Probe-Klausuraufgaben GDI

Dipl.-Ing. Klaus Knopper

06.01.2006

Viel Erfolg!

1.1	1.2	1.3	1.4	1.5	2.1	3.1	4.1	5.1	6.1	6.2	7.1	7.2	$\sum$	Bemerkungen

(bitte ignorieren)

1 Grundlagen

1.1 Ordnen Sie den folgenden Aussagen eine Instanz zu, die durch ein Buchstabenkürzel anzugeben ist. (6 Punkte)

Präprozessor (**P**), Compiler (**C**), Assembler (**A**), Linker (**L**), Entwicklungsumgebung (**E**) oder Keine (Feld leer lassen).

◯ kann weitere Quelltexte (z.B. Header-Dateien) in den aktuellen Quelltext einfügen.

◯ führt Textersetzungen aus (Makros),

◯ erzeugt aus dem architekturunabhängigen C-Quelltext architektur/prozessorabhängigen Assemblercode.

◯ übersetzt architektur/prozessorabhängigen Assemblerquelltext zu Objektdateien.

◯ untersucht den C-Quelltext auf syntaktische Fehler und meldet diese.

◯ korrigiert Fehler im Programmcode,

◯ integriert Editor, Übersetzungshilfen und Programmausgabe/Debugger (schrittweises Ausführen von Programmcode) in einer komfortablen Umgebung.

◯ kann, durch Anweisungen im Quelltext gesteuert, bestimmte Programmteile vor dem C-Compiler ”verstecken” oder sichtbar machen.

◯ bindet Objektdateien mit Systembibliotheken zu ausführbaren Programmen.

◯ stellt Hilfen zum leichten Auffinden von Klassen-Deklarationen und Funktionen, die auch auf mehrere Quelltextdateien verteilt sein können, zur Verfügung.

◯ erleichtert mit Hilfe von Syntax-spezifischen Hervorhebungen die Lesbarkeit von Quelltexten.

◯ beweist die Fehlerfreiheit des ausführbaren Programmes.

1.2 Welche der folgenden Aussagen trifft auf auf das gezeigte C-Programmfragment zu? (2 Punkte)

```
int counter()
{
 static int i=0;
 return i++;
}
```

(Nur eine Antwort ist richtig.)

- ◯ Die Variable **`i`** bleibt immer konstant 0.
- ◯ Die Variable **`i`** wird bei jedem Aufruf von `counter()` auf 0 gesetzt.
- ◯ `counter()` gibt bei jedem Aufruf eine größere Zahl zurück, bis der Wertebereich von `int` überschritten wird.
- ◯ Das Programm lässt sich wegen eines Syntaxfehlers in der Funktion `counter()` nicht übersetzen.

1.3 Welche der folgenden Aussagen trifft auf auf das gezeigte C-Programmfragment zu? (2 Punkte)

```
int main()
{
 char *zeiger;
 *zeiger = 0;
 printf("%s\n", zeiger);
 ...
```

(Nur eine Antwort ist richtig.)

- ◯ Die Zeichenkette `zeiger` zeigt auf die Adresse 0.
- ◯ Die Zuweisung `*zeiger = 0;` überschreibt hier ein Zeichen an zufälliger Stelle im Hauptspeicher, was zum Absturz des Programms führen kann.
- ◯ Die Anweisung `printf("%s\n", zeiger);` gibt hier die Zahl ”0” aus, da eine Umwandlung von `char` nach `int` erfolgt.
- ◯ Die Anweisung `printf("%s\n", zeiger);` funktioniert nicht, da die Ausgabe von Daten vom Typ **`char*`** nicht definiert ist.

2 Makros und Typumwandlungen

2.1 Was gibt das folgende C-Programm aus? (6 Punkte)

```
#include<stdio.h>

int zweimal(int x)
{
   return x + x;
}

#define ZWEIMAL(x) (2*x)

int main()
{
   int i = 2;
   double d = 0.5;
   printf("%d, ", zweimal((int)d));
   printf("%d, ", ZWEIMAL(i+4));
   printf("%d, ", zweimal((int)(d* 2.0)));
   printf("%d, ", ZWEIMAL(1*i)+1);
   return 0;
}
```

Programmausgabe: _____

3 Zeichenketten, Felder und Indizes

3.1 Was gibt das folgende Programm aus? (6 Punkte)

```
#include<stdio.h>

int main()
{
 char a[14] = "Hello, World!";
 char b[14] = "Hello, World!";
 int i;

 if(a == b)
   a[1] = b[1];
 else
   a[1] -= 4;

 for(i=1; i<3; i++) a[7+i] = b[i];
 a[10] = 't';
 a[11] = b[12];
 a[12] -= a[12];

 printf("%s\n", a);
 return 0;
}
```

Programmausgabe: _____

4 Zeiger

4.1 `invert`-Funktion (4 Punkte)

Gesucht ist eine C-Prozedur ohne Rückgabewert, die eine `int`-Variable aus dem aufrufenden Programm wie folgt verändern soll:

- `Variable` auf `0` setzen, falls `Variable` vorher ungleich `0` war,
- `Variable` auf `1` setzen, falls `Variable` vorher `0` war.

