6.5.A Extras/Funktionszeiger – Arbeitsblatt

Aufgabe 1: Funktionszeiger vereinbaren (definieren)

In den folgenden Programmbeispielen wird jeweils eine Funktion mithilfe eines Funktionszeigers aufgerufen. Es fehlt die Definition des Funktionszeigers. Bitte tragen Sie sie nach, so dass das Programmbeispiel funktioniert!

a) Beispiel 1:

```c
#include <stdio.h>
void rot(void)
{
    printf("rot\n");
}
int main(void)
{
    /* hier Vereinbarung einfuegen */
    funk=rot;
    funk();
    return 0;
}
```

b) Beispiel 2:

```c
#include <stdio.h>
int gelb(void)
{
    printf("gelb\n");
    return 2;
}
int main(void)
{
    int ergebnis;
    /* hier Vereinbarung einfuegen */
    funk=gelb;
    ergebnis=funk();
    printf("%i\n", ergebnis);
    return 0;
}
```

c) Beispiel 3:

```c
#include <stdio.h>
void gruen(int zahl)
{
    printf("gruen, %i\n", zahl);
}
int main(void)
{
    /* hier Vereinbarung einfuegen */
    funk=gruen;
    funk(3);
    return 0;
}
```

d) Beispiel 4:

```c
#include <stdio.h>
char *blau(void)
{
    return "blau";
}
int main(void)
{
    char *ergebnis;
    /* hier Vereinbarung einfuegen */
    funk=blau;
    ergebnis=funk();
    printf("%s\n", ergebnis);
    return 0;
}
```

Aufgabe 2: Ausgabe aller Zeichen einer Zeichenklasse

Die Funktionen `isalnum`, `isalpha` usw. geben an, ob ein Zeichen zu einer Zeichenklasse gehört. Sie sollen nun eine Funktion schreiben, die zu einer bestimmten Funktion (z. B. isdigit) alle Zeichen dieser Klasse ausgibt:

```c
printallezeichen(isdigit);
```

Dieser Aufruf soll die Zeichenkette `0123456789` ausgeben. Das geschieht sinnvollerweise so, dass man alle Zeichen von 32 bis 126 durchläuft und für jedes Zeichen die Funktion `isdigit` aufruft. Falls der Rückgabewert ungleich null ist, gibt man das Zeichen aus.

a) Wie lauten die Prototypen der Funktionen `isalnum`, `isalpha` usw.?

b) Wie muss eine Variable x vereinbart werden, die die Adresse einer solchen Funktion aufnehmen kann (Funktionszeiger)?

c) Wie lautet der Prototyp der Funktion `printallezeichen`, die einen solchen Funktionszeiger als Parameter hat?

d) Schreiben Sie die Definition der Funktion `printallezeichen` (siehe Struktogramm in Abbildung 1)! Testen Sie die Funktion mit dem oben genannten Aufruf (vergessen Sie nicht, `<ctype.h>` einzubinden)!

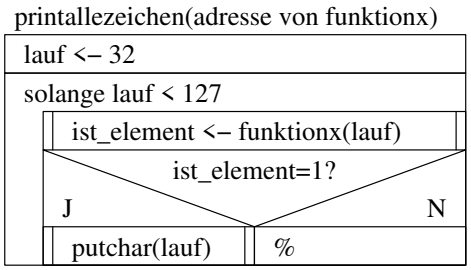

Abbildung 1: Struktogramm der Funktion `printallezeichen`

e) Testen Sie die Funktion `printallezeichen` mit einem Programm `print_az1.c`, in dem `printallezeichen` mit einer Ihnen bekannten Zeichenklasse aufgerufen wird!

f) Nun muss die Funktion `printallezeichen` für jede der elf Zeichenklassen aufgerufen werden, damit man weiß, welches Zeichen zu welcher Funktion gehört. Ergänzen Sie die `main`-Funktion des Programms (`print_az2.c`) entsprechend!

Aufgabe 3: Vergleichsfunktionen für `qsort`

Bei `qsort()` muss man in jedem Anwendungsfall eine Vergleichsfunktion schreiben. Schreiben Sie zu jedem der folgenden Anwendungsfälle die passende Vergleichsfunktion! Geben Sie zuerst den Prototypen an, dann die Funktionsdefinition!

a) Ein Array von positiven ganzen Zahlen soll in aufsteigender Reihenfolge sortiert werden:

```c
/* qsort1.c */
#define ANZAHL 5
int main(void)
{
    unsigned int lauf;
    unsigned int array[ANZAHL]={70,20,0,29,45};
    qsort(array, ANZAHL, sizeof(unsigned int), uintvergl);
    for(lauf=0; lauf<ANZAHL; ++lauf)
        printf("%i\n", array[lauf]);
    return 0;
}
```

b) Wie in der vorigen Aufgabe, nur diesmal soll in absteigender Reihenfolge sortiert werden (`qsort2.c` mit `uintvergl2()`).

c) Ein Array von Zeichen soll in aufsteigender Reihenfolge sortiert werden (aus `"zitrone"` wird dann `"einortz"`):

```c
/* qsort3.c */
int main(void)
{
    unsigned int lauf;
    char array[]="zitrone"; /* ...={'z','i','t','r','o','n','e'}; */
    qsort(array, strlen(array), sizeof(char), charvergl);
    for(lauf=0; lauf<strlen(array); ++lauf)
        printf("%c\n", array[lauf]);
    return 0;
}
```

d) Ein Array von Arrays von Zeichen soll sortiert werden, also ein Array von Strings. Auch hier soll aufsteigend sortiert werden:

```c
/* qsort4.c */
#define ANZAHL 5
#define SLEN 20
int main(void)
{
    unsigned int lauf;
    char array[ANZAHL][SLEN]={"apfel", "apfelsine", "acerola",
                              "ananas", "aprikose"};
    qsort(array, ANZAHL, SLEN, chararrvergl);
    for(lauf=0; lauf<ANZAHL; ++lauf)
        printf("%s\n", array[lauf]);
```

```
12      return 0;
13  }
```

e) Zusatzaufgabe, nicht einfach: Ein Array von Zeigern auf Zeichen soll sortiert werden, also ein Array von Strings. Auch hier soll aufsteigend sortiert werden. Achtung: Hier werden durch `qsort()` nicht die Zeichenketten-*Inhalte* vertauscht (und damit sortiert), sondern die *Zeiger* auf die Zeichenketten. Das ist wesentlich schneller und wird deshalb oft vorgezogen. Aber: die Vergleichsfunktion soll nicht die Zeiger der Größe nach vergleichen – denn da gewinnt immer der, der auf etwas zeigt, das weiter hinten im Speicher liegt. Sondern sie soll die *Inhalte* vergleichen, auf die die Zeiger zeigen.

```
1  /* qsort5.c */
2  #define ANZAHL 5
3  int main(void)
4  {
5      unsigned int lauf;
6      const char *array[ANZAHL]={"apfel", "zitrone", "ananas",
7                                 "apfelsine"};
8      qsort(array, ANZAHL, sizeof(char *), charptrvergl);
9      for(lauf=0; lauf<ANZAHL; ++lauf)
10         printf("%c\n", array[lauf]);
11     return 0;
12 }
```

f) Ändern Sie die vorige Aufgabe so ab, dass die Inhalte der Kommandozeilenparameter sortiert werden, dass also das Array `char *argv[]` sortiert wird (`qsort6.c`)!

g) Ein Array von Personen-Records soll sortiert werden, und zwar nach den Geburtstagen der Personen im Jahr. Das Geburtsjahr selbst soll dabei keine Rolle spielen.

```
1  /* qsort7.c */
2  #define ANZAHL 3
3  struct person_t{ char vn[20], nn[20]; int j,m,t;};
4  int main(void)
5  {
6      unsigned int lauf;
7      struct person_t array[ANZAHL]={{"Emil", "Meier",  1960,10,20},
8                                     {"Erwin","Mueller",1970,11,15},
9                                     {"Egon", "Moser",  1980,10,24}};
10     qsort(array, ANZAHL, sizeof(struct person_t), structvergl);
11     for(lauf=0; lauf<ANZAHL; ++lauf)
12         printperson(array[lauf]);
13     return 0;
14 }
```

Aufgabe 4: Weitere Funktionszeiger in der C-Standardbibliothek

In der C-Standardbibliothek sind neben `qsort` weitere Funktionen vorhanden, die Funktionszeiger benutzen.

a) Listen Sie die Funktionen auf!

b) Geben Sie zu jeder Funktion an, welche Headerdatei für die Funktion eingebunden werden muss!

Aufgabe 5: Werttabelle I

Die Funktion `printwtab()` soll eine Wertetabelle berechnen und ausdrucken. Der folgende Aufruf soll für alle x-Werte von 1 bis 3 (mit einer Schrittweite von 0,5) die entsprechenden Werte für `y=sin(x)` (im Bogenmaß) ausgeben:

```
1    printwtab(1,3,0.5,sin);
```

So soll ungefähr der Ausdruck aussehen:

```
  x   |  y
------+------
  1   | 0.841
  1.5 | 0.997
  2   | 0.909
  2.5 | 0.598
  3   | 0.141
```

a) Wie lautet der Prototyp der `sin`-Funktion aus `math.h`?

b) Wie muss eine Variable `fzeiger` vereinbart werden, die die Adresse der `sin`-Funktion aus `math.h` aufnehmen kann?

c) Wie lautet der Prototyp der Funktion `printwtab`, der die genannten Daten als Parameter hat?

d) Schreiben Sie die Definition von `printwtab`!

e) Schreiben Sie das gesamte Programm `printwtab.c`!

Aufgabe 6: Werttabelle II

Die Funktion `printwtab2` bekommt anstelle der Schrittweite einen weiteren Parameter, nämlich die Adresse einer Funktion, die aus dem momentanen Wert der Laufvariablen den nächsten berechnen kann. Eine solche Funktion kann so aussehen:

```
1  double weiterschalten(double aktlauf)
2  {
3     return aktlauf*10.0;
4  }
```

Und so soll `printwtab2` aufgerufen werden können:

```
1    printwtab2(1, 1000, weiterschalten, sin);
```

Dann soll die Wertetabelle so aussehen:

```
  x   |  y
------+------
    1 | 0.841
   10 | 0.544
  100 |-0.506
 1000 | 0.827
```

a) Wie muss eine Variable `wzeiger` vereinbart werden, die die Adresse der Funktion `weiterschalten` aufnehmen kann?

b) Wie lautet der Prototyp der Funktion `printwtab2`, der die genannten Daten als Parameter hat?

c) Schreiben Sie die Definition von `printwtab2` und betten Sie es ein in das Programm `printwtab2.c`!