

PR3 - Übungsblatt A.4

(Stand 2021-10-13 15:38)

Prof. Dr. Holger Peine
Hochschule Hannover
Fakultät IV – Abteilung Informatik
Raum 1H.2.60, Tel. 0511-9296-1830
Holger.Peine@hs-hannover.de

Thema

Zeiger

Termin

Ihre Arbeitsergebnisse zu diesem Übungsblatt führen Sie bitte bis zum 05.11.2021 vor.

1 Alignment und Padding (0 Punkte)

basiert auf Vorlesung bis einschl. Abschnitt 5.b

Betrachten Sie die folgende Struktur:

```
struct beispiel
{
    short i;
    int j;
    char s[10];
};
```

- Beschreiben Sie das Speicher-Layout dieser Struktur: Wo liegen die einzelnen Elemente der Struktur im Speicher? Hinweis: Es gibt keine allgemeingültige Antwort, weshalb Sie das Speicher-Layout mit einem geeigneten Testprogramm auf Ihrem Rechner untersuchen sollen. Für die Abnahme ist wichtig, dass sich Ihre Antwort auf dem Vorfürrechner verifizieren lässt.
- Können Sie die Struktur so verändern, dass weniger Speicher benötigt wird, aber die gleichen Datentypen gespeichert werden?
- In dem folgenden Programm (alignment.c) wird eine Variable vom Typ `struct beispiel` als Char-Array interpretiert.

```
void fill(char u[])
{
    /* Hier soll Ihr Code eingefuegt werden. */
}

int main(void)
{
    struct beispiel bsp;

    /* Die Struktur wird in ein Char-Array umgewandelt
       und der fill-Funktion uebergeben. */
    fill((char*)&bsp);
    printf("%d %d %s\n", bsp.i, bsp.j, bsp.s);
    return 0;
}
```

Implementieren Sie `fill`, d.h. schreiben Sie geeignete Werte in das char-Array) so dass sich folgende Programmausgabe ergibt:

```
89 32168 Rosi
```

2 Opake Datentypen (0 Punkte)

basiert auf Vorlesung bis einschl. Abschnitt 4.e

Bei der Definition einer Variablen eines Strukturtyps müssen Sie wie in den anderen Beispielen dieses Übungszettels das Schlüsselwort `struct` verwenden:

```
struct user u = { "", 0 };
```

Was können Sie tun, damit Sie das Schlüsselwort `struct` weglassen können?

```
user u = { "", 0 };
```

3 Funktion zum Tausch zweier Variableninhalte (1 Punkt)

basiert auf Vorlesung bis einschl. Abschnitt 5.a

Schreiben Sie eine Funktion `tausche_int()`, mit der ein Aufrufer den Inhalt zweier `int`-Variablen vertauschen kann wie im folgenden Programm:

```
#include <stdio.h>

void tausche_int(...)

int main(void) {

    int i = 1; int j = 2;

    printf("i = %d, j = %d\n", i, j); /* Gibt 1 und 2 aus */
    tausche_int(&i, &j);
    printf("i = %d, j = %d\n", i, j); /* Gibt 2 und 1 aus */

    return 0;
}
```

Optional:

Schreiben Sie dann eine Funktion `tausche_intPtr()` mit der ein Aufrufer den Inhalt zweier `int*`-Variablen vertauschen kann (Tipp: Zu *jedem* Typ *T* gibt es in C einen Typ *T** - auch dann, wenn *T* selbst schon ein Zeigertyp ist!). Bauen Sie einen Test für Ihre Funktion in das vorige Programm ein.

4 Array- und Zeiger-Code ergänzen (1 Punkt)

basiert auf Vorlesung bis einschl. Abschnitt 5.b

Betrachten Sie das Programm in der folgend abgedruckten Datei `main.c`, das das größte Element eines Arrays ermitteln soll.

```
/* main.c */
#include <stdio.h>
int main(void) {
    int a[] = { 1, 5, 19, -4, 3 };
    int* p;
    int i;

    /* Lasse p auf das 0-te Array-Element verweisen. */
    /* Ihr Code hier */
}
```

```

for (i=1; i<5; i++) {
    /* Prüfe, ob das Array-Element i größer als das von
       p referenzierte Element ist */
    if ( /* Ihr Code hier */ ) {
        /* Lasse p auf das Array-Element i verweisen */
        /* Ihr Code hier */
    }
}
/* Gib das von p referenzierte Element als das größte aus: */
printf("Maximum: %d\n", /* Ihr Code hier */ );
return 0;
}

```

Ergänzen Sie Ihren Code an den mit `/* Ihr Code hier */` markierten Stellen.

5 Array-Funktion erkennen (0 Punkte)

basiert auf Vorlesung bis einschl. Abschnitt 5.b

Was ist der Rückgabewert der folgenden Funktion? Erklären Sie, wie er zustande kommt.

```

char* foo(char* s1, char* s2) {
    char *start, *left, *right;
    for (start = s1; *start != '\0'; ++start) {
        for (left = start, right = s2;
            *left != '\0' && *right != '\0' && *left == *right;
            ++left, ++right)
            ;
        if (*right == '\0')
            return start;
    }
    return NULL;
}

```

6 Funktion zum Finden eines Zeichens in einem String (0 Punkt)

basiert auf Vorlesung bis einschl. Abschnitt 5.b

Schreiben Sie eine Funktion

```
char* find(char s[], char c)
```

die einen Zeiger auf das letzte(!) Vorkommen des Zeichens `c` im String `s` liefert, falls `c` in `s` vorkommt; falls nicht, soll die Funktion `NULL` liefern.

7 Array-Kopieren mit Zeigerarithmetik (1 Punkt)

basiert auf Vorlesung bis einschl. Abschnitt 5.b

Schreiben Sie ein Programm mit zwei Arrays `a` und `b`, die jeweils fünf `int`-Werte aufnehmen können. Das Array `a` soll unmittelbar bei seiner Definition mit Werten vorbesetzt werden. Das Programm soll dann die Werte von `a` nach `b` kopieren und dabei ihre Reihenfolge umkehren. Zum Abschluss soll es den Inhalt von `b` auf den Bildschirm ausgeben. Beim Zugriff auf die Arraykomponenten soll nicht die Schreibweise mit eckigen Klammern `[]`, sondern die Zeigerarithmetik benutzt werden.

8 Zeiger, Arrays, Funktionsaufruf (1 Punkt)

basiert auf Vorlesung bis einschl. Abschnitt 5.b

Gegeben seien die beiden folgenden Zahlen, die Sie bitte in einem Array vom Datentyp `time_t` (`#include <time.h>`) ablegen: 645703200, 49888800.

Geben Sie mit Hilfe der Bibliotheksfunktion `ctime` aus, um welches Datum es sich bei diesen zwei Zeitpunkten handelt. Mehr zur Bibliotheksfunktion `ctime` finden Sie z. B. hier: http://openbook.rheinwerk-verlag.de/c_von_a_bis_z/019_c_zeitroutinen_001.htm

Optional: Finden Sie mit Hilfe des Internet heraus, welche Bedeutung diese zwei Tage im Zusammenhang mit Ihrem Studium haben.

9 Zeiger (0 Punkte)

basiert auf Vorlesung bis einschl. Abschnitt 5.c

Betrachten Sie die folgenden Programme. Was wird jeweils ausgegeben?

```
// Aufgabe 9 a)
#include <stdio.h>
void main()
{
    int a[3] = {1, 2, 3};
    int *p = a;
    printf("%p %p", p, a);
}

// Aufgabe 9 b)
#include <stdio.h>
void main()
{
    char *s = "Hallo";
    char *p = s;
    printf("%p %p", p, s);
}

// Aufgabe 9 c)
#include <stdio.h>
void main()
{
    char *s = "Hallo";
    char *p = s;
    printf("%c %c", p[0], s[1]);
}

// Aufgabe 9 d)
#include <stdio.h>
void main()
{
    char *s = "Hallo";
    char *p = s;
    printf("%c %c", *(p + 3), s[1]);
}

// Aufgabe 9 e)
#include <stdio.h>
void main()
{
    char *s = "Hallo";
    char *p = s;
    printf("%c %c", 1[p], s[1]);
}
```

```

}

// Aufgabe 9 f)
#include <stdio.h>
void foo( int[] );
int main()
{
    int a[4] = {1, 2, 3, 4};
    foo(a);
    printf("%d ", a[0]);
}
void foo(int* p)
{
    int i = 10;
    p = &i;
    printf("%d ", p[0]);
}

// Aufgabe 9 g)
#include <stdio.h>
int main()
{
    int a[4] = {1, 2, 3, 4};
    int *p = a + 3;
    printf("%d\n", p[-2]);
}

//Aufgabe 9 h)
#include <stdio.h>
int main()
{
    int a[4] = {1, 2, 3, 4};
    int *p = a + 3;
    printf("%d %d\n", p[-2], a[*p]);
}

```

10 Operationen auf Zeichenketten (0 Punkte)

basiert auf Vorlesung bis einschl. Abschnitt **5.C**

Implementieren Sie die Funktionen `mystrlen`, `mystrcpy`, `mystrcat` und `mystrchr` in folgendem Programm (auch im Moodle). Dabei dürfen Sie keine Funktionen aus `string.h` verwenden. Hinweis: Statt der bisher genutzten Arrays sollen nun Zeiger verwendet werden, um über die Zeichenketten zu iterieren.

```

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

/*
 * The strlen() function calculates the length of the string pointed to by s,
 * excluding the terminating null byte ('\0').
 */
size_t mystrlen(const char *s) {
    /* Ihr Code. */
}

/*
 * The strcpy() function copies the string pointed to by src, including the
 * terminating null byte ('\0'), to the buffer pointed to by dest.
 * The strings may not overlap, and the destination string dest must be
 * large enough to receive the copy.
 * The strcpy() function returns a pointer to the destination string dest.
 */

```

```
char *mystrcpy(char *dest, const char *src) {
    /* Ihr Code. */
}

/*
 * The strcat() function appends the src string to the dest string,
 * overwriting the terminating null byte ('\0') at the end of dest, and then
 * adds a terminating null byte. The strings may not overlap, and the dest
 * string must have enough space for the result.
 * The strcat() function returns a pointer to the resulting string dest.
 */
char *mystrcat(char *dest, const char *src) {
    /* Ihr Code. */
}

/*
 * The strrchr() function returns a pointer to the last occurrence of the
 * character c in the string s. Here "character" means "byte".
 * The strrchr() function returns a pointer to the matched character or NULL if
 * the character is not found. The terminating null byte is considered part of
 * the string, so that if c is specified as '\0', these functions return a
 * pointer to the terminator.
 */
char *mystrrchr(char *s, int c) {
    /* Ihr Code. */
}

/*
 * The strstr() function finds the first occurrence of the substring needle in
 * the string haystack. The terminating null bytes ('\0') are not compared.
 * These functions return a pointer to the beginning of the located substring,
 * or NULL if the substring is not found.
 */
char *mystrstr(char *haystack, char *needle) {
    /* Diese Funktion ist nicht Teil der Aufgabe, kann aber implementiert werden. */
}

int main(void) {
    char s1[] = "Dies ist ein Teststring!";
    char s2[] = "Und dies noch einer.";
    char buffer1[2000];
    char buffer2[2000];

    printf("mystrlen:  %s\n",
           mystrlen(s1) == strlen(s1) ? "Richtig" : "Falsch");

    mystrcpy(buffer1, s1);
    printf("mystrcpy:  %s\n", strcmp(buffer1, s1) == 0 ? "Richtig" : "Falsch");

    mystrcpy(buffer1, s1);
    mystrcat(buffer1, s2);
    strcpy(buffer2, s1);
    strcat(buffer2, s2);
    printf("mystrcat:  %s\n", strcmp(buffer1, buffer2) == 0 ? "Richtig" : "Falsch");

    printf("mystrrchr: %s\n",
           mystrrchr(s1, 'i') == strrchr(s1, 'i') ? "Richtig" : "Falsch");

    printf("mystrstr:  %s\n",
           mystrstr(s1, "Test") == strstr(s1, "Test") ? "Richtig" : "Falsch");

    return 0;
}
```