Klausur Grundlagen der Informatik

11.03.2010

Hinweise:

- Bearbeitungszeit: 120 Minuten
- es sind **keine** Unterlagen (Bücher, Vorlesungsmitschriften, Übungen), Taschenrechner, Mobiltelefone (ausschalten und in die Tasche legen) sowie Computer jeglicher Art erlaubt
- legen Sie beschriebene Lösungsblätter abgedeckt ab
- Zwischenschritte für die Berechnungen/Umstellungen sind mit anzugeben
- die Klausur ist **mit** Aufgabenzetteln abzugeben

Wählen Sie aus den Aufgaben 1 bis 6 genau 5 Aufgaben aus!

Aufgabe 0

(1 Punkt) Schreiben Sie Ihren Namen, Vornamen, Matrikelnummer und Studiengang auf jedes Blatt das in die Bewertung einbezogen werden soll! Ergänzen Sie diese Angaben auch auf der letzten Seite der Aufgabenzettel.

Aufgabe 1 – Zahlensysteme

1.1 (4 Punkte) Stellen Sie die Dezimalzahl (Basis: 10) 88 im Hexadezimalsystem (Basis: 16) und im Dualsystem (Basis: 2) dar.
1.2 (3 Punkte) Ordnen Sie die 3 Zahlen $1D6_{16}$, 101110101_2 und 561_{10} aufsteigend.
1.3 (6 Punkte) Multiplizieren und Subtrahieren Sie dual die Dualzahlen 11101 mit 1011.
1.4 (4 Punkte) Geben Sie für die Dezimalzahlen 0 und -17 die zugehörigen Dualzahlen im Einer- und Zweierkomplement an (8 Bit).

Aufgabe 2 – Variablen, Arrays und Funktionen in C

2.1 (3 Punkte) Definieren Sie ein mehrdimensionales Array mit 88 Zeilen und 77 Spalten in der Programmiersprache C für Fließkommazahlen.
2.2 (5 Punkte) Schreiben Sie einen C Codeabschnitt für das Array aus Aufgabe 2.1 der jedem Element die Spaltennummer zuweist, in dem sich das Element befindet.

Beispiel:

1	2		77
1	2		77
1	2		77
1	2		77

Geben Sie anschließend das mehrdimensionale Array so wie im Beispiel aus. Zwischen den Spalten lassen Sie bitte ein Leerzeichen frei.

2.3 (5 Punkte) Definieren Sie in der Programmiersprache C eine (strukturierte) Variable für maximal 99 Endverbraucherprodukte. Jedes dieser Produkte besitzt eine ganzzahlige EAN Nummer (European Article Number), eine Produktbezeichnung mit maximal 100 Zeichen für den Produktnamen, das Herstellungsjahr, die Verpackungseinheit (Stückzahl innerhalb der Verpackung) sowie den Preis für Endverbraucher.

2.4 (4 Punkte) Verwenden Sie zwei alternative C-Schleifen zur Darstellung des folgenden Programmabschnitts.

```
01   int i, j = 1;
02   scanf( "%d", &i );
03   while ( i > 0 ) {
04        j * = ( j + i + 5);
05        i --;
06   }
```

Aufgabe 3 – C Programm

(20 Punkte) Schreiben Sie in der Programmiersprache C ein vollständiges Programm, das ganzzahlige Messwerte (maximal 1000) des Benutzers auswertet. Dabei müssen folgende Anforderungen erfüllt sein:

- Verwenden Sie keine globalen Variablen!
- Innerhalb der `main()`-Funktion muss der Benutzer als Erstes ein Endeerkennungszeichen eingeben. Dies ist ein Messwert, der nicht in der Messwertreihe auftritt. (Hinweis: dieses Zeichen hat den gleichen Datentyp wie die Messwerte)
- Anschließend soll der Benutzer gültige Messwerte (nicht durch 4 teilbare Werte) eingeben. Gibt der Benutzer einen ungültigen Wert ein, z.B. 80, dann ist die letzte Eingabe zu wiederholen. Die Eingabe der Messwerte endet, wenn der Benutzer das Endeerkennungszeichen oder 1000 gültige Messwerte eingegeben hat.
- In einer separaten Funktion ist der Funktionswert f folgender Formel zu bestimmen:

