103 – Strukturiert programmieren nach Vorgabe

```
#include <stdio.h>
#include <math.h>
#define Benzinverbrauch (6.9/100)
#define Benzinpreis 1.38
        void main()
{
        float x,k;
        printf("Geben Sie bitte Ihre gefahrenen
Kilometer ein: \t");
        scanf("%f",&x);
        k=Benzinverbrauch*Benzinpreis*x;
        printf("\nKosten: %42.2f Fr \n",k);
}
```

```
{
Int x=5, y=6, res=0;
Res = x + x*y;
Printf("res = %i n",
res);
}

Lösung: 35
```

Ganzzahlen:

char	c
short	li
Int	i, d
long	ld

Fliesskommazahl

float	f, g
double	lf, lg
long double	

fflush(stdin) Inhalt des Tastenpuffers löschen
getchar()

#include <stdlib.h> system(„cls“);

Konstanten
Symbolische Konstanten #define
(Zeichenkette, die vor dem Übersetzen durch eine andere ersetzt wird)
Stringkonstante: zwischen 2 **„“ printf(„Hallo“n);**
Zeichenkonstante printf(„Zeichenkonstante: %c \n“. ’X’)
Zahlenkonstante printf(„Zahlenkonstante: %d \n“. ’X’
const float const MWST = 15.0;

char z = ’C’;
printf(“Wert: %d (ASCII-codiert) → %c \n“, z,z);

\t = Zeilenvorschub (wie Tab)
\a = Warnsignal
\b = backspace
\n = new line
\“ = Ausgabe von “

ü = \x81
ä = \x84
ö = \x94

%d = Dezimal
%o = oktal
%x = hexadezimal
%c = ASCII

CASE = Computer Assisted Software Engineering

Strukturierte Programmierung

Sequenz: Stellt eine Abfolge von Befehlen dar, die nacheinander ausgeführt werden.

Selektion: Abhängig von einer gesetzten Bedingung wird die zutreffende Anweisung ausgeführt.

- einseitige Selektion (IF)
- zweiseitige Selektion (mit Nein-Zweig) (→ IF-ELSE)
- mehrseitige Selektion (CASE, SWITCH)

Iteration: Befehle werden solange wiederholt, solange bestimmte Bedingungen erfüllt sind.

- kopfgesteuert (abweisend, anfangsgeprüft)
- fussgesteuert (annehmend, endgeprüft)
- Zählschleife; for-Schleife (kopfgesteuert, Anzahl Durchläufe ist bekannt. Schleife wird solange durchlaufen, bis der Endwert erreicht ist.)

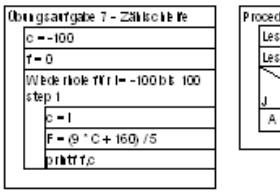
```
#include <stdio.h>
#include <math.h>

void main()
{
        int c= -100;
        float f=0;
        while(c <= 100)
        {
                f=(9*c + 160)/5;
                printf(" %i \tCelsius   =
%.2f \tFahrenheit\n", c,f);
                c++;
        }
}
```

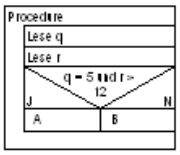
Wahrheitstabelle

q = 5	r > 12	Tätigkeit
0	0	B
1	0	B
0	1	B
1	1	A

AND = &&
OR = |

1 = wahr
0 = falsch






```
if (zahl == 50)
        {
        Anweisungen
        }
else if(zahl > 50)
        {
        Anweisungen
        }
else
{
Anweisungen
}
```

Argumente:
Werte, die beim Aufruf einer Funktion an deren Parameter übergeben werden.

Compiler
Ein Programm, mit dem man den Quellcode eines Programms in Maschinencode übersetzen kann.

Debugger
Programm, das zur Fehlersuche dient. Dabei kann man unter anderen ein Programm Schritt für Schritt ausführen oder Variableninhalte überwachen.

