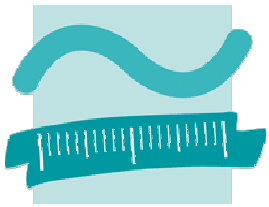
Dipl.Phys. Gerald Kempfer

Beuth Hochschule für Technik Berlin
- University of Applied Sciences -
Fachbereich VI – Informatik und Medien
Studiengang Technische Informatik Bachelor



Beispiel-Klausur zur Lehrveranstaltung IN2 TI-B (Informatik II) Sommersemester 2010

Hinweise:

Die Teilnahme an der Klausur ist nur bei Bestehen der Übungsaufgaben zulässig!

Zum Bestehen der Klausur benötigen Sie 50 Punkte von 100 möglichen Punkten.

Die Bearbeitungszeit der Klausur beginnt pünktlich um 10.00 Uhr und endet – ebenfalls pünktlich – um 11.30 Uhr. Sie haben also 90 Minuten Zeit.

Folgende Hilfsmittel in Papierform sind zugelassen: Skripte, Mitschriften und Bücher. **Nicht** zugelassen sind elektronische Geräte (Handys, Notebooks, PDAs, usw.).

Schreiben Sie mit Kugelschreiber oder Füller, **nicht** mit Bleistift! Verwenden Sie **nicht** die Farbe Rot. Schreiben Sie leserlich – was ich nicht lesen kann, ist grundsätzlich falsch! Beschreiben Sie nur die Vorderseiten!

Jeder Austausch mit anderen Personen wird als Täuschungsversuch gewertet und führt dazu, dass die Klausuren aller Beteiligten als „nicht bestanden“ gewertet werden.

Erläuterungen sollten kurz, aber dennoch präzise und vollständig sein. Wenn möglich ist eine stichpunktartige Beantwortung zu wählen, sofern die Verständlichkeit gegeben ist. Im Zweifelsfall können ganze Sätze Klarheit schaffen.

Kennzeichnen Sie jedes Blatt, das Sie abgeben, mit Ihrem Namen und / oder Ihrer Matrikelnummer.

Viel Erfolg!

Name: _____

Matrikelnummer: _____ letzter Prüfungsversuch: ☐

Bewertung:

Aufgabe	Mögliche Punktzahl	Erreichte Punktzahl
1	10	
2	15	
3	25	
4	25	
5	25	
Summe	**100**	

Note Klausur _____

Aufgabe 1: Finden Sie im folgenden Programm alle Syntaxfehler. (___ / 10)
Markieren Sie die Fehler und schreiben Sie hinter bzw. unter die Zeile, wie die Zeile richtig aussehen muss.

```
#include <stdio.h>
#inklude <stdlib.h>

#define DEBUG

int main()
{
   FILE Datei1, *Datei2;
   char Quelle[] = "datei1.txt";
   char *Modus = "w";
   int Zeichen1, Zeichen2;

   Datei1 = fopen(Quelle, "r");
   Datei2 = fopen('datei2.txt", Modus);

   if (Datei1 && Datei2)
   {
      while (! feof(Datei4))
      {
         Zeichen1 := fgetc(Datei1);
         if (Zeichen1 >= 0)
         {
            Zeichen2 = Zeichen1 + 17;
            Zeichen2 %= 256++;
            fputc(Zeichen2, Datei2);
            #ifdef DEBUG
               print("%c -> %c\n", Zeichen1, Zeichen2);
            #endif
         }
      endif
   }
   #ifndef DEBUG
   else
   {
      if (!Datei1)
         printf("%s konnte nicht geoeffnet werden!," Quelle);
      if (!Datei2)
         printf("datei2.txt konnte nicht geoeffnet werden!");
   }
   #endif

   fcloss(Datei1);
   fclose(Datei2);

   return 0;
}
```

Aufgabe 2: Multiple-Choice-Fragen. (___ / 15)
Kreuzen Sie an, wie der Satz richtig heißen muss. Es gibt immer nur eine richtige Antwort.

