Klausur Programmieren 1

SS 17

Prof. Dr.-Ing. Jürgen Brauer

Prüfungstag	Dienstag, 18.07.2017
Prüfungszeitraum	10:00 - 12:00
Prüfungsdauer	120min
Prüfungsraum	A002/003

Hochschule Kempten
University of Applied Sciences

Bitte ausfüllen!	
Name (Blockschrift):	
Vorname (Blockschrift):	
Matrikelnummer	
Unterschrift	

Hinweise zur Klausur

- Bitte tragen Sie als erstes Ihren Namen, Vornamen, Matrikel-Nr und Ihre Unterschrift ein
- Legen Sie bitte als Identitätsnachweis Ihren Personal- oder Studienausweis neben sich
- Schreiben Sie nur mit dokumentenechten Stiften (kein Bleistift). Verwenden Sie nicht die Farbe rot oder grün.
- Schreiben Sie deutlich! Unleserliche Passagen können nicht korrigiert werden.
- Es sind keine Hilfsmittel erlaubt!
- Täuschungsversuche führen zum Nichtbestehen der Klausur.
- Tragen Sie Ihre Lösung zu jeder Aufgabe in das vorgesehene Feld nach der Aufgabenbeschreibung ein. Eigenes Papier darf nicht verwendet werden!
- Die nicht bedruckten Rückseiten dürfen Sie bei Bedarf als zusätzlichen Platz für Ihre Antworten verwenden. Kennzeichen Sie in diesem Falle dann deutlich bei einer Aufgabe, dass Sie von diesem zusätzlichen Platz Gebrauch machen und kennzeichnen Sie Ihre Antwort auf der Rückseite eindeutig mit der Aufgaben-Nr.
- Diese Klausur enthält 10 Aufgaben und besteht inklusive diesem Deckblatt aus insgesamt 12 Seiten.
- Insgesamt können Sie maximal 86 Punkte erreichen:

Aufgabe	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
Punkte	10	6	10	10	10	10	8	6	6	10

Gesamtpunkte	
Note	

Aufgabe 1	Source-Code Formatierung

Betrachten Sie folgenden Source-Code:

```
#include <stdio.h>
 float f543(float* a879, int b03) {
   float opmv980 = 0.0f; for(int op230=0; op230<b03; op230=op230-5+10-5+1)
      opmv980 += a879[op230];  return opmv980; }
int main() {   float mm0289[]={1.0f,2.0f,3.0f,4.0f,5.0f};
   printf("mm0289=%.2f", f543(mm0289, 5));   _getch();
}
```

---

i) Welche Maßnahmen könnte man treffen, damit der Source-Code besser lesbar ist?
(bis zu 5 Pkte, pro sinnvoller Maßnahme 1 Pkt)

- Sprechende Bezeichner für Funktionsnamen
- Sprechende Bezeichner für Variablennamen
- Nur ein Befehl pro Zeile
- Einrückung des Schleifenkörpers
- Kommentare
- Zählerinkrement klar um eins erhöhen
- Besser verständlichere Ausgaben auf Konsole für Benutzer
- Leerzeilen zwischen Sinnabschnitten

ii) Schreiben Sie den Source-Code nochmal neu unter Berücksichtigung der von Ihnen vorgeschlagenen Maßnahmen: (3 Pkt)

```
#include <stdio.h>

float compute_sum(float* ptrArray, int N) {
   float sum = 0.0f;
   for(int i=0; i<N; i++)
      sum+=ptrArray[i];
   return sum;
}

int main() {
   float testArray[]={1.0f,2.0f,3.0f,4.0f,5.0f};
   printf("sum of all elements in testArray is %.2f", compute_sum(testArray, 5));
   _getch();
}
```

iii) Was macht die Funktion f543? (2 Pkt)

Sie berechnet die Summe der ersten N Arrayelemente des angegebenen Arrays.

Erreichte Punktzahl für diese Aufgabe	von 10 Punkten

Aufgabe 2	**for**-Schleife & **do-while** Schleife

Schreiben Sie jeweils eine Schleife, die die angegebenen Zahlen in der angegebenen Reihenfolge ausgibt!

Bei jedem Schleifendurchlauf soll nur eine weitere Zahl ausgegeben werden.

