Inhalt

Übung Hello World 2
HelloWorld 2
Übung 1 3
1.1 – for Schleife 3
1.2 – while Schleife 5
1.3 – Integer nach Binary (iToBinary) 6
1.4 – Text Menu 7
1.5 – Celsius in Fahrenheit 10
1.6 – ggT mit while Schlaufe 11
1.7 – Wörter zählen 12
Übung 2 13
2.1 – String Reverse 13
2.2 – Integer to String 15
2.3 – Vektoren, malloc, realloc 17
2.4 – Enum to String 19
2.5 – strcat 21
2.6 – Fibonacci Zahlen 22
Übung 3 23
3.1 – Liste mit Namen/Vornamen 23
3.2 – Union 25
3.3 – Wörter Zählen 28
3.4 – Structure Tree 31
Übung 4 35
4.1 – Taschenrechner polnische Notation: 35
4.2 – Makros 42
4.3 – Ampel 43
4.4 – Message Queue 46
5.1 – Matrix erstellen 49
Übung 5 52
5.2 – Zeigerarithmetik 52
5.3 – Zeiger auf Funktion 56
5.4 – Mehrdimensionales Array mit Druckmesswerten 60
Übung 6 63
6.1 – Projekt Adressverwaltung 63

Übung Hello World

HelloWorld

```c
/*============================================================================
Hochschule Luzern - Technik & Architektur            Mikrocontroller - HS2012
Copyright 2012 Felix Rohrer                              http://hslu.ximit.ch
------------------------------------------------------------------------------
Name        : HelloWorld.c
Author      : Felix Rohrer <felix.rohrer@stud.hslu.ch>
Version     : 2012-09-22v1.0
Description : Hello World Sample!
============================================================================*/

#include <stdio.h>
#include <stdlib.h>

/**
 * The main procedure to run the application.
 * @param argc Number of command line arguments
 * @param argv The forwarded command line arguments
 * @return Application return (error) code
 */
int main(int argc, char** argv)
{
  printf("Hello World!\n");

  getchar();
  return(EXIT_SUCCESS);
}
```

Übung 1

1.1 – for Schleife

```
/**============================================================================
Hochschule Luzern - Technik & Architektur            Mikrocontroller - HS2012
Copyright 2012 Felix Rohrer                              http://hslu.ximit.ch
------------------------------------------------------------------------------
Name        : Uebung_1-1.c
Author      : Felix Rohrer <felix.rohrer@stud.hslu.ch>
Version     : 2012-09-22v1.0
Description : Uebung_1-1
              Entwurf Übung mit Funktionen und for - Schleife
============================================================================*/

#include <stdio.h>
#include <stdlib.h>

/**
* Draw "Raute" based on the Parameters
* Example: drawRaute(3, 5)
* #####
* #####
* #####
* @param zeile Anzahl Zeilen
* @param spalte Anzahl Spalten
*/
void drawRaute(int zeile, int spalte);

/**
* Print out all Numbers from 0..value, ex, printNumbers(4): 0 01, 012, 0123, 01234
* @param value Wert > 0 bis wohin gezählt werden soll
*/
void printNumbers(int value);

/**
* Draw a triangle with *'s
* @param size Grösser des Dreiecks
*/
void drawTriangle(int size);

/**
 * The main procedure to run the application.
 * @param argc Number of command line arguments
 * @param argv The forwarded command line arguments
 * @return Application return (error) code
 */
int main(int argc, char** argv)
{
  printf("drawRaute(3, 5)...\n");
  drawRaute(3, 5);

  printf("drawRaute(2, 2)...\n");
  drawRaute(2, 2);

  printf("printNumbers(10)...\n");
  printNumbers(10);

  printf("drawTriangle(6)...\n");
  drawTriangle(6);


        printf("\n\nPress [Enter] to exit...");
        getchar();
        return(EXIT_SUCCESS);
}



void drawRaute(int zeile, int spalte)
{
        int lineCount, rowCount;
        for (lineCount = 0; lineCount < zeile; lineCount++)
        {
                for (rowCount = 0; rowCount < spalte; rowCount++)
                {
                        printf("#");
                }
                printf("\n");
        }
```

```
        printf("\n");
}

void printNumbers(int value)
{
        int count, currCount;
        for (count = 0; count <= value; count++)
        {
                for (currCount = 0;currCount<count;currCount++)
                {
                        printf("%i", currCount);
                }
                printf("\n");
        }
        printf("\n");
}

void drawTriangle(int size)
{
        int i, j;
        for (i=size; i>=0; i--)
        {
                for (j=i; j>0; j--)
                {
                        printf(" ");
                }
                for(j=i; j<size; j++)
                {
                        printf("*");
                }
                for(j=i; j<=size; j++)
                {
                        printf("*");
                }
        printf("\n");
        }
}
```