Die Definition einer Struktur wird mit folgendem Schlüsselwort eingeleitet:
- ❑ `define structure`
- ❑ `structure`
- ❑ `struct`

Ein Zeiger (mit dem Namen `Z`) auf eine Zeichenkette (Länge 10 Zeichen) wird wie folgt definiert:
- ❑ `char Z[11];`
- ❑ `char * * Z[11];`
- ❑ `char * Z[11];`

Eine Zeiger-Variable auf einen Datenstrom hat in C den Datentyp
- ❑ STREAM *.
- ❑ FILESTREAM *.
- ❑ FILE *.

Der Präprozessor-Befehl `#include <stdio.h>` fügt die Headerdatei `stdio.h` ein aus dem
- ❑ Include-Verzeichnis.
- ❑ Windows-Systemverzeichnis.
- ❑ aktuellen Verzeichnis.

Mit `int **Zeiger;` wird
- ❑ ein Zeiger auf `int` definiert.
- ❑ beim Compilieren ein Fehler auftreten.
- ❑ ein Zeiger auf Zeiger auf `int` definiert.

Der Name einer Funktion
- ❑ ist ein Zeiger auf eine Variable.
- ❑ ist eine Variable.
- ❑ ist ein Zeiger auf die Funktion.

Mit `int const * Zeiger;` wird folgendes definiert:
- ❑ Ein veränderbarer Zeiger auf eine unveränderbare Variable.
- ❑ Ein unveränderbarer Zeiger auf eine unveränderbare Variable.
- ❑ Ein unveränderbarer Zeiger auf eine veränderbare Variable.

Die Anweisung `free(NULL);` bewirkt
- ❑ einen Speicherzugriffsfehler.
- ❑ eine Fehlermeldung.
- ❑ nichts.

Eine make-Datei ist
- ❑ eine Text-Datei.
- ❑ binär verschlüsselt.
- ❑ Passwort-geschützt.

Die `malloc`-Funktion ist deklariert in der Headerdatei
- ❑ `stdio.h`
- ❑ `stdlib.h`
- ❑ `string.h`

Um den binären Lesemodus beim Aufruf der `fopen`-Funktion anzugeben, muss folgender Modus angegeben werden:
- ❑ `"rb"`
- ❑ `"r" | "b"`
- ❑ `"r" + "b"`

Mit folgender Funktion aus der Headerdatei `stdlib.h` wird dynamisch Arbeitsspeicher reserviert:
- ❑ `memalloc`
- ❑ `malloc`
- ❑ `alloc`

Dateien mit langen Dateinamen (ab Windows 95) können nur geöffnet werden mit der Funktion
- ❑ `flfnopen` (*lfn* steht für *long file name*).
- ❑ `fopen` (wie alle anderen Dateien auch).
- ❑ können gar nicht geöffnet werden.

Mit `#error` wird gezielt eine
- ❑ Präprozessor-Fehlermeldung erzeugt.
- ❑ Compiler-Fehlermeldung erzeugt.
- ❑ Linker-Fehlermeldung erzeugt.

In einer `struct`-Variablen
- ❑ können die verschiedenen Felder unterschiedliche Datentypen besitzen.
- ❑ müssen alle Felder den gleichen Datentypen besitzen.
- ❑ dürfen keine zwei Felder den gleichen Datentypen besitzen.

