

Klausur Programmieren 1

SS 17

Prof. Dr.-Ing. Jürgen Brauer



Hochschule Kempten
University of Applied Sciences

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

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 Studenausweis 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. Kennzeichnen 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
<p>Betrachten Sie folgenden Source-Code:</p> <pre data-bbox="165 309 1203 517">#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(); }</pre>	
<p>i) Welche Maßnahmen könnte man treffen, damit der Source-Code besser lesbar ist? (bis zu 5 Pkte, pro sinnvoller Maßnahme 1 Pkt)</p> <p>ii) Schreiben Sie den Source-Code nochmal neu unter Berücksichtigung der von Ihnen vorgeschlagenen Maßnahmen: (3 Pkt)</p> <p>iii) Was macht die Funktion f543? (2 Pkt)</p>	
Erreichte Punktzahl für diese Aufgabe	von 10 Punkten

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

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

Aufgabe 3	while-Schleife & Fibonacci-Zahlen ohne rekursive Funktion berechnen
<p>Schreiben Sie eine while-Schleife, die die Fibonacci-Zahlen $f(2)$ bis $f(15)$ ausgibt, wobei die Fibonacci Zahlen folgendermaßen definiert sind:</p> $f(0) := 0, \quad f(1) := 1 \quad \text{und} \quad f(n) := f(n-1) + f(n-2)$ <p>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.</p> <p>Gewünschte Ausgabe:</p> $f(2)=1 \quad f(3)=2 \quad f(4)=3 \quad f(5)=5 \quad f(6)=8 \quad f(7)=13 \quad f(8)=21 \quad f(9)=34 \quad f(10)=55$ $f(11)=89 \quad f(12)=144 \quad f(13)=233 \quad f(14)=377 \quad f(15)=610$ <p>Hinweis: Verwenden Sie bei Ihrer Lösung keine rekursive Funktion!</p> <hr/>	
Erreichte Punktzahl für diese Aufgabe	von 10 Punkten

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();
}
```

Erreichte Punktzahl für diese Aufgabe

von 10 Punkten

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);
}
```

Erreichte Punktzahl für diese Aufgabe

von 10 Punkten

Aufgabe 6	Statische & Dynamische Speicherallozierung
<p>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 <code>int</code> Wert.</p> <p>(i) 3 Punkte: Wie deklariert man in C zur Compilezeit (d.h. vor dem Start eines Programmes = statische Speicherallozierung) ein 3D Array, bei dem die 3 Dimensionen die Größen <code>dimx</code>, <code>dimy</code> und <code>dimz</code> haben?</p> <p>(ii) 2 Punkte: Geben Sie ein Codestück in C an, um alle Einträge des 3D Arrays auf den Wert 0 zu setzen!</p> <p>(iii) 5 Punkte: Geben Sie ein Codestück in C an, bei dem zur Laufzeit (d.h. nach dem Start eines Programmes = dynamische Speicherallozierung) die Größen <code>dimx</code>, <code>dimy</code> und <code>dimz</code> vom Benutzer eingegeben werden und ein entsprechend großes 3D Array erzeugt wird!</p>	
Erreichte Punktzahl für diese Aufgabe	von 10 Punkten

Aufgabe 7	Klassen
<p>Gegeben sei folgender Code:</p> <pre data-bbox="165 282 783 1077">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) }</pre> <p>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)!</p> <hr data-bbox="165 1249 1359 1252"/>	
Erreichte Punktzahl für diese Aufgabe	von 8 Punkten

Aufgabe 8	Übersetzungsprozess
<p>Beschreiben Sie mit eigenen Worten, was beim Übersetzen eines C bzw. C++ Programmes in Maschinencode passiert!</p> <p>Gehen Sie hierbei v.a. auf die Aufgaben des</p> <ul style="list-style-type: none">- Präprozessors- Compilers- Linkers <p>ein!</p> <hr/>	
Erreichte Punktzahl für diese Aufgabe	von 6 Punkten

Aufgabe 9	Regeln in boolesche Ausdrücke fassen
<p>Es soll eine C Funktion <code>teste_ob_schaltjahr</code> angegeben werden, die in der Lage ist, zu überprüfen, ob ein angegebenes Jahr ein Schaltjahr ist oder nicht!</p> <p>Hierzu die Regeln von https://de.wikipedia.org/wiki/Schaltjahr:</p> <ul style="list-style-type: none">➤ 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 <u>doch</u> wiederum Schaltjahre. Damit sind z.B. 1600, 2000 und 2400 jeweils wieder Schaltjahre. <p>Die Codezeilen</p> <pre>teste_ob_schaltjahr(1900); teste_ob_schaltjahr(1976); teste_ob_schaltjahr(2000); teste_ob_schaltjahr(2001); teste_ob_schaltjahr(2100); teste_ob_schaltjahr(2400);</pre> <p>sollen dann zum Beispiel zu folgender Ausgabe führen:</p> <pre>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.</pre>	
Erreichte Punktzahl für diese Aufgabe	von 6 Punkten

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 );
}

```

Erreichte Punktzahl für diese Aufgabe

von 10 Punkten