1.2 – while Schleife

```
/**============================================================================
Hochschule Luzern - Technik & Architektur            Mikrocontroller - HS2012
Copyright 2012 Felix Rohrer                              http://hslu.ximit.ch
------------------------------------------------------------------------------
Name        : Uebung_1-2.c
Author      : Felix Rohrer <felix.rohrer@stud.hslu.ch>
Version     : 2012-09-22v1.0
Description : Uebung_1-2
              Entwurf Übung mit Funktionen und while Schlaufe
============================================================================*/

#include <stdio.h>
#include <stdlib.h>

/**
* Increment of an Integer between start and end
* @param start Start Value to increment
* @param end Limit Value
*/
void increment(int start, int end);

/**
* Decrement of an Integer between start and end
* @param start Start Value to decrement
* @param end Limit Value
*/
void decrement(int start, int end);


/**
 * The main procedure to run the application.
 * @param argc Number of command line arguments
 * @param argv The forwarded command line arguments
 * @return Application return (error) code
 */
int main(int argc, char** argv)
{
        printf("increment(0, 5)...\n");
        increment(0, 5);

        printf("decrement(10, -3)...\n");
        decrement(10, -3);

        printf("\n\nPress [Enter] to exit...");
        getchar();
        return(EXIT_SUCCESS);
}

void increment(int start, int end)
{
        while(start <= end)
        {
                printf("%d,", start);
                start++;
        }
        printf("\b \n\n"); // "\b  " = Backspace & "Space" => remove last ","
}

void decrement(int start, int end)
{
        while(start >= end)
        {
                printf("%d,", start);
                start--;
        }
        printf("\b \n\n"); // "\b  " = Backspace & "Space" => remove last ","
}
```

1.3 – Integer nach Binary (iToBinary)

```
/**============================================================================
Hochschule Luzern - Technik & Architektur            Mikrocontroller - HS2012
Copyright 2012 Felix Rohrer                              http://hslu.ximit.ch
------------------------------------------------------------------------------
Name        : Uebung_1-3.c
Author      : Felix Rohrer <felix.rohrer@stud.hslu.ch>
Version     : 2012-09-22v1.0
Description : Uebung_1-2
              Integer nach Binary (iToBinary)
============================================================================*/

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

/**
* Integer to Binary String Converter
* @param value Integer to convert
*/
void iToBinary(int value);


/**
 * The main procedure to run the application.
 * @param argc Number of command line arguments
 * @param argv The forwarded command line arguments
 * @return Application return (error) code
 */
int main(int argc, char** argv)
{
        int wert;

        printf("Wert als Integer (auch negativ): ");
    scanf("%d", &wert);
        fflush(stdin); //flushes the input stream and gets rid of '\n'

        iToBinary(wert);

        printf("\n\nPress [Enter] to exit...");
        getchar();
        return(EXIT_SUCCESS);
}

void iToBinary(int value)
{
        int cmp = 1;
        int cnt;
        char res[sizeof(int) * 8 + 1]; //get size of Integer * 8 (bits) + 1
        memset(res, 0, sizeof(res)); // Sets buffers to a specified character

        cnt = sizeof(int) * 8;
        while (cnt != 0)
        {
                cnt--;
                res[cnt] = (value & cmp) ? '1' : '0';
                cmp <<= 1;
        }
        printf("Integer: %d\n", value);
        printf("Binary:  %s\n", res);

}
```