Aufgabe 3: Was gibt das folgende Programm aus? (___ / 25)
Schreiben Sie die Ergebnisse der vorgegebenen Ausdrücke sowie die Bildschirm-Ausgaben auf!

```
#include <stdio.h>

// Das folgende definiert "Funktionszeiger" als ein Zeiger auf
// Funktion, die ein char erhält und ein char zurückgibt:
typedef char (*Funktionszeiger)(char);

Funktionszeiger getFPtr(int);
void druckeZeichen(Funktionszeiger FPtr, char, int);

int main()
{
   Funktionszeiger FktPtr = NULL;

   druckeZeichen(getFPtr(1), 'A',  0);
   druckeZeichen(getFPtr(4), 'A', 17);
   druckeZeichen(getFPtr(2), 'b',  4);
   druckeZeichen(getFPtr(1), 'u', -1);
   druckeZeichen(getFPtr(0), 'f', -3);
   druckeZeichen(getFPtr(2), 'x', -3);
   druckeZeichen(getFPtr(3), 'x',  7);
   printf("\n");
}

char nextChar(char c)   {     return c + 1;     }
char prevChar(char c)   {     return c - 1;     }
char firstChar(char c)  {     return 'a';       }

Funktionszeiger getFPtr(int FktNr)
{
   switch (FktNr)
   {
      case 1: return nextChar;
      case 2: return prevChar;
      case 3: return firstChar;
   }
   return NULL;
}

void druckeZeichen(Funktionszeiger FPtr, char Zeichen, int Offset)
{
   if (FPtr)
      printf("%c", FPtr(Zeichen) + Offset);
}
```

Zwischenergebnisse:

	zeigt auf Funktion	Zeichen	`FPtr(Zeichen)`	Offset	`FPtr(Zeichen) + Offset`
`getFPtr(1)`		`'A'`		`0`	
`getFPtr(4)`		`'A'`		`17`	
`getFPtr(2)`		`'b'`		`4`	
`getFPtr(1)`		`'u'`		`-1`	
`getFPtr(0)`		`'f'`		`-3`	
`getFPtr(2)`		`'x'`		`-3`	
`getFPtr(3)`		`'x'`		`7`	

Hinweis: Die Spalten `FPtr(Zeichen)` und `FPtr(Zeichen) + Offset` nur dann ausfüllen, wenn der Funktionszeiger kein `NULL`-Zeiger ist!

Bildschirm-Ausgabe des oben stehenden Programms:

Aufgabe 4: Schreiben Sie die drei fehlenden Funktionen zum vorgegebenen Hauptprogramm.
Die Funktion `liesDatei` soll die Textdatei mit dem angegebenen Dateinamen einlesen und in dem angegebenen Array von Zeichenketten (übergeben wird die Adresse vom Zeiger auf das Array von Zeichenketten) ablegen. Der Speicherbereich für das Array sowie für die einzelnen Zeichenketten muss zuvor noch reserviert werden. Es kann eine korrekt-formatierte Textdatei vorausgesetzt werden!
Die Funktion `sortiere` soll das angegebene Array von Zeichenketten mittels der BubbleSort- und der Vergleichsfunktion sortieren.
Schließlich muss noch die Vergleichsfunktion erstellt werden, die zwei Zeichenketten erhält und diese miteinander vergleicht (z.B. mit `strcmp`). (___ / 25)

```
#include <stdio.h>
#include <string.h>
#include <malloc.h>

#define MAX 10
#define MAXLEN 100

void BubbleSort(char **Array, int Anzahl, int (*Vergleich)(char *, char *))
{  int i, j;
   char temp[MAXLEN];

   for (i = 1; i < Anzahl; i++)
      for (j = Anzahl - 1; j >= i; j--)
         if (Vergleich(*(Array + j), *(Array + j - 1)) < 0)
         {  strcpy(temp, *(Array + j));
            strcpy(*(Array + j), *(Array + j - 1));
            strcpy(*(Array + j - 1), temp);
         }
}

void schreibeDatei(char *Name, char **AZ)
{  FILE *D;
   int i;

   D = fopen(Name, "w");
   if (D != NULL)
      for (i = 0; i < MAX; i++)
         fprintf(D, "%s\n", *(AZ + i));
   free(AZ);
}

// Platz für Ihre Funktionsdeklaration (3 Punkte):









int main()
{  char **ArrayZeiger;

   liesDatei("unsortiert_char.txt", &ArrayZeiger);
   if (ArrayZeiger != NULL)
   {
      sortiere(ArrayZeiger);
      schreibeDatei("sortiert_char.txt", ArrayZeiger);
   }
}

// Platz für Ihre Funktionsdefinitionen ist auf dem nächsten Blatt!
```

Funktionsdefinitionen für Aufgabe 4 (22 Punkte):

