Selbststudium 03 GENERIC FUNCTIONS

Zürich, 22.03.2015

PROG C

Stefan R. Bachmann - bachmste

Grundidee

Ich habe angefangen mit den Pointern herumzuspielen. Auch habe ich diverse Dinge im Netz gelesen betreffend generischen Funktionen in C um im Anschluss eine generische PrinterFunktion zu kreiiren.

Leider war der Anfangsweg etwas steinig und es dauerte bis ich eine funktionierende Lösung hatte. Das erste und für mich ein erneuter Erstversuch in C, war das Auslagern des Typs in a) die Headerdatei und b) die Typen als ENUM zu speichern und definieren.

HeaderSnipplet

```
typedef enum {
    STRING, INT, CHAR, FLOAT
} type;

void print(void* source, int numElements, char type);
```

Pointer Switch

Zuerst habe ich mich entschlossen den voidPointer des Inputs je na case in den entsprechenden TypPointer zu casten. Als Mehrwert habe ich mir überlegt, auch einen STRING ausgeben zu können und jeweils anzugeben, wieviele Elemente jeweils im Array gespeichert sind.

Pointer Switch Snipplet

```
 case CHAR: {
            for (i = 0; i < numElements; i++) {
                (void)printf("%c from %d Elements\n", *(char *) (source+i), numEle-
ments); // Alternative zu (char*)source)[i]
            }
            break;
        }
        case INT: {
            for (i = 0; i < numElements; i++) {
                (void)printf("%d from %d Elements\n",((int *)source)[i], numElements);
            }
            break;
        }
        case FLOAT: {
            for (i = 0; i < numElements; i++) {
                (void)printf("%.2f from %d Elements\n", ((float *) source)[i], numEle-
ments);
            }
            break;
        }
        case STRING: {
            (void)printf("%s with %d Elements\n", (char *) source, numElements);
            break;
        }
        default:
            (void)printf("Error. Please try again. The silence will fall.");
```

Änderungen am Input in main

Durch die Anpassung des Typs in die ENUM Definition hat sich ergeben, dass der Input nicht mehr als 'c' sondern als CHAR (enum) eingegeben werden muss. Das hat lediglich Einfluss auf den Aufruf der Print Funktion.

Was auch zu sehen ist, ist das Array s[] mit dem gespeicherten String. In diesem wird dann gezählt, wie lang der Satz ist. Das ist nicht effektiv ein String, sondern zählt auch einfach die CHAR.

HeaderSnipplet

```
char c[] = { 'X', 'M', 'A' };
    int d[] = { 54, 10, 11, 125, 12, 0 };
    float f[] = { 1.23, 2.34, 3.45, 4.56 };
    char s[] = {"In Transelore the silence will fall, when the eleventh comes to his
end."};

    print((void *) c, sizeof(c) / sizeof(char), CHAR);
    print((void *) d, sizeof(d) / sizeof(int), INT);
    print((void *) f, sizeof(f) / sizeof(float), FLOAT);
    print((void *) s, sizeof(s) / sizeof(char), STRING);
```

Output der Funktion

Den Output habe ich so angepasst, dass jeweils gesagt wird, wieviele Elemente im gesammten im Array gespeichert sind. Ausserdem gibt es den ganzen String aus. Um mehrere Strings auszugeben, müsste ein zweidimensionales Array benutzt werden.

HeaderSnipplet

```
X from 3 Elements

M from 3 Elements

A from 3 Elements

54 from 6 Elements

10 from 6 Elements

11 from 6 Elements

125 from 6 Elements

12 from 6 Elements

0 from 6 Elements

1.23 from 4 Elements

2.34 from 4 Elements

3.45 from 4 Elements

4.56 from 4 Elements

In Transelore the silence will fall, when the eleventh comes to his end. with 73 Ele-
ments
```