Fakultät IV
NI & CV

Probeklausur – C/C++

Einführung in die Informatik

Wintersemester 2014/2015

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 sein: 40 % Rechneraufbau und 60 % Programmieren

Aufgabe 1 (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 unter Verwendung der Zweierkomplementdarstellung mit 4 Bit. Der Lösungsweg muss erkennbar sein. Bei welchen Aufgaben findet ein Über- bzw. Unterlauf statt? Woran erkennen Sie jeweils den Über- bzw. Unterlauf?

   (a) $6 + (-2)$

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

   (c) $4 + 7$

Aufgabe 2 (Fließkommazahlen).

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

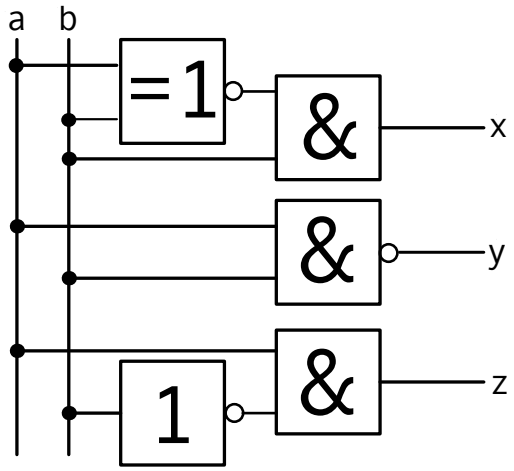
Vorzeichen	Exponent	Mantisse
1 Bit	4 Bits	5 Bits

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 3 (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.)

Aufgabe 4 (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 5 (Schleifen).

1. Was ist der Wert von `k` nach Ausführung der folgenden Anweisungen?

```
int k = 2;
for(int i = 2; i < 7; i += 2) {
    k *= 2;
}
```

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

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

3. 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 6 (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 7 (Pointer).

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					

Aufgabe 8 (Klassen und Objekte).

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

```
    #include <iostream>

class C {

    public:
        int a;

        C(int a) {
            // initialisiere Attribut mit uebergebenen Parameter

        }

        void 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 und die Anzahl der Gänge beschrieben. Beide Attribute sind vom Typ `int`. Sowohl die Geschwindigkeit als auch die Anzahl der Gänge 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 die Anzahl der Gänge 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 der Anzahl der Gänge. Die Anzahl wird dabei als Parameter der Methode übergeben.

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

Aufgabe 9 (Vererbung).

Betrachten Sie die Klassenhierarchie in der folgenden Abbildung.

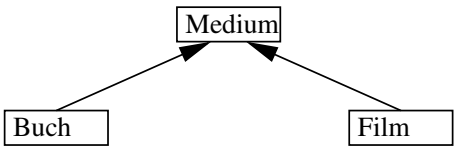


Abbildung 1: Klassenhierarchie

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. Gehen Sie 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.