© by Flavio De Roni

Deklaration
Bekanntmachung einer Variablen, Funktion usw im Programmquellcode

In & Dekrement
Den Wert einer Variablen um den Wert 1 erhöhen bzw. verringern

Funktion
Unter einem Namen zusammengefasster Programmcode

Headerdatei
Eine Quelldatei mit der Dateierweiterung .h oder .hpp, in der normalerweise die Deklaration zu den Definitionen einer Implementierungsdatei (*.c oder *.cpp) zusammengefasst werden.

Linker
Programm, das aus dem Maschinencode eines Programms (vom Compiler erzeugt) und den vom Programm verwendeten Bibliotheksfunktionen eine ausführbare Datei erstellt.

Literale Konstante
Eine Konstante, die im Quellcode direkt als Wert geschrieben wird.

main()-Funktion
Sie Hauptfunktion des C-Programms. Dies ist die erste Funktion, die im Programm aufgerufen wird. Ohne eine main()-Funktion ist das Programm nicht ausführbar.

Parameter
Variablen einer Funktion, die innerhalb der Klammern des Funktionkopfes deklariert sind. Diese Variablen werden beim Aufruf der Funktion über Argumente initialisiert.

Typkonvertierung (casting)
Umwandlung eines Datentyps in einen anderen

Präprozessor
Der Präprozessor ist ein Teil des Compilers. der vor dem Übersetzen des Quellcodes in den Maschinencode temporäre Änderungen am Quellcode vornimmt. Das heisst, der Präprozessor fügt ihrem Quellcode etwas hinzu, das mitkompiliert wird, er entfernt aber auch etwas, so dass bestimmte Zeilen nicht mitkompiliert werden.

Initialisieren
einer Variable einen Wert zuweisen

Welche Nachteile ergeben sich, wenn ein zu berechnender Ausdruck direkt in der printf-Anweisung steht?
Er ist nur temporär gespeichert, d. h., es kann nicht auf ihn zurückgegriffen werden.

Bei scanf ist man schutzlos der Bereichsüberschreitung der Benutzer ausgeliefert. BSP. Der Benutzer schreibt hugo statt eine Zahl!

1.1 Funktionen

Eine Funktion ist ein zusammengefasster Programmcode, der unter einem Namen – dem Funktionsnamen abgelegt wird.

Vorteile:

- Wenn man den gleichen Programmcode mehrmals im selben Programm benutzt, lohnt es sich eine Funktion zu schreiben, da man weniger Code schreiben muss.
- einfache Wartbarkeit. Änderungen und Verbesserungen müssen nur in der Funktion durchgeführt werden und nicht im kompletten Programm verteilt.

```
Rückgabetyp Funktionsname (Parameter)
{
  Anweisung(en);
}
```

Rückgabetyp – Hier wird der Datentyp festgelegt, den die Funktion an den Aufrufe zurückgibt. Soll die Funktion keinen wert zurückgeben ➔ void

Funktionsname – keine Funktionsnamen verwenden, die in den Laufzeitbibliotheken definiert sind, um Namenskonflikte zu vermeiden.

Parameter – Werte, die man in einer Funktion als Argumente übergeben kann. Parameter werden durch den Datentyp und den Parameternamen spezifiziert.

Anweisungen – Die Anweisungen der Funktion (Deklaration, Zuweisungen, Shcleifen usw).

Vorwärtsdeklaration einer Funktion (vor der Main-Funktion)
void func_sinnlos();

Rückgabewert – return (Variable)

Unterschied zwischen Parameter und Argument

Als Parameter bezeichnet man die in runden Klammern angegebenen Variablen der Funktionsdefinition. Argumente sind dagegen die Werte, die beim Aufruf der Funktion an diese Parameter übergeben werden können.

Lokale Variablen sind nur innerhalb der Funktion, in der sie definiert wurden, gültig.

Globale Variablen sind über alle Funktionen in einem Programm sichtbar du somit nutz- und änderbar. Sind anfällig auf unbeabsichtigte Änderungen!!!

1.2 Arrays

Mit Arrays kann man mehrere Variablen eines Typs zusammenfassen.

1.2.1

Arrays an Funktionen übergeben:

void function(int arrayname[], int anzahl_elemente)

Rückgabewert = funktionsname (arrayname)

1.3 Strings (char-Array)

Als String bezeichnet man eine Kette von Zeichen.

- Bevor einer Variable einen Wert zugewiesen werden kann, muss diese definiert werden
- Unter initialisieren versteht man, dass der Variablen einen Wert zugewiesen wird.
- Ein sinnvolles Arbeiten mit Variablen erfordert immer eine Initialisierung.

char string[100];

Es muss immer Platz für das Terminierungszeichen „\0“ vorhanden sein!!!

Formatbezeichner für Strings: %s

#include <string.h>

String einlesen: fgets()
fgets(String, Zeichenlänge, Quelle);
Bsp: fgets(name_vorname, 60, stdin);
Strings aneinanderhängen: strcat(ziel_string, quell_string);

scanf(„%s“,&txt);
gets(txt);

String ausgeben: puts()
puts(txt);
printf(„%s“,txt);

Strings kopieren: strcpy()
strcpy(zielstring, quellstring);

Länge eines Strings: strlen()
anzahl_zeichen = strlen(string);

Strings miteinander vergleichen:
strcmp(String1, String2);
(Diese Funktion liefert bei Gleichheit den Wert 0 zurück)

1.4 Zeiger (Pointer)

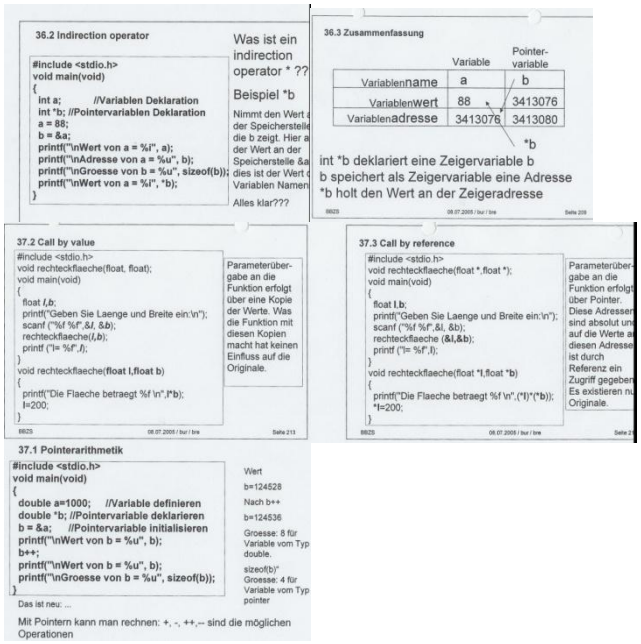
Mit Pointern kann direkt auf die Maschinenspeicheradresse einer Variablen zugegriffen werden!

- Dynamische Reservieren eines Speicherplatzes
- Werte können direkt an Funktonen übergeben (call-by-reference) und auch wieder als Adresse zurückgegeben werden
- Komplexe Datenstrukturen wie Listen und Bäume lassen sich nur mit Zeigern erstellen.

Datentyp *Zeigername

Mittels Adressoperator & wird die **Adresse** einer Variablen nicht ihr Wert angesprochen.

- Speicherplatz wird durch Adressoperator & geliefert
- Variablen haben einen Wert, Namen, Speicheradresse
- Eine Pointervariabel speichert eine Adresse (Referenz)
- Mit dem Indirection Operator (* in einer Programmanweisung) kann auf den Wert der Speicherstelle, wo der Zeiger hinzeigt, zugegriffen werden: *ptr liefert also den Wert an der Adresse ptr.
- Eine der wichtigsten Aufgaben eines Pointers ist es, durch call by reference den Zugriff auf das Original für die Funktion bereitzustellen (Adresse wird übergeben)
- Pointer können durch +1 auf das nächste Element seines Datentyps zeigen (+,-,++,-- sind möglich)
- Der Name eines Feldes (Array) ist ein Pointer auf das erste Element dieses Feldes.




```
(1)   #include <stdio.h>
      #define MAX 50
      int main(void)
      {
(2)     int index;
(3)     int nummer;
(4)     int flag;
(5)     char buchstabe;
(6)     char liste[MAX];
        printf("Bitte geben Sie die Liste der Buchstaben ein: \n");
        for(index = 0; index < MAX; index++)
        {
(7)             scanf("%c", &liste[index]);
(8)             if(liste[index] < 65)
                  index--; /*Entfernen der Steuerzeichen */
        }
        nummer = 0; /* NUMMER DES DURCHLAUFS */
        do
        {
(9)             flag = 0;
                nummer++;
                for(index = 0; index < (MAX - nummer); index++)
                {
                  if(liste[index] > liste[index + 1])
                  {
(10)                      flag = 1;
                          buchstabe = liste[index];
                          liste[index] = liste[index + 1];
liste[index + 1] = buchstabe;
                  }
                }
        }
(11)    while (flag == 1);
        printf("Sortierte Liste: \n");
        for(index = 0; index < MAX; index++)
                printf("%c ", liste[index]);
        printf("Sortierung erfolgte nach %d Durchläufen.", nummer);
        return 0;
      }
```

1. Die Konstante MAX wird per Präcompiler-Anweisung festgelegt. Das ermöglicht später mit einem grösseren Array das Sortieren durchzuführen. Setzen Sie dafür die Konstante MAX auf 50. So können Sie das Programm mit 50 Elementen testen.
2. Die Variable index wird vereinbart. Sie dient zur Festlegung des Elementes innerhalb der Liste.
3. Die Variable nummer wird vereinbart. Sie dient zum Zählen der vollständigen Durchläufe.
4. Diese Variable enthält den Wert 1, wenn in einem Durchgang die benachbarten Elemente getauscht werden mussten, sonst 0. Es dient damit zum Signalisieren, dass alle Elemente richtig alphabetisch geordnet sind.
5. Diese Variable dient beim Tauschen der Elemente als Zwischenspeicherplatz.
6. Vereinbarung eines Arrays mit dem Bezeichner liste mit MAX Elementen vom Typ char. Die Zählung beginnt bei 0 und endet bei MAX-1.
7. Die Buchstaben werden über die Tastatur eingegeben. Dies erfolgt entweder, indem Sie alle Zeichen eingeben und die Eingabe mit der Taste RETURN abschliessen, oder die Eingabe jedes einzelnen Buchstabens mit RETURN abschliessen.
8. In dieser Zeile werden alle überflüssigen Zeichen, z. B. Steuerzeichen, die nicht in das Array geschrieben werde sollen, ignoriert.
9. Vor jedem Durchlauf wird die Variable flag auf den Wert 0 gesetzt. Das Flag wird in der folgenden Schleife in Zeilenmarkierung (10) auf 1 gesetzt, falls ein Tausch nötig war.
10. Da ein Tausch nötig ist, wird das Flag auf 1 gesetzt.
11. In dieser Zeile wird das Flag auf 1 getestet. Der Test ist negativ (0 - false), wenn die Liste von Buchstaben bereits sortiert ist. Ist das Flag 1 ist, ist ein weiterer Durchlauf nötig.

Variablendeklaration:
Erlaubt:meineVar, myVar5, meine_Var
Nicht erlaubt: 5meineVar, meine#Var, meine Var

```
  for(index = 0; index < MAX; index++)
  {
            scanf("%c", &liste[index]);
            if(liste[index] < 65)
              index--; /*Entfernen der Steuerzeichen */
  }
  nummer = 0; /* NUMMER DES DURCHLAUFS */
  do
  {
            flag = 0;
            nummer++;
            for(index = 0; index < (MAX - nummer); index++)
            {
              if(liste[index] > liste[index + 1])
              {
                        flag = 1;
                        buchstabe = liste[index];
                        liste[index] = liste[index            + 1]; liste[index +
1] = buchstabe;
              }
            }
  }
```

2 Theorie-Fragen

Die kopfgesteuerte Iteration ist:

- eine abweisende Schleife
- verhindert gegebenenfalls, dass die Schleife durchlaufen wird.

Die Enwurfsmethode nach Nassi Shneidermann heisst:

- Struktogramm

Welche Kontrollstrukturen werden bei der strukturierten Programmierung eingesetzt?

- Sequenz
- Selektion
- Iteration

Welche Variablennamen sind zulässig?

_5, Int
(nicht zulässig sind Zins%, 3A)