1.4 – Text Menu

```
/**============================================================================
Hochschule Luzern - Technik & Architektur            Mikrocontroller - HS2012
Copyright 2012 Felix Rohrer                              http://hslu.ximit.ch
------------------------------------------------------------------------------
Name        : Uebung_1-4.c
Author      : Felix Rohrer <felix.rohrer@stud.hslu.ch>
Version     : 2012-09-22v1.0
Description : Uebung_1-4
              Schreiben Sie das Gerüst zu einer textbasierten Menusteuerung.
============================================================================*/

#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>


/**
* Show Main Menu
*/
void showMenuMain();

/**
* Show SubMenu A, A-A, A-B, A-C
*/
void showMenuSubA();
void showMenuSubA_A();
void showMenuSubA_B();
void showMenuSubA_C();

/**
* Show SubMenu B
*/
void showMenuSubB();

/**
* Show SubMenu C
*/
void showMenuSubC();

/**
 * The main procedure to run the application.
 * @param argc Number of command line arguments
 * @param argv The forwarded command line arguments
 * @return Application return (error) code
 */
int main(int argc, char** argv)
{

        // Show MainMenu
        showMenuMain();


        //printf("\n\nPress [Enter] to exit...");
        //getchar();
        return(EXIT_SUCCESS);
}

void showMenuMain()
{
        char input;

        do {
                // Show MainMenu
                system("cls");
                printf("MainMenu\n-------------------------------------------------------\n");
                printf("A --> Select menu item A\n");
                printf("B --> Select menu item B\n");
                printf("C --> Select menu item C\n");
                printf("Q --> Exit\n");

                // get input
                printf("Enter selection: ");
                while (!isalpha(input = toupper(getchar())));
                fflush(stdin); //flushes the input stream and gets rid of '\n'

                // Select SubMenu (or Exit)
                switch(input) {
                case 'A':
                        system("cls");
```

```
                        showMenuSubA();
                        break;
                case 'B':
                        system("cls");
                        showMenuSubB();
                        break;
                case 'C':
                        system("cls");
                        showMenuSubC();
                        break;
                case 'Q':
                        break;
                }
        } while (input != 'Q');
}

void showMenuSubA()
{
        char input;

        do {
                // Show SubMenu
                system("cls");
                printf("SubMenu A\n-----------------------------------------------------\n");
                printf("A --> Select submenu item A-A\n");
                printf("B --> Select submenu item A-B\n");
                printf("C --> Select submenu item A-C\n");
                printf("Q --> Back to MainMenu\n");

                // get input
                printf("Enter selection: ");
                while (!isalpha(input = toupper(getchar())));
                fflush(stdin); //flushes the input stream and gets rid of '\n'

                // Select SubMenu (or Exit)
                switch(input) {
                case 'A':
                        showMenuSubA_A();
                        break;
                case 'B':
                        showMenuSubA_B();
                        break;
                case 'C':
                        showMenuSubA_C();
                        break;
                case 'Q':
                        break;
                }
        } while (input != 'Q');
}

void showMenuSubA_A()
{
        char input;

        system("cls");
        printf("SubMenu A-A\n-------------------------------------------------------\n");
        printf("Q --> Back to SubMenu A\n");
        do { // get input
                printf("Enter selection: ");
                while (!isalpha(input = toupper(getchar())));
                fflush(stdin); //flushes the input stream and gets rid of '\n'
        } while (input != 'Q');
}

void showMenuSubA_B()
{
        char input;

        system("cls");
        printf("SubMenu A-B\n-------------------------------------------------------\n");
        printf("Q --> Back to SubMenu A\n");
        do { // get input
                printf("Enter selection: ");
                while (!isalpha(input = toupper(getchar())));
                fflush(stdin); //flushes the input stream and gets rid of '\n'
        } while (input != 'Q');
}

void showMenuSubA_C()
```

```
{
        char input;

        system("cls");
        printf("SubMenu A-C\n-----------------------------------------------------\n");
        printf("Q --> Back to SubMenu A\n");
        do { // get input
                printf("Enter selection: ");
                while (!isalpha(input = toupper(getchar())));
                fflush(stdin); //flushes the input stream and gets rid of '\n'
        } while (input != 'Q');
}


void showMenuSubB()
{
        char input;

        system("cls");
        printf("SubMenu B\n-----------------------------------------------------\n");
        printf("Q --> Back to MainMenu\n");
        do { // get input
                printf("Enter selection: ");
                while (!isalpha(input = toupper(getchar())));
                fflush(stdin); //flushes the input stream and gets rid of '\n'
        } while (input != 'Q');
}

void showMenuSubC()
{
        char input;

        system("cls");
        printf("SubMenu C\n-----------------------------------------------------\n");
        printf("Q --> Back to MainMenu\n");
        do { // get input
                printf("Enter selection: ");
                while (!isalpha(input = toupper(getchar())));
                fflush(stdin); //flushes the input stream and gets rid of '\n'
        } while (input != 'Q');
}
```

