

Prof. Dr. Wolfgang Effelsberg

A5, Raum B223  
68131 Mannheim  
Telefon: (0621) 181-2600  
Email: effelsberg@informatik.uni-mannheim.de

Marcel Busse

A5, Raum B221  
68131 Mannheim  
Telefon: (0621) 181-2616  
Email: busse@informatik.uni-mannheim.de

---

Programmierkurs in C für Bachelor IMI  
Frühjahrssemester 2007

Klausur  
31. August 2007

---

---

Hinweise

---

1. Überprüfen Sie bitte Ihr Klausurexemplar auf Vollständigkeit (7 Seiten).
2. Unterschreiben Sie die Klausur auf der Rückseite des letzten Blatts.
3. Bearbeiten Sie die Aufgaben *ausschließlich* auf dem Aufgabenblatt der jeweiligen Aufgabe. Benutzen Sie ggf. auch die Rückseite der Aufgabenblätter.
4. Schreiben Sie auf jedes Blatt, das bewertet werden soll, oben Ihren Namen und Ihre Matrikelnummer.
5. Verwenden Sie nur dokumentenechte Stifte (z. B. keinen Bleistift) und keine roten Stifte.
6. Es sind keine Hilfsmittel zugelassen.
7. Die Bearbeitungszeit beträgt 33 Minuten.

---

Korrekturzeile

Bitte *nicht* ausfüllen!

---

Aufgabe	1	2	3	Summe
Max. Punktzahl	11	12	10	33
Erreichte Punktzahl				

Name:

Matrikelnummer:

---

Aufgabe 1

11 Punkte

---

Aufgabe 1 a)

2 Punkte

Welche Werte haben die Variablen `a` und `c` nach Ausführung des folgenden C-Codes (mit kurzer Begründung)?

```
int a = 1;  
int* b = &a;  
int c = *b;  
*b = 2;
```

Name:

Matrikelnummer:

Aufgabe 1 b)

4 Punkte

Implementieren Sie die Standard-String-Funktion `char* strcat(char* s1, char* s2)` aus der C-Headerdatei `<string.h>`, die den String `s2` an den String `s1` anhängt und einen Zeiger auf `s1` zurückliefert. Verwenden Sie keine weiteren String-Funktionen wie bspw. `strlen()`. Sie können davon ausgehen, dass `s1` genügend Platz für die Aufnahme von `s2` bereitstellt.

Aufgabe 1 c)

2 Punkte

Geben Sie beispielhaft einen Aufruf der Funktion `strcat(char* s1, char* s2)` an.

Name:

Matrikelnummer:

Aufgabe 1 d)

3 Punkte

Der Zugriff auf Elemente eines zweidimensionalen Feldes kann sehr unterschiedlich erfolgen. Betrachten Sie am Beispiel des Feldes  $c$  die folgende Tabelle. Ergänzen Sie die fehlenden Tabellenzellen.

Zugriff auf...	1. Möglichkeit	2. Möglichkeit	3. Möglichkeit
1. Zeile, 1. Spalte	<code>**c</code>	<code>*c[0]</code>	
i. Zeile, 1. Spalte		<code>*c[i-1]</code>	<code>c[i-1][0]</code>
1. Zeile, j. Spalte	<code>*(*c+j-1)</code>		
i. Zeile, j. Spalte			<code>c[i-1][j-1]</code>

---

 Aufgabe 2
 

---

12 Punkte

Das Spiel 4-Gewinnt basiert auf einem vertikal aufgestellten  $n \times m$  großen Spielfeld, in das zwei Spieler, 'x' und 'o', abwechselnd ihre Spielsteine von oben hineinwerfen. Wir gehen vereinfachend davon aus, dass derjenige Spieler gewonnen hat, der als erster in einer Spalte vier direkt übereinander liegende Steine hat.

	o				
	x	x		o	
	x	o	o	x	
x	x	o	o	x	o

Abbildung 1: Beispielfeld für  $n = 5$  und  $m = 6$ 

Zur Repräsentation des Spielfelds verwenden wir ein **char**-Feld `feld` mit  $n$  Reihen und  $m$  Spalten, wobei der Inhalt des Felds in Zeile  $i$  und Spalte  $j$  durch `feld[i][j]` gegeben ist. Die Spielsteine werden durch die Zeichen 'x' und 'o' voneinander unterschieden.

Implementieren Sie eine Funktion **int** `gewinner(char** feld, int n, int m)`, die das übergebene Spielfeld analysiert und folgende Werte zurückliefert:

- 1, falls 'x' gewonnen hat,
- 2, falls 'o' gewonnen hat,
- 0, falls das Spielfeld voll ist, aber weder 'x' noch 'o' gewonnen hat
- -1 sonst

Name:

Matrikelnummer:

---

Aufgabe 3

10 Punkte

---

Für die Implementierung einer Warteschlange (Queue) mit Hilfe einer verketteten Liste verwenden wir folgende Datenstruktur:

```
typedef struct node {  
    int data;          /* Inhalt des Knotens */  
    struct node* next; /* Zeiger auf den Nachfolgerknoten */  
} queue_t;
```

```
queue_t* queue;
```

Die Warteschlange wurde bereits mittels folgender Anweisungen initialisiert:

```
queue = (queue_t*)malloc(sizeof(queue_t));  
queue->next = NULL;
```

Die Variable `queue` repräsentiert den Kopf der Warteschlange. Auf das erste Element kann daher über `queue->next` zugegriffen werden.

Aufgabe 3 a)

5 Punkte

Implementieren Sie eine Funktion `void insert(queue_t* queue, int wert)`, die einen `int`-Wert am Ende der Warteschlange einfügt.

Name:

Matrikelnummer:

Aufgabe 3 b)

5 Punkte

Implementieren Sie eine Funktion **int** `extract(queue_t* queue)`, die das erste Element am Anfang der Warteschlange entfernt und dessen `int`-Wert zurückliefert. Falls die übergebene Warteschlange leer ist, soll die Funktion den Wert `-1` zurückgeben.