Welche der folgenden Anweisungen verursacht eine Warnung?

   ☐ `int d[] = {0};`
   ☐ `int c[10] = {1,2};`
   ☐ `int b[2] = {1,2,3};`
   ☐ `int a[0];`

3. Welcher Schleifentyp existiert in C++ **nicht**?

   ☐ while-Schleife
   ☐ for-Schleife
   ☐ repeat-until-Schleife
   ☐ do-while-Schleife

4. Welcher der folgenden Ausdrücke wird garantiert zu `true` ausgewertet?

   ☐ `(a < b) && (b < c)`
   ☐ `!(b || a)`
   ☐ `!!(a || !a)`
   ☐ `(a >= b && b >= a)`

5. Welche Aussage zu objektorientierter Programmierung in C++ ist **falsch**?

   ☐ C++ erlaubt das Erben von mehreren Superklassen.
   ☐ C++ erlaubt es, Konstruktoren mit dem Sichtbarkeitslevel `private` zu versehen.
   ☐ Von einer Klasse dürfen im allgemeinen beliebig viele andere Klassen erben.
   ☐ Von einer Superklasse geerbte Attribute sind immer vom Sichtbarkeitslevel `public`.

6. Welche C-Anweisung allokiert Speicher für 10 double-Variablen auf dem Stack?

   ☐ `double d = malloc(10);`
   ☐ `double d[] = {10};`
   ☐ `double d[] = malloc(10*sizeof(double));`
   ☐ `double d[10];`

7. Wofür werden **Exceptions** verwendet?

   ☐ Compileroptimierung
   ☐ Präprozessorbeschleunigung
   ☐ Netzwerkverbindungen
   ☐ Ausnahmebehandlung

8. Was bietet die Programmiersprache C **nicht**?

   ☐ `Primitive Datentypen`
   ☐ `Namespaces`
   ☐ `return-Anweisungen`
   ☐ `Referenzdatentypen`

9. Welche der folgenden Aussagen zu Rekursion ist **korrekt**?

   ☐ Endrekursionen haben oft eine kürzere Laufzeit als Lineare Rekursionen oder Kaskadenartige Rekursionen
   ☐ Rekursive Implementierungen sind häufig speicherlastiger als iterative Implementierungen
   ☐ Kaskadenartige Rekursionen sind immer Lineare Rekursionen
   ☐ Verschränkte Rekursionen bestehen aus genau 3 Funktionen, die sich im Kreis aufrufen

10. Wann wird eine Klasse in C++ als **abstrakt** bezeichnet?

    ☐ wenn der Konstruktor das Sichtbarkeitslevel `private` hat
    ☐ wenn alle Methoden mit dem Schlüsselwort `virtual` versehen sind
    ☐ wenn sie mindestens eine rein virtuelle Methode enthält
    ☐ wenn von der Klasse mindestens 2 Klassen erben

Aufgabe 2 (Bedingte Anweisungen).

1. Betrachten Sie die folgende Funktion:

```
int f(int x, int y) {
    if(x) {
        return y;
    } else if(y) {
        return x;
    } else {
        return x;
    }
}
```

(a) Vervollständigen Sie die folgende Tabelle, so dass sie für die angegebenen Wertepaare (x, y) der Parameter den zugehörigen Rückgabewert der Funktion `f` angibt.

x	y	f(x, y)
1	1	
1	0	
0	1	
0	0	

(b) Implementieren Sie eine möglichst einfache Funktion `int f2(int x, int y)`, die ohne `if`-Anweisung auskommt und für die obigen Wertekombinationen von `x` und `y` die gleichen Werte wie die Funktion `f` zurückliefert.

2. Betrachten Sie das folgende Punktesystem zur Notenvergabe:

Punkte	Note
36-40	1
31-35	2
26-30	3
21-25	4
0-20	5

Implementieren Sie eine Funktion `int getNote(int punkte)`, die als Wert die Note der übergebenen Punkte zurückliefert. Die Funktion soll den Wert `-1` zurückgeben, falls der Parameter `punkte` keine gültige Punktezahl darstellt.

Aufgabe 3 (Schleifen).

1. Wandeln Sie die folgende `for`-Anweisung in eine `while`-Anweisung um.

```
for(int i = 533; i > -12; i -= 7) {
    printf("%d \n", i);
}
```

2. Implementieren Sie eine Funktion `int zaehleNullen(int array[], int laenge)`, die die Anzahl der Nullen in dem übergebenen Array zurückgibt. Gehen Sie davon aus, dass `laenge` die Anzahl der Array-Elemente enthält.

Aufgabe 4 (Schleifen (MC)).