1.5 – Celsius in Fahrenheit

```
/**============================================================================
Hochschule Luzern - Technik & Architektur            Mikrocontroller - HS2012
Copyright 2012 Felix Rohrer                              http://hslu.ximit.ch
------------------------------------------------------------------------------
Name        : Uebung_1-5.c
Author      : Felix Rohrer <felix.rohrer@stud.hslu.ch>
Version     : 2012-09-22v1.0
Description : Umrechnung Fahrenheit -> Celsius mit Funktion
============================================================================*/

#include <stdio.h>
#include <stdlib.h>

/**
* Convert Fahrenheit -> Celsius
* @param value Fahrenheit
* @return Value in Celsius
*/
float fahrenheitToCelsius(float fahrenheit);

/**
 * The main procedure to run the application.
 * @param argc Number of command line arguments
 * @param argv The forwarded command line arguments
 * @return Application return (error) code
 */
int main(int argc, char** argv)
{
        float f, c;
        int start = 0;
        int end = 210;
        int step = 15;

        printf("Fahrenh.\tCelsius\n");
        for(f = start; f <= end; f += step) {
                c = fahrenheitToCelsius(f);
                printf("%d\t\t%f\n", (int)f, c);
        }

        printf("\n\nPress [Enter] to exit...");
        getchar();
        return(EXIT_SUCCESS);
}

float fahrenheitToCelsius(float fahrenheit)
{
        return (5.0f / 9.0f) * (fahrenheit - 32.0f);
}
```

1.6 – ggT mit while Schlaufe

```
/**============================================================================
Hochschule Luzern - Technik & Architektur            Mikrocontroller - HS2012
Copyright 2012 Felix Rohrer                              http://hslu.ximit.ch
------------------------------------------------------------------------------
Name        : Uebung_1-6.c
Author      : Felix Rohrer <felix.rohrer@stud.hslu.ch>
Version     : 2012-09-23v1.0
Description : Grösster gemeinsamer Teiler (ggT) mit while Schlaufe
============================================================================*/

#include <stdio.h>
#include <stdlib.h>

/**
* Berechnet den GGT von zwei Zahlen
* @param a erste Zahl
* @param b zweite Zahl
*/
int ggT(int a, int b);

/**
 * The main procedure to run the application.
 * @param argc Number of command line arguments
 * @param argv The forwarded command line arguments
 * @return Application return (error) code
 */
int main(int argc, char** argv)
{
        int ersteZahl, zweiteZahl, ggtZahl;

        printf("Erste Zahl: ");
        scanf("%d", &ersteZahl);
        printf("Zweite Zahl: ");
        scanf("%d", &zweiteZahl);

        ggtZahl = ggT(ersteZahl, zweiteZahl);
        printf("GGT von %d und %d ist: %d\n", ersteZahl, zweiteZahl, ggtZahl);

        fflush(stdin); //flushes the input stream and gets rid of '\n'
        printf("\n\nPress [Enter] to exit...");
        getchar();
        return(EXIT_SUCCESS);
}

int ggT(int a, int b)
{
        while (a != b) {
                if (a > b) {
                        a -= b;
                }
                else {
                        b -= a;
                }
        }
        return a;
}
```

