

Prüfungsleistung: PROGRAMMIEREN 2

Datum: 14.7.2000, 8.00 Uhr Raum: 1101 Semester: E2A Prüfungsdauer: 90 min

zugelassene Hilfsmittel: Skripten der *FH Hamburg* zum Thema Programmieren in C.
Buch: *Kernighan, B.W.; Ritchie, D.M.: Programmieren in C, ANSI C.*
eigene schriftliche Unterlagen.

Diese Klausur besteht aus **6 Aufgaben auf 1 Blatt. Maximal 108 Punkte** können erreicht werden.

Zum Bestehen der Prüfung sind **mindestens 40 Punkte** erforderlich. Die Klausur enthält 8 Punkte mehr als normal. Diese zusätzlichen 8 Punkte benötigen Sie nicht, um die Note 15 zu erhalten.

Name, Vorname	(Bitte Name, Vorname und Matrikelnummer eintragen!)		
Matrikel- Nr.			
	Tag der Bewertung	Punktzahl	NOTE

Den **Status der Klausurkorrektur und Ihr Prüfungsergebnis** können Sie aus dem *Internet* unter <http://ds2.etech.fh-hamburg.de/~vogelsang> abrufen.

Dort ist auch der **Termin für die Einsichtnahme / Rückgabe der Klausuren** angegeben.

Die Ergebnisse stehen wahrscheinlich **nach dem 30.8.2000** zur Verfügung.

Einspruch gegen das Klausurergebnis ist nur am Rückgabetermin möglich.

Klausuren, die am Termin der Einsichtnahme nicht abgeholt worden sind, hinterlege ich im Prüfungsamt, Raum 1207.

Aufgabe 1: (15 Punkte)

Felder und Zeiger

Gegeben sind folgende Ausdrücke in der Programmiersprache C:

- 1.1. `long (*a [5]) ( double m ) ;`
- 1.2. `char b ( long *n [ ] ) ;`
- 1.3. `double *c ( long o [ ] ) ;`
- 1.4. `long ( *d ) ( short *p ) ;`
- 1.5. `long double *(*j [3]) ( char *v ) ;`

- 1.1.1 Erläutern Sie jede der obigen Anweisungen: Um welche Typen bzw. Operatoren handelt es sich?
Beachten Sie dabei die Vorrangbedingungen (*priority list*) der Operatoren in C!

Aufgabe 2: (8 Punkte)

mehrdimensionale Felder, Feldoperationen

Gegeben ist das folgende C-Programm mit einem 2-dimensionalen Feld:

```
#include <stdio.h>
int main (void)
{
    int i;
    short feld [4] [5]= { {1,2,3,4,5}, {10, 20, 30, 40}, {100,0,200}, {1000} };
    short *p = &feld [0][3], f;
    for (i = 0; i <4; i++)
    {
        f = *(p + 3 + 2*i);
        printf ("i = %d: f =%d\n", i, f);
    }
    return 0;
}
```

- 2.1 Geben Sie an, welche Bildschirmausgabe das Programm erzeugt!

Aufgabe 3: (12 Punkte)

Fehlerausgabe mit Zeigerfeldern

Gegeben ist folgender Programmausschnitt:

```
#include <stdio.h>
unsigned char Daten_verarbeiten ( char *name, int array [ ] );
/*Prototyp der Funktion Daten_verarbeiten, die Fehler als unsigned char zurückgibt.
Daten_verarbeiten verarbeitet Daten aus int array [ ] und greift dabei auf die Datei mit dem
Namen * name und auf dynamischen Speicher zu */
void Fehlerausgabe( unsigned char ); /* Prototyp der Funktion Fehlerausgabe */
int main (void) /* Hauptprogramm */
{
    int Feld [1000]; /* Variablen vereinbaren und ggf. initialisieren */
    char *Dateiname = "c:\\datei.dat" ;
    unsigned char Fehler ;
    Fehler = Daten_verarbeiten ( Dateiname, Feld ) ;
    Fehlerausgabe ( Fehler ) ;
    return 0 ;
}
```

Es soll die möglichst einfache Funktion **void Fehlerausgabe (error)** geschrieben werden, die abhängig von der **int**-Variablen **Fehler** folgende Fehlermeldungen ausgibt:

Fehler	Fehlermeldung
0	
1	Fehler: Überschreiten der Feldgrenzen!
2	Fehler in malloc: Zu wenig Speicherplatz vorhanden!
3	Fehler beim Öffnen: Datei ist nicht vorhanden!
>=4	Unzulässige Fehlernummer in Funktion 'Fehlerausgabe' aufgetreten!