$$f = \frac{2}{n} * \sum_{i=0}^{n-1} (y[i]^3 - y[i]^2)$$

- Bestimmen Sie in einer zweiten Funktion die Anzahl der positiven Messwerte. Die Funktion soll den Rückgabetyp void verwenden und das Ergebnis in einem übergebenen Zeiger speichern.
- Geben Sie in der `main()` Funktion den Funktionswert sowie die Anzahl der positiven Messwerte aus.

Aufgabe 4 – Minimalmaschine und FlipFlops

4.1 (15 Punkte) Übersetzen Sie das folgende Programmfragment in Befehle der in der Vorlesung behandelten Minimalmaschine MiMa. Sie können die in der Vorlesung vorgestellten Subtraktions-, Multiplikations- und Divisionsbefehle verwenden. Beachten Sie, dass n jeden möglichen Wert annehmen kann. Auch ist vorher eine semantisch äquivalente Umstellung des C-Codes möglich. Eine kurze Referenz der MiMa-Instruktionen finden Sie auf der nächsten Seite.

```
01   int n;
02   a = 0, b = 1, tmp;
03   if ( n > 0)
04        while ( n ) {
05             tmp = a;
06             a = b;
07             b = tmp + b;
08             n--;
09        }
```

Instr.	Bedeutung	Instr.	Bedeutung
ADD n	acc=acc + mem[n]	JMP a	springe zu a
AND n	acc=acc & mem[n]	JMN a	springe zu a, wenn acc < 0
OR n	acc=acc | mem[n]	EQL n	wenn acc==mem[n],dann acc:=-1; sonst acc:=0
XOR n	acc=acc^mem[n]	HALT	Prozessor anhalten
LDV n	acc=mem[n]	NOT	acc=~acc (bitweise Komplement)
STV n	mem[n]=acc	RAR	acc um 1 Bit nach rechts rotieren
LDC n	acc=n	SUB n	acc=acc - mem[n]
MUL n	acc=acc * mem[n]	DIV n	acc=acc / mem[n]

4.2 (5 Punkte) Erklären Sie anhand einer Skizze die Funktionsweise eines FlipFlops und nennen Sie zwei mögliche Anwendungen für diese Schaltung.

Aufgabe 5 – Fehlersuche und Container

5.1 (10 Punkte) Das angehängte Programm A (Zusatzblatt) enthält mehrere Fehler. Finden Sie sie und geben Sie jeweils eine Lösungsmöglichkeit an.
Hinweis: Das Programm berechnet per Zufall einige Punkte und sucht dann den Punkt mit minimalem Abstand zur Nutzereingabe. Arbeiten Sie direkt auf dem Zusatzblatt.

5.2 (10 Punkte) Implementieren Sie bitte für die angegebene Struktur `NODE` die beiden angegebenen Funktionen. Setzen und verwenden Sie dabei die beiden Zeiger `g_head` und `g_tail`.

- Einfügen eines neuen Knotens nach einem gegebenen (gültigen) Knoten
  `void insertAfter(char* newdata, Node *pParentNode)`
- Entfernen/löschen eines gegebenen Knotens
  `void removeNode(Node *pNodeToRemove)`

```
01  struct NODE {
02    struct NODE *previous;
03    char *data;
04    struct NODE *next;
05  };
06  typedef struct NODE Node;
07
08  Node *g_head = NULL;      /* Beginn der Liste */
09  Node *g_tail = NULL;      /* Ende der Liste */
10  /* Beispielfkt. für das Anfügen eines Knotens am Ende */
11  void insertBack(char *data) {
12    Node *n = (Node*)malloc(sizeof(Node));
13    n->data = data;
14    n->previous = g_tail;
15    n->next = NULL;
16    if (g_tail)
17      g_tail->next = n;
18    g_tail = n;
```

```
19
20    if (!g_head)
21      g_head = n;
22  }

23  /* Ausgabe der Liste */
24  void printNodes(Node *list, uint32_t max)
25  {
26     Node *node = list;
27     uint32_t i = 0;
28     while (node!=NULL || i<max) {
29        printf("%s\n",node->data);
30        node = node->next;
31        ++i;
32     }
33  }
```