1.7 – Wörter zählen

```
/**============================================================================
Hochschule Luzern - Technik & Architektur            Mikrocontroller - HS2012
Copyright 2012 Felix Rohrer                              http://hslu.ximit.ch
------------------------------------------------------------------------------
Name        : Uebung_1-7.c
Author      : Felix Rohrer <felix.rohrer@stud.hslu.ch>
Version     : 2012-09-23v1.0
Description : Wörter Zählen
              Trennzeichen: ' ' (space), '\t' (tab) und '\n' (enter)
============================================================================*/

#include <stdio.h>
#include <stdlib.h>

#define IN 1
#define OUT 2

/**
 * The main procedure to run the application.
 * @param argc Number of command line arguments
 * @param argv The forwarded command line arguments
 * @return Application return (error) code
 */
int main(int argc, char** argv)
{
        int c; // current Char
        int counter = 0; // Word Counter
        int state = OUT; // In or Out

        while ((c = getchar()) != EOF)
        {
                if ((c == ' ') || (c == '\t') || (c == '\n'))
                {
                        state = OUT;
                }
                else if (state == OUT)
                {
                        state = IN;
                        counter++;
                }
        }
        printf("Anzahl Woerter: %d\n", counter);


        fflush(stdin); //flushes the input stream and gets rid of '\n'
        printf("\n\nPress [Enter] to exit...");
        getchar();
        return(EXIT_SUCCESS);
}
```

Übung 2

2.1 – String Reverse

```
/**============================================================================
Hochschule Luzern - Technik & Architektur            Mikrocontroller - HS2012
Copyright 2012 Felix Rohrer                              http://hslu.ximit.ch
------------------------------------------------------------------------------
Name        : Uebung_2-1.c
Author      : Felix Rohrer <felix.rohrer@stud.hslu.ch>
Version     : 2012-09-23v1.0
Description : Reverse
              Schreiben Sie eine Funktion reverse(s),
              die die Zeichenkette s zeilenweise umkehrt.
============================================================================*/

#include <stdio.h>
#include <stdlib.h>



/**
* Liest eine Zeile von maximal limit Zeichen ein.
* Die Zeichen werden (inklusive Zeilenende-Zeichen) im übergebenen * Vektor s /0 terminiert abgelegt.
* @param s Zeiger auf den Vektor zum Speichern der Eingabe
* @param limit Maximale Grösse des Vektors
* @return Anzahl eingelesene Zeichen
*/
int readLine(char s[], int limit);

/**
* Kehrt den Input Stream um speichert ihn in out.
* @param in String Input
* @param out Input String Reverse
*/
void reverseString(char in[], char out[], int len);


/**
 * The main procedure to run the application.
 * @param argc Number of command line arguments
 * @param argv The forwarded command line arguments
 * @return Application return (error) code
 */
int main(int argc, char** argv)
{

        char stringInput[100], stringReverse[100];
        int readCount;

        // Read Input String
        readCount = readLine(stringInput, 100);
        printf("Anz. Zeichen: %d\nInput:  %s", readCount, stringInput);
        //Reverse String
        reverseString(stringInput, stringReverse, readCount-1);
        //Output
        printf("Output: %s\n", stringReverse);







        fflush(stdin); //flushes the input stream and gets rid of '\n'
        printf("\n\nPress [Enter] to exit...");
        getchar();
        return(EXIT_SUCCESS);
}


int readLine(char s[], int limit)
{
        int i = 0;
        int c;
        c = getchar();                  /* Buchstabe einlesen */
        while((c != EOF) &&             /* Ende File ... */
          (c != '\n') &&/* oder Ende Zeile ... */
                  (i < limit - 1))      /* oder Limite des Speichers? */
        {
```

```
                s[i] = c;
                i++;
                c = getchar();
        }
        if (c == '\n')
        {
                s[i] = '\n';            /* Zeilenumbruch anfügen! */
                i++;
        }
        s[i] = '\0';                    /* Zeichenkette-Ende anfügen! */
        return i;                               /* Anzahl gelesene Zeichen zurückgeben */
}

void reverseString(char in[], char out[], int len)
{
        int i = 0;
        for (; i<len; i++)
        {
                out[i] = in[len-i-1];
        }
        out[i] = '\0';
}
```