---

i) Schreiben Sie eine **for**-Schleife, die folgende Zahlen ausgibt: (2 Pkt)

```
   2  4  6  8  10  12  14  16  18
```

```
for (int i = 2; i <= 18; i+=2) printf("%d  ", i);
```

ii) Schreiben Sie eine **do-while**-Schleife, die alle Zahlen zwischen 1 und 100 ausgibt, die durch 7 oder 17 teilbar sind: (4 Pkt)

```
    7  14  17  21  28  34  35  42  49  51  56  63  68  70  77  84  85  91  98
```

```
int i=0;
do
{
   i++;
   if ((i % 7==0) || (i%17)==0)
      printf("%d  ", i);
} while (i<=100);
```

Erreichte Punktzahl für diese Aufgabe	von 6 Punkten

Aufgabe 3	**while**-Schleife & Fibonacci-Zahlen ohne rekursive Funktion berechnen

Schreiben Sie eine while-Schleife, die die Fibonacci-Zahlen f(2) bis f(15) ausgibt, wobei die Fibonacci Zahlen folgendermaßen definiert sind:

`f(0):=0, f(1):=1 und f(n):=f(n-1)+f(n-2)`

Bei dem i-ten Schleifendurchlauf soll die nächste Fibonacci-Zahl f(i) auf Basis vorher berechneter Fibonacci-Zahlen berechnet werden und der Wert von f(i) ausgegeben werden.

Gewünschte Ausgabe:

```
f(2)=1  f(3)=2  f(4)=3  f(5)=5  f(6)=8  f(7)=13  f(8)=21  f(9)=34  f(10)=55
f(11)=89  f(12)=144  f(13)=233  f(14)=377  f(15)=610
```

Hinweis: Verwenden Sie bei Ihrer Lösung **<u>keine</u> rekursive Funktion**!

```c
int pp = 0;
int p = 1;
int n = 2;
while (n <= 15)
{
   int result = pp + p;
   printf("f(%d)=%d  ", n, result);
   pp = p;
   p = result;
   n++;
}
```

Erreichte Punktzahl für diese Aufgabe	von 10 Punkten

Aufgabe 4	Verschachtelte Schleifen & Programmausführungspfad verfolgen

Was gibt folgendes Programm aus?

```
void foo2() {
   printf("=345");
}

void foo(int i, int j) {
   int sum=i+j;
   printf("%d", sum);

   if (sum%2==0)
      printf("+");

   (i==3) && (j==2) ? foo2() : printf("") ;
}


int main() {

   for (int i=1; i<=3; i++)
      for (int j=1; j<=2; j++)
         foo(i,j);

   _getch();
}
```

Lösung:

2+334+4+5=345

10 Punkte bei komplett richtiger Ausgabe
6 Punkte bei 1 falscher Zahl / Zeichen
2 Punkte bei 2 falschen Zahlen / Zeichen
0 Punkte sonst

Erreichte Punktzahl für diese Aufgabe	von 10 Punkten

Aufgabe 5	Pointer & Referenzen, Parameterübergabe an Funktionen

Gegeben sei folgendes C++ Programm. Geben Sie an, welche Ausgabe das Programm erzeugt!

```
void foo(int* A, int* c) {
   *c=*A + *(A+2);
}

void foo2(int c) {
   c += 100;
}

void foo3(int& c) {
   c += 50;
}

int main() {

   int A[]={10,20,30,40,50};
   int c=0;

   foo(A, &c);
   printf("c1=%d\n", c);

   foo2(c);
   printf("c2=%d\n", c);

   foo3(c);
   printf("c3=%d\n", c);
}
```

Lösung:
c1=40
c2=40
c3=90

Erreichte Punktzahl für diese Aufgabe	von 10 Punkten

Aufgabe 6	Statische & Dynamische Speicherallozііerung

In der Computertomographie arbeitet man mit sog. Voxeldatensätzen. Dabei ist ein Voxeldatensatz ein 3D Array, bei dem für jeden Eintrag des 3D Arrays ein Wert gespeichert werden kann, z.B. ein `int` Wert.

(i) 3 Punkte: Wie deklariert man in C zur Compilezeit (d.h. vor dem Start eines Programmes = statische Speicherallozііerung) ein 3D Array, bei dem die 3 Dimensionen die Größen dimx, dimy und dimz haben?

(ii) 2 Punkte: Geben Sie ein Codestück in C an, um alle Einträge des 3D Arrays auf den Wert 0 zu setzen!