Es soll die C++-Funktion `sumArray` vervollständigt werden, welche die Summe der Zahlen in dem übergebenen Array `zahlen` zurückgibt. Die Länge des Arrays ist in dem Parameter `length` gespeichert. Betrachten Sie dazu den folgenden Quellcode:

```
0   int sumArray(int zahlen[], int length){
1           int sum = 0;
2           for(___(1)___; ___(2)___; ___(3)___){
3                   ___(4)___
4           }
5           return sum;
6   }
```

Geben Sie an, was bei den Stellen `___(1)___` bis `___(4)___` eingetragen werden soll.

1. Stelle `___(1)___`:

   ☐ `int i=0`

   ☐ `int i=1`

   ☐ `i<length`

   ☐ `i++`

2. Stelle `___(2)___`:

   ☐ `i<length`

   ☐ `i<<length`

   ☐ `i>length`

   ☐ `i>=length`

3. Stelle `___(3)___`:

   ☐ `i++`

   ☐ `i+=length`

   ☐ `i<length`

   ☐ `i=1`

4. Stelle `___(4)___`:

   ☐ `sum += zahlen[i];`

   ☐ `sum++;`

   ☐ `sum = zahlen[i];`

   ☐ `sum = zahlen[i]++;`

Aufgabe 5 (Heap-Arrays).

1. Vervollständigen Sie die folgende Funktion, in welcher die Anzahl der Zeichen in einem übergebenen C-String bestimmt und diese Zahl anschließend zurückgegeben werden sollen.

```
int zaehle(char* text)
{









}
```

2. Schreiben Sie eine C-Funktion `void harmonisch(int anzahlElemente)`, in welcher Folgendes implementiert werden soll:

   (a) Allokation von Heap-Speicher für einen `double`-Array mit `anzahlElemente` Elementen.

   (b) Befüllung des Arrays nach dem Muster: 0. Element = 1.0/1.0, 1. Element = 1.0/2.0, 2. Element = 1.0/3.0, 3. Element = 1.0/4.0 ...

   (c) Ausgabe der Werte aller Array-Elemente auf die Konsole.

   (d) Freigabe des belegten Speichers.

```
#include <stdio.h>
#include <stdlib.h>

void harmonisch(int anzahlElemente)
{









}
```

Aufgabe 6 (Pointer und Strukturen).

1. Schreiben Sie eine Funktion `void halbiere(int* zahl)`, die eine `int`-Zahl halbiert (Ganzzahldivision), auf welche der übergebene Pointer `zahl` zeigt.

```
void halbiere(int* zahl){



}
```

2. Wie sieht der Array `arr` nach Ausführung der letzten Zeile aus?

```
int arr[5] = {1, 3, 5, 7, 9};
int* ptr = arr + 2;
ptr[1] = 0;
```

Index	0	1	2	3	4
Wert					

3. Legen Sie eine Struktur `complex` an, um komplexe Zahlen bestehend aus einem Real- und einem Imaginärteil, jeweils vom Typ double, speichern zu können.
Die Struktur soll mit dem folgenden Code kompatibel sein.
Implementieren Sie weiterhin die Funktion `print_real_numbers`, die ein `complex`-Array sowie seine Länge übergeben bekommt, und die Realteile aller Elemente, deren Imaginärteil 0 ist, kommasepariert auf die Konsole schreibt.

```
// Struktur complex anlegen




// Funktion print\_real\_numbers implementieren









int main() {
    struct complex c[] = {{0.0,0.0},{42.0,0.0},{19.99,13.0}};
    complex[0].r = 1.0;
    complex[0].i = 1.0;
    print_real_numbers(c, sizeof(c)/sizeof(c[0]));
}
```

Aufgabe 7 (Klassen und Objekte).

1. Ergänzen Sie das folgende C++-Programm wie in den Kommentaren im Quellcode beschrieben.

```
#include <iostream>

class C {
    public:
        int a;
        C(int a);
        void f();
};

C::C(int a) {
    // Initialisiere Attribut mit uebergebenen Parameter
    ...
}

void C::f() {
    a *= 2;
    std::cout << "a = " << a << std::endl;
}

int main() {
    // Erzeuge ein Objekt der Klasse C. Dabei soll der Wert des
    // Attributs a mit 5 initialisiert werden.
    ...

    // Rufe die Methode f() auf.
    ...
}
```

2. Implementieren Sie eine Klasse `Fahrrad`. Die Klasse soll folgenden Anforderungen genügen:

   (a) Der Zustand eines Objekts der Klasse `Fahrrad` wird durch seine Geschwindigkeit `speed` und den eingestellten Gang `gears` beschrieben. Beide Attribute sind vom Typ `int`. Sowohl die Geschwindigkeit als auch der Gang dürfen nicht negativ werden. Ein Zugriff auf die Attribute außerhalb der Klasse ist nicht möglich.

   (b) Die Klasse besitzt einen parameterlosen Konstruktor, der die Geschwindigkeit mit 0 und den Gang mit 1 initialisiert.

   (c) Die Klasse besitzt einen erweiterten Konstruktor mit zwei Parametern zur Initialisierung der Attribute.