Aufgabe 6 - Theroetische Informatik

6.1 (12 Punkte) Die hypothetische WHILE-Programmiersprache oops sei durch die folgende EBNF definiert:

```
Programm ::= varid <- ausdruck |
     IF bedingung THEN Programm ELSE Programm |
     WHILE bedingung DO Programm DONE |
     Programm; Programm

bedingung ::= ausdruck <= ausdruck

ausdruck ::= varid | 0 | 1 | succ(ausdruck)
```

Die Funktion `succ(x)` ist die Nachfolgerfunktion `(succ(x)==x+1)`.

Bitte schreiben Sie ein oops-Programm, das die Zahl in der Variablen `divident` durch die Zahl in der Variablen `divisor` teilt und das Ergebnis in der Variablen `quotient` ablegt. (Nehmen Sie dabei an, dass Dividend und Divisor größer Null sind.)

6.2 (8 Punkte) Geben Sie bitte an, ob die folgenden Aussagen richtig oder falsch sind:

a. Reguläre Ausdrücke erzeugen gerade die Klasse der Typ-2 Sprachen.
b. Deterministische und nichtdeterministische endliche Automaten erkennen die gleiche Sprachklasse.
c. Für die Sprachen der Chomsky-Hierarchie gilt die folgende Inklusionskette: $Typ-0 \subset Typ-1 \subset Typ-2 \subset Typ-3$
d. Jede primitiv rekursive Funktion kann iterativ dargestellt werden.
e. Ein Stack hat FIFO-Semantik.
f. Zur Sicherstellung der perfekten Rekonstruierbarkeit eines digital abgetasteten analogen Signals ist mindestens eine Abtastfrequenz nötig, die größer ist, als die größte im Signal enthaltene Frequenz (Abtast-Theorem).
g. Es ist eine bewiesene Tatsache, dass die Komplexitätsklassen *P* und *NP* nicht identisch sind.
h. Ein Mutex ist eine Primitive zur Synchronisation nebenläufiger Prozesse

Programm A (zur Aufgabe 5.1)

```
#include ... /* ausgeblendet */

#define MAX_POINTS 300
typedef struct POINT
{
    int x;
    int y;
} Point;
void createPoint(int x, int y)
{
    Point p;
    p->x = x;
    p->y = y;
    return p;
}
float distance(Point a, Point b)
{
    int dx = a.x - b.x;
    int dy = a.y - b.y;
    float d = sqrt(dx^2 + dy^2);
    return d;
}
int main()
{
    Point pp[MAX_POINTS];
    int16_t i = MAX_POINTS;
    float x, y;
    float d=0, dd;
    d = 0;
    dd = FLT_MAX; /* größtmöglicher Wert für den Datentyp float */

    printf("X-Koordinate:\n");
    scanf("%d", x);
    printf("Y-Koordinate:\n");
    scanf("%d", y);
    Point p = createPoint(x, y);
    srand(time(NULL));
    for (uint8_t cnt=1; cnt<=MAX_POINTS;)
        int x = rand()%100;
        int y = rand()%100;
        pp[cnt] = createPoint(x, y);

    while (i > 0) {
        d = distance(p, pp[i]);
        if (d < dd)
            dd == d;
        --i;
     }
    printf("Die minimale Distanz beträgt: %s\n", &dd);
}
```

Vorname:
Nachname:
Studiengang:
Matrikelnummer:

Aufgabe:	Punkte:		Bemerkung:
0	1		
1.1	4		
1.2	3		
1.3	6		
1.4	4		
2.1	3		
2.2	5		
2.3	5		
2.4	4		
3	20		
4.1	15		
4.2	5		
5.1	10		
5.2	10		
6.1	12		
6.2	8		

Gesamt:		
Übung:		
Note:		