(iii) 5 Punkte: Geben Sie ein Codestück in C an, bei dem zur Laufzeit (d.h. nach dem Start eines Programmes = dynamische Speicherallozііerung) die Größen dimx, dimy und dimz vom Benutzer eingegeben werden und ein entsprechend großes 3D Array erzeugt wird!

```c
#include <stdio.h>
#include <conio.h>
#include <malloc.h>
#define dimstatz 10
#define dimstaty 20
#define dimstatx 30

int main() {

   // (i)
   int A_static[dimstatz][dimstaty][dimstatx];

   // (ii)
   for (int z = 0; z < dimstatz; z++)
     for (int y = 0; y < dimstaty; y++)
       for (int x = 0; x < dimstatx; x++)
         A_static[z][y][x] = 0;

   // (iii)
   int dimx,dimy,dimz;
   printf("\nLaenge x-Dimension: "); scanf_s("%d", &dimx);
   printf("\nLaenge y-Dimension: "); scanf_s("%d", &dimy);
   printf("\nLaenge z-Dimension: "); scanf_s("%d", &dimz);
   char*** A;
   A = malloc(dimz * sizeof(char**));
   for (int z = 0; z < dimz; z++) {
     A[z] = malloc(dimy * sizeof(char*));
     for (int y = 0; y < dimy; y++)
       A[z][y] = malloc(dimx * sizeof(char));
   }
   _getch();
}
```

Erreichte Punktzahl für diese Aufgabe	von 10 Punkten

Aufgabe 7	Klassen

Gegeben sei folgender Code:

```
class A {
   public:
      virtual void f1() = 0;
      virtual void f2() = 0;
      int var1;
   protected:
      int var2;
   private:
      int var3;
};

class B : public A {
   void f1() { printf("We are in f1()\n"); }
};

class C : public B {
   void f2() { printf("We are in f2()\n"); }
};

int main() {
   A* myA = new A(); // (Z1)
   B* myB = new B(); // (Z2)
   C* myC = new C(); // (Z3)
   myC->var1 = 1;    // (Z4)
   myC->var2 = 2;    // (Z5)
   myC->var3 = 1;    // (Z6)
}
```

Identifizieren Sie von den Zeilen Z1-Z6 alle Zeilen die nicht kompiliert werden können und begründen Sie jeweils wieso (jeweils 2 Pkt bei richtiger Zeile und Begründung)!

Z1: kompiliert nicht, da Klasse A abstrakt
Z2: kompiliert nicht, da Klasse B abstrakt
Z5: kompiliert nicht, da auf var2 nicht zugegriffen werden darf aus main() Fkt
Z6: kompiliert nicht, da auf var3 nicht zugegriffen werden darf aus main() Fkt

Erreichte Punktzahl für diese Aufgabe	von 8 Punkten

Aufgabe 8	Übersetzungsprozess

Beschreiben Sie mit eigenen Worten, was beim Übersetzen eines C bzw. C++ Programmes in Maschinencode passiert!

Gehen Sie hierbei v.a. auf die Aufgaben des
- Präprozessors
- Compilers
- Linkers

ein!

Für Präprozessor, Compiler, Linker jeweils 2 Pkt

Erreichte Punktzahl für diese Aufgabe	von 6 Punkten

Aufgabe 9	Regeln in boolesche Ausdrücke fassen

Es soll eine C Funktion `teste_ob_schaltjahr` angegeben werden, die in der Lage ist, zu überprüfen, ob ein angegebenes Jahr ein Schaltjahr ist oder nicht!

Hierzu die Regeln von https://de.wikipedia.org/wiki/Schaltjahr:

- Die durch 4 ganzzahlig teilbaren Jahre sind Schaltjahre.
- Säkularjahre, also die Jahre, die ein Jahrhundert abschließen (z.B. 1800, 1900, 2100 und 2200) sind doch keine Schaltjahre.
- Schließlich sind die durch 400 ganzzahlig teilbaren Säkularjahre doch wiederum Schaltjahre. Damit sind z.B. 1600, 2000 und 2400 jeweils wieder Schaltjahre.