Die Änderung soll sich auf die Variable im aufrufenden Programm auswirken. Kreuzen Sie alle richtigen Antworten an.

Bewertungsschema:
Es können mehrere Antworten richtig sein. Falsch angekreuzte Antworten oder fehlende Kreuze bei richtigen Antworten führen zu einem Punkt Abzug, solange die erreichte Punktzahl dieser Aufgabe nicht kleiner als 0 wird.

◯ `int invert() { return 0 ? 1 : 0; }`

◯ `void invert(int b)  { if(b) b=0; else b=1; }`

◯ `void invert(int *b) { if(*b) *b=0; else *b=1; }`

◯ `void invert(int &b) { if(b) b=false; else b=true; }`

◯ `void invert(int &b) { b=b?false:true; }`

◯ `void invert(int *b) { *b = !*b; }`

◯ `void invert(int b) { b -= b; }`

5 Klassen, Klassenmethoden, Konstruktor

5.1 Was gibt das folgende Java-Programm aus? (6 Punkte)

```
class zeichen
{
 private char _z;
 zeichen(char z) {
  setze(z);
  System.out.println( z + "," + _z);
 }
 void setze(char z) { _z = ++z; }
}

public class Hauptprogramm {
 public static void main(String[] args)
 {
  zeichen z1 = new zeichen('A');
 }
}
```

Programmausgabe: _____

6 Finden Sie die Fehler

6.1 Programmfragment 1 (4 Punkte)

Welche Fehler (syntaktisch und semantisch) sind in folgendem C-Programmfragment vorhanden, das dazu dienen soll, einen Buchstaben daraufhin zu überprüfen, ob er ein Vokal oder Konsonant ist, und ggf. eine entsprechende Meldung an den Aufrufer zurückzugeben? Markieren Sie die entsprechenden Stellen durch Einkreisen.

```
boolean is_vowel(const char c)
{
 const int ja=1;    /* Vokal */
 const int nein=0;  /* kein Vokal */
 int antwort=nein;  /* Enthält den Rückgabewert */
 c = toupper(c);    /* c in Großbuchstaben wandeln */
 if(c == 'A') Antwort = ja;
 else if(c == 'E') antwort = ja
 else if(c == 'I') antwort = ja;
 else if(c == 'O') antwort = ja;
 else if(c == 'U') antwort = ja;

}
```

6.2 Programmfragment 2 (4 Punkte)

Das angegebene C-Programm soll zehn Messwerte einlesen und den Mittelwert berechnen. Abgesehen davon, dass es nicht besonders effizient geschrieben ist, enthält es auch einige Fehler, die nicht nur das Übersetzen verhindern, sondern auch das Ergebnis verfälschen. Markieren Sie die Fehler wieder durch Einkreisen.

```
#include<stdio.h>

int main()
{
 double feld[10];
 double summe = 0.0;
 unsigned int anzahl = 0;
 for(int i=0; i!=10; i++)
 {
  scanf("%lf", &feld[i]);
  summe = feld[i];
  ++anzahl;
 }
 printf("%d Messwerte wurden eingelesen.\n", anzahl);
 printf("Der Mittelwert beträgt: %lf\n", summe/(double)anzahl);
 if(i == 10) return 0; else return 1;
}
```

7 Programmieraufgaben

7.1 Verkettete Listen (8 Punkte)

Gegeben sei folgende C-Datenstruktur:

```
struct node
{
  char *data;
  struct node *next;
};
```

a) Schreiben Sie eine C-Funktion, um an ein durch einen Zeiger als Parameter übergebenes Listenelement vom Typ `node` ein weiteres, das ebenfalls als Zeiger übergeben wird, anzufügen.

b) Schreiben Sie eine C-Funktion, die den Zeiger auf das nächste Listenelement in einer als Zeiger-Parameter übergebenen `node`-Struktur auf `NULL` setzt.

7.2 Rekursion (4 Punkte)

Gegeben sei die folgende, abschnittsweise definierte Funktion:

$$f(x) = \begin{cases} 1 & \text{für alle } x \leq 1 \\ f(x-1) + f(x-2) & \text{sonst} \end{cases}$$

Schreiben Sie eine rekursive C-Funktion, die eine Ganzzahl als Parameter erwartet, den Funktionswert der dargestellten Funktion berechnet und diesen als Ganzzahl zurückgibt.

Hinweis: Aufgaben in Probeklausuren tendieren mitunter zu einem etwas höheren Schwierigkeitsgrad als diejenigen in der tatsächlichen Klausur.