   (d) Die Klasse besitzt eine Methode zum Ändern der Geschwindigkeit. Die Differenz zur alten Geschwindigkeit wird als Parameter der Methode übergeben.

   (e) Die Klasse besitzt eine Methode zum Setzen des Gangs. Die Anzahl wird dabei als Parameter der Methode übergeben.

   (f) Auf die Konstruktoren und auf die Methoden soll von überall zugegriffen werden können.

Aufgabe 8 (Vererbung).

Betrachten Sie die Klassenhierarchie in der folgenden Abbildung.

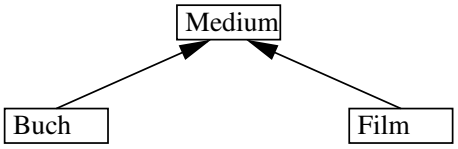


Abbildung 1: Klassenhierarchie

Beachten Sie außerdem: Bücher besitzen einen Titel und einen Autor. Filme werden durch ihren Titel und ihre Spieldauer beschrieben. Implementieren Sie die Klassenhierarchie der obigen Abbildung in C++. Sämtliche Attribute der zu implementierenden Klassen sollen `private` sein, während Konstruktoren und Methoden `public` sein sollen. Der Einfachheit halber soll sämtlicher Code in einer einzelnen Datei sein.

Gehen Sie beim Implementieren folgendermaßen vor:

1. Implementieren Sie eine Klasse `Medium`. Die Klasse `Medium` besitzt als Attribut einen Titel vom Typ `std::string`. Implementieren Sie einen Konstruktor zur Initialisierung des Titels mit einem Parameter. Definieren Sie eine Methode `void ausgeben()`, welche den Titel auf der Konsole ausgibt.
2. Implementieren Sie eine Unterklasse `Buch` der Klasse `Medium`. Die Klasse `Buch` besitzt als zusätzliches Attribut einen Autor vom Typ `std::string`. Implementieren Sie einen Konstruktor zur Initialisierung von Titel und Autor mit Parametern. Definieren Sie eine Methode `ausgeben()`, welche den Titel sowie den Autor auf der Konsole ausgibt.
3. Implementieren Sie eine Unterklasse `Film` der Klasse `Medium`. Die Klasse `Film` besitzt als zusätzliches Attribut die Spieldauer vom Typ `int`. Implementieren Sie einen Konstruktor mit Parametern zur Initialisierung von Titel und Spieldauer. Definieren Sie eine Methode `ausgeben()`, welche den Titel sowie die Spieldauer auf der Konsole ausgibt.

Aufgabe 9 (Zahlensysteme).

1. Rechnen Sie die Zahl $423_{(6)}$ in eine Zahl zur Basis 16 um. Der Lösungsweg muss erkennbar sein.

2. Berechnen Sie die folgenden Aufgaben, dargestellt in Basis 10, unter Verwendung der Zweierkomplementdarstellung mit 4 Bit. Der Lösungsweg muss erkennbar sein. Bei welchen Aufgaben findet ein Über- bzw. Unterlauf statt und woran erkennen Sie diesen jeweils?

   (a) $6 + (-2)$

   (b) $-7 + (-2)$

   (c) $4 + 7$

Aufgabe 10 (Fließkommazahlen).

1. Addieren Sie die Zahlen $45_{(10)}$ und $0.0625_{(10)}$ im angegebenen Gleitkommaformat und konvertieren Sie das Ergebnis zurück in eine Dezimalzahl.

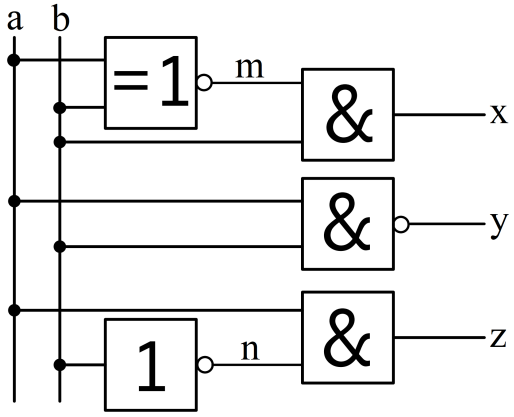
Vorzeichen	Exponent	Mantisse
1 Bit	4 Bits	5 Bits

Der Exzess beträgt: $7_{(10)} = 0111_{(2)}$

2. Wie groß ist der Rundungsfehler, der bei der Addition in der ersten Teilaufgabe entsteht. Geben Sie den Rundungsfehler als Dezimalzahl an.

Aufgabe 11 (Schaltungen).

1. Stellen Sie die Wertetabelle für folgende Schaltung auf:



(a und b sind die Eingänge, x, y und z sind die Ausgänge der Schaltung.)

2. Zeichnen Sie eine Schaltung, die der folgenden Wertetabelle entspricht.

a	b	c	y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

(a, b und c sind die Eingänge, y ist der Ausgang der Schaltung.)