Die Codezeilen

```
teste_ob_schaltjahr(1900);
teste_ob_schaltjahr(1976);
teste_ob_schaltjahr(2000);
teste_ob_schaltjahr(2001);
teste_ob_schaltjahr(2100);
teste_ob_schaltjahr(2400);
```

sollen dann zum Beispiel zu folgender Ausgabe führen:

```
Das Jahr 1900 ist kein Schaltjahr.
Das Jahr 1976 ist ein Schaltjahr.
Das Jahr 2000 ist ein Schaltjahr.
Das Jahr 2001 ist kein Schaltjahr.
Das Jahr 2100 ist kein Schaltjahr.
Das Jahr 2400 ist ein Schaltjahr.
```

```c
#include <stdio.h>
#include <conio.h>

void teste_ob_schaltjahr(int jahr)
{
  printf("Das Jahr %d ist ", jahr);
  if ( (jahr%4==0) && ((jahr%100!=0) || (jahr%400==0)) )
    printf("ein Schaltjahr.\n");
  else
    printf("kein Schaltjahr.\n");
}

int main() {
   teste_ob_schaltjahr(1900);
   teste_ob_schaltjahr(1976);
   teste_ob_schaltjahr(2000);
   teste_ob_schaltjahr(2001);
   teste_ob_schaltjahr(2100);
   teste_ob_schaltjahr(2400);
   _getch();
}
```

Erreichte Punktzahl für diese Aufgabe	von 6 Punkten

Aufgabe 10	Klassen und Ableitung

Betrachten Sie den Code auf nachfolgender Seite und ergänzen Sie alle 10 Codestellen die mit

_____

gekennzeichnet sind (d.h. Schreiben Sie dort hin, was dort fehlt), so dass das Programm compiliert und folgende Ausgabe erzeugt:

```
Spritztour mit Rennwagen:
Das Auto ist 2.0 Stunde(n) weiter gefahren. Neuer km-Stand: 500.00
Das Auto ist 1.0 Stunde(n) weiter gefahren. Neuer km-Stand: 750.00

Spritztour mit LahmeEnte:
Das Auto ist 2.0 Stunde(n) weiter gefahren. Neuer km-Stand: 180.00
Das Auto ist 1.0 Stunde(n) weiter gefahren. Neuer km-Stand: 270.00
```

(Pro richtigem Eintrag 1 Punkt)

```
_____________ <stdio.h>


____________ PKW {


  double km_stand;
  double max_geschwindigkeit;


  _____________:


    PKW(double ___________________________) {
      this->km_stand = 0.0;
      this->max_geschwindigkeit = max_geschwindigkeit;
    }


    void fahre_mit_max_geschwindigkeit(double ___________________________) {


      ______________________________ += max_geschwindigkeit * fahrzeit;
      printf("Das Auto ist %.1f Stunde(n) weiter gefahren. Neuer km-Stand:
              %.2f\n", fahrzeit, km_stand);
    }
};


class Rennwagen : public _____________ {
public: Rennwagen() : PKW(250)  {}
};



class LahmeEnte : public _____________ {
public: LahmeEnte() : PKW(90) {}
};



int _____________() {

   Rennwagen wagen1;
   printf("Spritztour mit Rennwagen:\n");
   wagen1.fahre_mit_max_geschwindigkeit( 2.0 );
   wagen1.fahre_mit_max_geschwindigkeit( 1.0 );


   LahmeEnte ____________________;
   printf("\nSpritztour mit LahmeEnte:\n");
   wagen2.fahre_mit_max_geschwindigkeit( 2.0 );
   wagen2.fahre_mit_max_geschwindigkeit( 1.0 );

}
```

Lösung:

```
#include <stdio.h>

class PKW {
  double km_stand;
  double max_geschwindigkeit;

  public:
    PKW(double max_geschwindigkeit) {
      this->km_stand = 0.0;
      this->max_geschwindigkeit = max_geschwindigkeit;
    }

    void fahre_mit_max_geschwindigkeit(double fahrzeit) {
      km_stand += max_geschwindigkeit * fahrzeit;
      printf("Das Auto ist %.1f Stunde(n) weiter gefahren. Neuer km-Stand: %.2f\n",
fahrzeit, km_stand);
    }
};

class Rennwagen : public PKW {
public: Rennwagen() : PKW(250)  {}
};

class LahmeEnte : public PKW {
public: LahmeEnte() : PKW(90) {}
};

int main() {

   Rennwagen wagen1;
   printf("Spritztour mit Rennwagen:\n");
   wagen1.fahre_mit_max_geschwindigkeit( 2.0 );
   wagen1.fahre_mit_max_geschwindigkeit( 1.0 );

   LahmeEnte wagen2;
   printf("\nSpritztour mit LahmeEnte:\n");
   wagen2.fahre_mit_max_geschwindigkeit( 2.0 );
   wagen2.fahre_mit_max_geschwindigkeit( 1.0 );

}
```

Erreichte Punktzahl für diese Aufgabe	von 10 Punkten