C-ÜBUNGSAUFGABEN DYNAMISCHER SPEICHER 1

1)

Ein Programm enthält folgenden Quellcode:

```
int *i;
*i = 123;
```

Was passiert während der Laufzeit?
Testen Sie dieses kleine Programm!

2)

Welchen großen Nachteil hat dieses Programm:

```
#include "stdafx.h"
#include <stdlib.h>
#include "stdio.h"
#include <malloc.h>

int main(void){
  int *pint;
  int beenden=0;
  int summe = 0;

  while(beenden==0){
    pint = (int *) malloc (10000);
    if(pint!=NULL){
      summe = summe + 10000;
    }
   else
      beenden = 1;
     }

  printf("Insgesamt belegter Speicher= %d\n", summe);
}
```

3)

Versuchen Sie in einem Testprogramm einen möglichst großen Speicherbereich zu reservieren.
Falls das Betriebssystem diese angeforderte Speichermenge bereitstellen kann (bzw. nicht kann), soll eine entsprechende Meldug auf dem Bildschirm erscheinen.

4)

a) Schreiben Sie ein Testprogramm, in dem solange z.B. 1000 Byte-Blöcke vom Betriebssystem angefordert werden, bis das Betriebssystem keinen Speicher mehr bereitstellen kann.
Geben Sie dann die gesamte Summe des bis dahin bereitgestellten Speichers aus.

b) Wie kann man nach der Beendigung der Schleife z.B. auf den bei dem ersten Schleifendurchgang reservierten Speicherplatz zugreifen ?

c) Auf welchen dynamisch reservierten Speicherplatz kann man nach der Beendigung der Schleife zugreifen ?
Was versteht man wohl in diesem Zusammenhang unter einem Speicherleck ?

5)

Schreiben Sie ein Programm, das die Widerstandswerte einer elektrischen Schaltung in einem Feld verwaltet:
Sie geben über Tastatur die Anzahl der abzuspeichernden Widerstände ein.
Danach geben Sie die einzelnen Widerstandswerte über Tastatur ein.
Ermitteln Sie dann den Mittelwert der eingegebenen Widerstände.
Benutzen Sie dazu dynamisch allokierten Speicher.
Benutzen Sie den Debugger, um ein tieferes Verständnis für den dynamisch allokierten Speicherplatz zu bekommen.

6)

Entwickeln Sie ein Programm, welches Zahlen wie folgt kodiert:
a) Lesen Sie Zufallszahlen (je einschließlich zwischen 1 und 6) in einen dynamisch reservierten Speicherbereich.

b) Kopieren Sie diese Zahlen in einen anderen auch dynamisch reservierten Speicherbereich.

c) Die Zahlen des 2. dynamisch reservierten Speicherbereichs sollen dann in umgekehrter Reifenfolge in den ersten dynamisch reservierten Speicherbereich kopiert werden.

7)

Lösen Sie die früheren Übungsaufgaben, die sich auf Felder bezogen, mit dynamisch reserviertem Speicher.

8)

Demonstration unerlaubter Speicherzugriff (auf nicht reservierten Speicher):
a) Legen Sie in einem Programm zwei dynamisch reservierte Speicherbereiche (Datentyp char) mit den Anfangsadresse pa1 und pa2 an und initialisieren diese. Schauen Sie im Debugger nach, welche Adresse pa1 und pa2 hat. Verändern Sie dann das Programm so, daß man von dem Feld mit der niederen Adresse auf das Feld der höheren Adresse zugreifen kann. (Siehe Übungsaufgabenblatt zu Feldern) .

b) Machen Sie das gleiche mit zwei dynamisch reservierten Speicherbereiche mit dem Datentyp Integer.

9)

Schreiben Sie eine Funktion, die an das Ende eines Feldes (dynamischer Speicher) ein bestimmtes Element anfügt.

10) Funktionen zur dynamische Speicherverwaltung

10.1)

Schreiben Sie die Funktion vertauscheFelder (char feld1[], char feld2[]), die den Inhalt zweier Felder vertauscht, d.h. nach dem Aufruf dieser Funktion befindet sich der alte Inhalt von feld1 in feld2 und der alte Inhalt von feld2 in feld1.
Tipp:
Es nützt nichts, in der Funktion vertauscheFelder(...) lokal ein Feld zu deklarieren, da man bei der Deklaration eine Länge angeben muß. Diese kennt man aber nicht, da diese z.B. erst während der Laufzeit festgelegt werden kann.
Also in der Funktion Speicher reservieren mit malloc(...)
return true: wenn intern Speicher reserviert werden konnte

10.2) etwas schwieriger
Schreiben Sie die Funktion sortiereFeld (char feld[]), die ein Feld sortiert (nach dem ASCII-Code)

10.3)
a) Schreiben Sie die Funktion dynamischesArray(...), die eine Zahl an das Ende eines Speicherbereichs anfügt.
Beim 1. Aufruf der Funktion muss diese intern einen Speicherbereich reservieren.
Bei den nächsten Aufrufen dieser Funktion muss diese nachprüfen, ob dieser Speicherbereich durch das Anfügen dieser Zahlen ausgeschöpft wurde und evtl. neuen Speicherbereich reservieren, d.h. den alten Speicherbereich in den neuen kopieren und danach den alten freigeben.
Wenn kein Speicher mehr reserviert werden kann, gibt man dies in der Variablen fehler zurück (z.B. durch den Wert -1)

```
int *dynamischesArray(int *dynFeld, int wert, int *fehler)
```

Mögliche Aufrufe:

```
int main(){
     int *feldAnfang;
     int fehler;

     feldAnfang = dynamischesArray(NULL, 100, &fehler);
     feldAnfang = dynamischesArray(feldAnfang, 101, &fehler);
     feldAnfang = dynamischesArray(feldAnfang, 102, &fehler);

     return 0;
}
```

b) Erstellen Sie die Funktion printDynamischesArray(...), die die Elemente dieses Arrays auf dem Bildschirm ausgibt.

c) Erstellen Sie die Funktion deleteDynamischesArray(...), die das dynamsiche Array freigibt.

A4) Entwickeln noch andere Funktionen zu dynamischen Arrays:
a) Einfügen eines Elements an einer beliebige Stelle
b) Entfernen eines Elements an einer beliebige Stelle