2.2 – Integer to String

```
/**============================================================================
Hochschule Luzern - Technik & Architektur            Mikrocontroller - HS2012
Copyright 2012 Felix Rohrer                              http://hslu.ximit.ch
------------------------------------------------------------------------------
Name        : Uebung_2-2.c
Author      : Felix Rohrer <felix.rohrer@stud.hslu.ch>
Version     : 2012-09-23v1.0
Description : Integer to String
              Schreiben Sie eine Funktion char* itoa(int i), welche die
              übergebene Integerzahl in dezimaler Darstellung als
              Zeichenkette zurück gibt.
============================================================================*/

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>

/**
 * Berechnet die Anzahl stellen einer Zahl
 * @param number Zahl von der die anzahl Stellen gesucht sind
 * @return Anzahl Stellen
 */
int getDigitCount(int number);

/**
 * Gibt die Zahl als String zurück
 * @param i Zahl als Integer
 * @return Zahl als String
 */
char* itoa(int i);

/**
 * Kehrt den Input Stream um speichert ihn in out.
 * @param in String Input
 * @return String Reverse
 */
char* reverseString(char in[]);


/**
 * The main procedure to run the application.
 * @param argc Number of command line arguments
 * @param argv The forwarded command line arguments
 * @return Application return (error) code
 */
int main(int argc, char** argv)
{
        int wert;
        char *result;

        // Wert einlesen
        printf("Wert als Integer (auch negativ): ");
        scanf("%d", &wert);

        // Integer to ASCII
        result = itoa(wert);

        printf("Result: %s\n", result);

        fflush(stdin); //flushes the input stream and gets rid of '\n'
        printf("\n\nPress [Enter] to exit...");
        getchar();
        return(EXIT_SUCCESS);
}

int getDigitCount(int number) {
    if(number != 0) {
        return (int)log10(abs(number)) + 1;
    } else {
        return 1;
    }
}

char* itoa(int i)
{
        int countDigit = 0;
        int pos = 0;
        int isPositive = 1;
```

```
        char *res;

        // Anzahl Stellen ermitteln
        countDigit = getDigitCount(i);
        // printf("GetDigitCount(%d): %d\n", i, countDigit);
        if (i < 0)
        {
                isPositive = 0;
                countDigit++;
                i = abs(i);
        }

        // Speicher für String reservieren
        res = malloc(sizeof(char) * countDigit + 1);
        if (res)
        {
                while(i != 0)
                {
                        res[pos] = (char) (i % 10 + 48); // + 48 für ASCII Wert
                        i /= 10;
                        pos++;
                }
                if (!isPositive)
                {
                        res[pos] = '-';
                        pos++;
                }
                res[pos] = '\0';
                res = reverseString(res);
        }
        else
        {
                printf("Error bei Speicherallokation...\n");
        }

        return res;
}

char* reverseString(char in[])
{
        int len = strlen(in);
        int i = 0;
        char tmp;
        for (; i<(len/2); i++)
        {
                tmp = in[i];
                in[i] = in[len-i-1];
                in[len-i-1] = tmp;
        }
        return in;
}
```