Aufgabe 5: Schreiben Sie die drei fehlenden Funktionen zum vorgegebenen Hauptprogramm.
Die Funktion `ErzeugeEintrag` soll den notwendigen Speicher für einen neuen Wörterbucheintrag reservieren. Dazu gehört auch der Speicher für die beiden Wörter, die als Parameter übergeben und nach dem Speicher-Reservieren in den Eintrag kopiert werden sollen. Zurückgegeben wird ein Zeiger auf den reservierten Speicherbereich des neuen Eintrags.
Die Funktion `HaengeEintragAn` soll den angegebenen Eintrag (2. Parameter) am Ende der einfach verketteten Liste des angegebenen Wörterbuches (1. Parameter) anhängen.
Die Funktion `LoescheEintrag` soll den Eintrag mit dem angegebenen, deutschen Wort (2. Parameter) aus der verketteten Liste des angegebenen Wörterbuches (1. Parameter) wieder entfernen. Dazu gehört auch das Freigeben der dazugehörigen Speicherbereiche. (___ / 25)

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct WBE
{  char *Deutsch;  // dt. Wort
   char *Englisch; // engl. Wort
   struct WBE *Next;
} Eintrag;

typedef struct
{  char *Titel;
   int AnzahlEintraege;
   Eintrag *Erster;  // erster Eintrag in verketteter Liste
   Eintrag *Letzter; // letzter Eintrag  "      "       "
} Woerterbuch;

Woerterbuch *ErzeugeWoerterbuch(char *Titel);
void DruckeWoerterbuch(Woerterbuch *W);
void DruckeEintrag(Eintrag *E);

// Platz für Ihre Funktionsdeklarationen (3 Punkte):









int main()
{  Woerterbuch *DeutschEnglisch;

   DeutschEnglisch = ErzeugeWoerterbuch("Deutsch-Englisch-Woerterbuch");

   HaengeEintragAn(DeutschEnglisch, ErzeugeEintrag("Tisch", "table"));
   HaengeEintragAn(DeutschEnglisch, ErzeugeEintrag("Stuhl", "chair"));
   HaengeEintragAn(DeutschEnglisch, ErzeugeEintrag("Lampe", "lamp"));
   HaengeEintragAn(DeutschEnglisch, ErzeugeEintrag("Tuer", "door"));

   DruckeWoerterbuch(DeutschEnglisch);

   LoescheEintrag(DeutschEnglisch, "Lampe");
   LoescheEintrag(DeutschEnglisch, "Tuer");
   LoescheEintrag(DeutschEnglisch, "Stuhl");
   LoescheEintrag(DeutschEnglisch, "Tisch");

   Free(DeutschEnglisch->Titel);
   Free(DeutschEnglisch);
```

```
}

Woerterbuch *ErzeugeWoerterbuch(char *Titel)
{  Woerterbuch *Neu = malloc(sizeof(Woerterbuch));

   if (Neu)
   {  Neu->Titel = malloc(strlen(Titel) + 1);

      if (Neu->Titel)
         strcpy(Neu->Titel, Titel);
      Neu->AnzahlEintraege = 0;
      Neu->Erster = Neu->Letzter = NULL;
   }
}

 void DruckeWoerterbuch(Woerterbuch *W)
{  Eintrag *Temp;

   if (W)
   {  printf("\n\n%s\n\n", W->Titel);
      printf("%i Eintraege:\n\n", W->AnzahlEintraege);
      printf("Deutsch             -> Englisch\n");
      printf("----------------------------------------\n");
      Temp = W->Erster;
      while (Temp)
      {  DruckeEintrag(Temp);
         Temp = Temp->Next;
      }
      printf("----------------------------------------\n\n");
   }
}

void DruckeEintrag(Eintrag *E)
{  printf("%-20s-> %-20s\n", E->Deutsch, E->Englisch);
}

// Platz für Ihre Funktionsdefinitionen (22 Punkte):
```

Fortsetzung der Funktionsdefinitionen für Aufgabe 5: