Lösung Übung 4 infT

1.
Studieren Sie in Ihren Unterlagen (Buch, Man-Pages, Internet), wozu **enum** dient und wie es angewendet wird. Schreiben Sie dann ein Programm, das für die Codes der Zeichen *Backspace* (**BS**: 8hex), *Tabulator* (**TAB**: 9hex), *Linefeed* (**LF**: Ahex), *Vertical Tab* (**VT**: Bhex), *Formfeed* (**FF**: Chex) und *Carriage Return* (**CR**: Dhex) eine Aufzählung (**enum**) definiert und diese Zeichen (**char**) dann ausgibt.
L:

```
#include <string.h>

int i;
enum {BS = 0x08, TAB, LF, VT, FF, CR};
char ctrls[] = {BS, TAB, LF, VT, FF, CR, '\0'};

void main(void) {
  for (i = 0; i < strlen(ctrls); ++i) {
    printf("ctrls[%i] = %i\n", i, ctrls[i]);
  }
}
```

2.
Welche Ausgabe liefert das folgende Programm?

```
#include <stdio.h>

struct person {
  char name[21];
  long int knr;
};

int main() {
  static struct person kunde[4] = {
    "Meier", 20123,
    "Mueller", 82765,
    "Zuercher", 98761
  };

  printf("\n%s", kunde[0].name);
  printf("\n%c", kunde[2].name[0]);
  printf("\n%i", strlen(kunde[1].name));
  printf("\n%li", kunde[1].knr);
  printf("\n%li", kunde[3].knr);

  return 0;
}
```

L:

```
Meier
Z
7
82765
0       // static Variablen werden in das Data Segment geschrieben, das
        // bei Programmbeginn mit Nullen initialisiert sein kann...
```

3.
Zeigen Sie an einem Beispiel, dass bei der Zuweisung einer Auto-Variablen vom Typ einer Struktur **Xy** (mit allen Feldern initialisiert) an eine andere Auto-Variablen desselben Typs **Xy** jedes einzelne Strukturfeld (Member) initialisiert wird.

L:

```
#include <stdio.h>

struct Xy {
  char name[31];
  int i;
  float f;
};


int main() {
  struct Xy var1 = {"Isidor Sanchez", 1234, -98.765};
  struct Xy var2;

  printf("var1.name: %s\n", var1.name);
  printf("var1.i:    %i\n", var1.i);
  printf("var1.f:    %f\n", var1.f);

  var2 = var1;

  strcpy(var1.name, "Aeberhard-Moser");
  var1.i = -979;
  var1.f = -999.222;

  printf("var1.name: %s\n", var1.name);
  printf("var1.i:    %i\n", var1.i);
  printf("var1.f:    %f\n", var1.f);

  printf("var2.name: %s\n", var2.name);
  printf("var2.i:    %i\n", var2.i);
  printf("var2.f:    %f\n", var2.f);

  return 0;
}
```

4.

Testen Sie, ob das Felder-weise Kopieren (vgl. Aufgabe **3.**) auch bei Funktionsargumenten und Return-werten funktioniert.

L:

```
#include <stdio.h>

struct Xy {
  char name[31];
  int i;
  float f;
};


void printXy(struct Xy var) {
  printf("var.name: %s\n", var.name);
  printf("var.i:    %i\n", var.i);
  printf("var.f:    %f\n", var.f);
}


struct Xy modifyXyInFct(struct Xy var) {
  strcpy(var.name, "Aeberhard-Moser");
  var.i = -979;
  var.f = -999.222;
  return var;
}
```

```
int main() {
  struct Xy var1 = {"Isidor Sanchez", 1234, -98.765};
  struct Xy var2;

  // Show that access is possible
  printXy(var1);

  // Show that copy is accessed and modified
  var2 = modifyXyInFct(var1);

  printf("var2.name: %s\n", var2.name);
  printf("var2.i:    %i\n", var2.i);
  printf("var2.f:    %f\n", var2.f);

  printf("var1.name: %s\n", var1.name);
  printf("var1.i:    %i\n", var1.i);
  printf("var1.f:    %f\n", var1.f);

  return 0;
}
```

5.
Wie muss eine Funktion **dMin** aussehen, die den kleineren von zwei **double**-Werten zurückliefert?
L:

```
double dMin(double val1, double val2) {
  return val1 < val2 ? val1 : val2;
}
```

6.
Schreiben Sie eine Funktion mit Namen **dAbs**, die den Absolutwert eines **double**-Wertes zurückliefert.
L:

```
double dAbs(double val) {
  return val < 0 ? -val : val;
}
```

7.
Schreiben Sie eine Funktion namens **beep**, die Töne erzeugt. Die Anzahl der Töne wird als Parameter übergeben.
L:

```
void beep(int nofBeeps) {
  int i;

  for (i = 0; i < nofBeeps; ++i) {
    printf("\a");
  }
}
```

8.
Welche Aufgabe hat die folgende Funktion?

```
int umlaut(int zeichen) {
  switch(zeichen) {
    case 'ä':  case 'Ä':  case 'ö':  case 'Ö':  case 'ü':  case 'Ü':
      return 1;
    default:
      return 0;
  }
}
```

L:
umlaut gibt mit **1** (**TRUE**) bekannt, dass **zeichen** ein **ä**, **Ä**, **ö**, **Ö** oder **Ü** ist, bzw. mit **0** (**FALSE**), dass nicht.

9.
Schreiben Sie eine Funktion mit dem Namen **`vorzeichen`**, die den Rückgabetyp **`int`** hat und als Parameter einen Wert vom Typ **`double`** erhält. Wenn der Wert grösser als Null ist, soll **`1`**, wenn er kleiner als Null ist, **`-1`**, und wenn er gleich Null ist, **`0`** zurückgegeben werden.
L:

```
int vorzeichen(double val) {
  if (val > 0) return 1;
  else if (val < 0) return -1;
  else return 0;
}
```

10.
Schreiben Sie eine Funktion, die einen **`double`**-Wert potenziert. Der Prototyp der Funktion sieht so aus: **`double hoch(double wert, int potenz);`**. Wird als Potenz **`0`** angegeben, soll der Wert **`1`**, wird als Potenz ein negativer Wert angegeben, soll der Fehlerwert **`-1`** zurückgeliefert werden. Ansonsten wird nach der Formel *funktionsrückgabe = wert hoch potenz* gerechnet.
L:

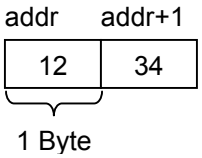
```
#define ERROR -1

double hoch(double wert, int posPotenz) {
  double res;

  if (posPotenz < 0) return ERROR;

  res = 1;
  while (posPotenz--) {
    res *= wert;
  }
  return res;
}
```

11.
Ein Programm soll einen Satz einlesen und die Anzahl der Buchstaben *e* ermitteln. Lassen Sie auch den prozentualen Anteil dieses Buchstabens an der Gesamtanzahl berechnen.
L:

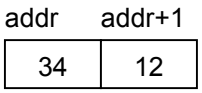
```
#include <stdio.h>
#include <string.h>

#define MAX_LAENGE 80

int main(void) {
  char satz[MAX_LAENGE];
  int i, laenge, anzahl = 0;
  double prozent;

  printf("Geben Sie einen Satz ein (max. %i Zeichen).\n>", MAX_LAENGE);
  scanf("%s", satz);
  laenge = strlen(satz);
  for (i = 0; i < laenge; ++i) {
    if (satz[i] == 'e') {
      anzahl++;
    }
  }
  prozent = 100.0 / laenge * anzahl;
  printf("%.2lf%% der Zeichen sind ein 'e'.", prozent);
}
```

12.
Fakultativ:
Zeigen Sie mit Hilfe einer Union, wie ob Ihr Computer Rechner-intern die *Big Endian* oder *Little Endian*-Darstellung benutzt.

Hinweis:
Interne Darstellung der Ganzzahl 0x1234 nach *Big Endian*-Manier:



Interne Darstellung der Ganzzahl 0x1234 nach *Little Endian*-Manier:



L:

```
/* endian.c */
#include <stdio.h>

#define SIZE 11


union Data {
  char str[SIZE];
  int l;
};


int main() {
  //                  0123456789
  union Data data = {"          "};
  int i;

  data.l = 0x12345678;
  for (i = 0; i < SIZE - 1; i++) {
    printf("%X ", data.str[i]);
  }
  putchar('\n');

  return 0;
}


/*
Ausgabe:
78 56 34 12 20 20 20 20 20 20
*/
```

Hier handelt es sich um eine Maschine mit interner Little-Endian-Darstellung

13.
Wie sind die Warnungen und Fehler zu interpretieren, wenn Sie nachfolgendes Programm kompilieren?

```
 1  /* prog.c */
 2
 3  void hello(int i[]) {
 4    return i;
 5  }
 6
 7  float hello(float f) {
 8    return f;
 9  }
10
11  int main() { }
```

L:
(Meldungen für gcc)

```
prog.c: In function 'hello':
prog.c:4: warning: 'return' with a value, in function returning void
```

return mit Rückgabewert in Funktion und **void** als Rückgabewert widersprechen sich.

```
prog.c: At top level:
prog.c:7: conflicting types for 'hello'
prog.c:3: previous declaration of 'hello'
```

Widersprüchliche Parameterwerte der Funktionen **hello**. Namenskollision, **hello** existiert bereits.