

1 String-Konkatenation

Implementieren Sie `string_concat` im folgenden Programm so, dass die beiden übergebenen Strings `string1` und `string2` zusammenhängend in den String `out` geschrieben werden. Benutzen Sie für die eigentliche Konkatenation keine von der Standardbibliothek bereitgestellten Funktionen wie z. B. `sprintf`. Testen Sie auch Spezialfälle wie leere Strings.

Erinnerung: Ein String ist ein Array von `char`-Werten, das mit einem Null-Byte (`\0`) terminiert ist.

```
1 #include <stdio.h>
2
3 void string_concat (char out[], char string1[], char string2[])
4 {
5 }
6
7 int main (void)
8 {
9     char out[128] = { '\0' };
10    string_concat(out, "Hello", "World");
11    printf("%s\n", out);
12    return 0;
13 }
```

Abgabe

Abzugeben ist ein gemäß den bekannten Richtlinien erstelltes und benanntes Archiv. Das enthaltene und gewohnt benannte Verzeichnis soll folgenden Inhalt haben:

- Alle Quellen, aus denen Ihr Programm besteht (`exercise0.c`); gut dokumentiert (Kommentare im Code!)

Senden Sie Ihre Abgabe an `cp-abgabe@wr.informatik.uni-hamburg.de`.

2 Strukturen und Unions

Modifizieren Sie das folgende Programm so, dass auf oberster Ebene nur noch ein `struct foobar` deklariert wird, das die beiden Strukturen `struct foo` und `struct bar` enthält. Um den Speicherplatz gering zu halten, soll außerdem eine Union eingesetzt werden, so dass `struct foobar` maximal 8 Bytes groß ist.

```

1 #include <stdio.h>
2
3 struct foo
4 {
5     int a;
6     int b;
7 };
8
9 struct bar
10 {
11     float a;
12     float b;
13 };
14
15 int main (void)
16 {
17     struct foo a = { 42, 23 };
18     printf("foo: %d %d\n", a.a, a.b);
19     struct bar b = { 23.0, 42.0 };
20     printf("bar: %f %f\n", b.a, b.b);
21     return 0;
22 }

```

2.1 Enums

Passen Sie das gegebene Programm weiterhin so an, dass ein enum genutzt wird, um in struct foobar zu speichern, ob die Integer- oder die Float-Komponente aktiv ist. Die Größe von struct foobar darf sich dadurch erhöhen.