2.3 – Vektoren, malloc, realloc

```
/**============================================================================
Hochschule Luzern - Technik & Architektur            Mikrocontroller - HS2012
Copyright 2012 Felix Rohrer, Mike Estermann
------------------------------------------------------------------------------
Name        : Uebung_2-3.c
Author      : Felix Rohrer <felix.rohrer@stud.hslu.ch>
              Mike Estermann <mike.estermann@stud.hslu.ch>
Version     : 2012-09-23v1.0
Description : Entwurf Übung mit Vektoren und malloc / realloc
============================================================================*/

#include <stdio.h>
#include <stdlib.h>
#include <windows.h>
#include <time.h>

/**
 * Memory Allozieren
 * @param size Anzahl MB die alliziert werden sollen
 * @return int-Vector
 */
int* allocate(long size);

/**
 * Sensless-Methode um den Vector zu benutzen.
 * Random Nummern speichern und wieder lesen.
 * @param vector Pointer auf den Vector
 */
void useVector(int *vector);


/**
 * The main procedure to run the application.
 * @param argc Number of command line arguments
 * @param argv The forwarded command line arguments
 * @return Application return (error) code
 */
int main(int argc, char** argv)
{
        // Variablen definieren
        int wert;
        long size;
        int *vector;

        // get size
        printf("Wie viel MB soll alloziert werden: ");
        scanf("%d", &wert);
        size = wert * 1024 * 1024; // Eingaben in MB umrechnen
        fflush(stdin); //flushes the input stream and gets rid of '\n'

        // allocate vector
        vector = allocate(size);

        // use vector
        printf("\n[Enter] druecken um den Speicher zu benutzen...");
        getchar();
        useVector(vector);

        // free memory
        printf("\n[Enter] druecken um den Speicher freizugeben...");
        getchar();
        free(vector);

        // exit
        printf("\n\nPress [Enter] to exit...");
        getchar();
        return(EXIT_SUCCESS);
}


/**
 * Memory Allozieren
 * @param size Anzahl MB die alliziert werden sollen
 * @return int-Vector
 */
int* allocate(long size)
{
        // Variablen definieren
        const int steps = 20;
```

```
        int index = 0;
        int *vector;

        // calc size based on integer-size
        size = size / sizeof(int);

        // Allocate the memory
        for (; index < steps; index++)
        {
                // Allocate or reallocate memory
                vector = (int *)((index == 0) ? malloc(sizeof(int) * size / steps) : realloc(vector,
(sizeof(int) * size / steps) * index));

                if (vector == NULL)
                {
                        // Error
                        perror("\nNicht genug Speicher vorhanden!\n\n");
                        // Exit
                        exit(EXIT_FAILURE);
                }

                // delay for demo
                Sleep(500);
                printf("Memory allocated: %d\n", ((sizeof(int) * size * (index + 1)) / steps));
        }

        // return pointer of the new vector
        return vector;
}

/**
 * Sensless-Methode um den Vector zu benutzen.
 * Random Nummern speichern und wieder lesen.
 * @param vector Pointer auf den Vector
 */
void useVector(int *vector)
{
        // Variablen definieren
        int index;
        int countNumbers = 10;

        // Initialize random generator
        srand(time(NULL));

        // generate random numbers
        for (index = 0; index < countNumbers; index++)
        {
                *(vector + index) = rand();
        }

        // print numbers
        for(index = 0; index < countNumbers; index++)
        {
                printf("%i. %i\n", index + 1, *(vector + index));
        }
}
```