Bei **Fehler** = 0 ist kein Fehler aufgetreten. Es erfolgt keine Fehlermeldung!

3.1 Schreiben Sie die Funktion **void Fehlerausgabe (error)**! Verwenden Sie ein Zeigerfeld!

Aufgabe 4: (28 Punkte)

Funktionen und Zeiger

Es soll ein Programm geschrieben werden, welches die Quadratwurzel und das Quadrat der einzulesenden **unsigned short**-Variablen **kurz** berechnet. Das Programm verwendet ein Zeigerfeld auf die Funktionen wurzel_2 (..) und hoch_2 (..), welche die Quadratwurzel bzw. das Quadrat von **kurz** berechnen. Der Typ des Funktionsrückgabewertes soll so gewählt werden, dass kein Zahlbereichsüberlauf auftreten kann und das Ergebnis möglichst korrekt dargestellt wird!

Bei der Berechnung der Quadratwurzel können sie die Bibliothek <math.h> verwenden.

Die Art der Operation wird ebenfalls folgendermaßen eingelesen:

'w' : Die Funktion **wurzel_2** mit dem Rückgabewert **kurz^{1/2}** wird aufgerufen

'2' : Funktion **hoch_2** mit dem Rückgabewert **kurz²** wird aufgerufen

4.1 Schreiben Sie das Programm mit den Unterprogrammen **wurzel_2** und **hoch_2**!

Aufgabe 5: (20 Punkte)

Programmfehleranalyse, dynamischer Speicher

Das folgende C-Programm enthält formale, logische und syntaktische Fehler!

Zur Vereinfachung der Fehleranalyse sind die Zeilen nummeriert.

Der Programmierer wollte erreichen,

- dass mit Hilfe von Zeigeroperationen eine Typumwandlung vorgenommen wird.
- dass die in das **long**-Format gewandelte Zahl im Heap gespeichert wird.

```
1. include <stdio.h> ;
2.
3. int main (void)
4. {
5.     float *s_p ;
6.     void *m_pointer ;
7.     unsigned long s1 = -1234.98 ;
8.     m_pointer = malloc (sizeof(s_p)) ;
9.
10.    s_p = &s1 ;
11.    m_pointer = *s_p ;
12.    printf ("float s1 = %f, long *m_pointer = %ld", s1, *m_pointer) ;
13.    return 0 ;
14. }
```

5.1 Untersuchen Sie das Programm auf logische, syntaktische Fehler und Formatfehler! Erläutern Sie bei jedem Fehler die Fehlerursache!

5.2 Korrigieren Sie das Programm!

Aufgabe 6: (25 Punkte)

Strukturen

Es soll ein Hauptprogramm geschrieben werden, das folgende Operationen vornimmt:

- Definieren Sie die Struktur **POINT** für die Koordinaten eines Bildpunktes im Grafikmodus des Bildschirms. Es wird eine Auflösung von 640*480 Punkten verwendet. Die Komponenten sollen das kürzest mögliche geeignete Ganzzahlformat aufweisen.
- Verwenden Sie die Struktur **POINT**, um die Struktur **QUADRAT** zu definieren: Die Struktur **QUADRAT** verwendet die Struktur **POINT**, um mit Hilfe von 2 benachbarten (nicht gegenüberliegen!) Punkten ein Quadrat zu definieren.
- Schreiben Sie die Funktion **Quadrat_Flaeche**, die als Eingangsvariable eine Variable des Typs **QUADRAT** verwendet und als Funktionsrückgabewert die Fläche des Quadrats zurückgibt! Wählen Sie für die Fläche des Quadrats ein geeignetes Zahlenformat, bei dem keine Zahlbereichsüberläufe auftreten!
- Lesen Sie im Hauptprogramm mit **scanf(..)** die 2 Punkte einer Struktur vom Typ **QUADRAT** von der Tastatur ein, rufen Sie die Funktion **Quadrat_Flaeche** auf und geben Sie die berechnete Fläche auf dem Bildschirm aus!