2.4 – Enum to String

```
/**============================================================================
Hochschule Luzern - Technik & Architektur            Mikrocontroller - HS2012
Copyright 2012 Felix Rohrer                              http://hslu.ximit.ch
------------------------------------------------------------------------------
Name        : Uebung_2-4.c
Author      : Felix Rohrer <felix.rohrer@stud.hslu.ch>
Version     : 2012-09-23v1.0
Description : Enum to String
============================================================================*/

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

typedef enum Color_
{
        BLACK,  // Wert 0
        RED,    // Wert 1
        ORANGE, // Wert 2, usw
        YELLOW,
        GREEN,
        BLUE,
        VIOLET
} Color_t;

/**
 * Gibt den Namen einer Farbe zurück
 * @param color ENUM der Farbe
 * @return String mit dem Namen der Farbe
 */
char* getColorName(Color_t color);

/**
 * The main procedure to run the application.
 * @param argc Number of command line arguments
 * @param argv The forwarded command line arguments
 * @return Application return (error) code
 */
int main(int argc, char** argv)
{
        // use return value
        char *colorName;
        colorName = getColorName(RED);
        printf("Color: %s\n", colorName);

        // pass direct to printf()
        printf("Color: %s\n", getColorName(YELLOW));

        // use Number insted of ENUM
        printf("Color: %s\n", getColorName(0));

        // unknow Color
        printf("Color: %s\n", getColorName(99));



        fflush(stdin); //flushes the input stream and gets rid of '\n'
        printf("\n\nPress [Enter] to exit...");
        getchar();
        return(EXIT_SUCCESS);
}


/**
 * Gibt den Namen einer Farbe zurück
 * @param color ENUM der Farbe
 * @return String mit dem Namen der Farbe
 */
char* getColorName(Color_t color)
{
        const int COLORNAME_MAX_LEN = 10; // Max. 9 Zeichen für den Namen + \0 !
        char* colorName = malloc(sizeof(char) * COLORNAME_MAX_LEN);
        if (colorName) // Memory allocation successfully
        {
                switch (color) {
                case BLACK:
                        strcpy(colorName, "Black");
                        break;
                case RED:
```

```
                        strcpy(colorName, "Red");
                        break;
                case ORANGE:
                        strcpy(colorName, "Orange");
                        break;
                case YELLOW:
                        strcpy(colorName, "Yellow");
                        break;
                case GREEN:
                        strcpy(colorName, "Green");
                        break;
                case BLUE:
                        strcpy(colorName, "Blue");
                        break;
                case VIOLET:
                        strcpy(colorName, "Violet");
                        break;
                default:
                        strcpy(colorName, "UNKNOW");
                        break;
                }
        }
        else
        {
                // Memory allocation failed!
                perror("Error allocate Memory!");
                exit(EXIT_FAILURE);
        }
        return colorName;
}
```

2.5 – strcat

```
/**============================================================================================
Hochschule Luzern - Technik & Architektur                                  Mikrocontroller - HS2012
Copyright 2012 Felix Rohrer                                                   http://hslu.ximit.ch
--------------------------------------------------------------------------------------------
Name        : Uebung_2-5.c
Author      : Felix Rohrer <felix.rohrer@stud.hslu.ch>
Version     : 2012-09-23v1.0
Description : strcat
              Schreiben Sie die Funktion mystrcat(s, t) (Sting concatenate) mit Zeiger, welche
              eine Zeichenkette t an das Ende der Zeichenkette s kopiert.
              Zur Erinnerung: Das '/0' Zeichen markiert das Ende einer Zeichenkette.
============================================================================================*/

#include <stdio.h>
#include <stdlib.h>

/**
 * Kopiert String t ans Ende von String s.
 * Vorsicht: String s muss genügend gross sein!
 */
void mystrcat(char *s, const char *t);

/**
 * The main procedure to run the application.
 * @param argc Number of command line arguments
 * @param argv The forwarded command line arguments
 * @return Application return (error) code
 */
int main(int argc, char** argv)
{
        char input1[256];
        char input2[128];

        printf("Erste Zeichenkette: ");
        scanf("%s", input1);

        printf("\nZweite Zeichenkette: ");
        scanf("%s", input2);

        mystrcat(input1, input2);
        printf("\nErgebnis von strcat: %s\n", input1);


        fflush(stdin); //flushes the input stream and gets rid of '\n'
        printf("\n\nPress [Enter] to exit...");
        getchar();
        return(EXIT_SUCCESS);
}

/**
 * Kopiert String t ans Ende von String s.
 * Vorsicht: String s muss genügend gross sein!
 */
void mystrcat(char *s, const char *t)
{
        while (*s) //solange kein \0 gefunden
                s++;   // bis ans ende von s gehen
        while (*s++ = *t++) // t an s anhängen
                ; // work done in der while-bedingung
}
```

2.6 – Fibonacci Zahlen

```
/**============================================================================================
Hochschule Luzern - Technik & Architektur                                  Mikrocontroller - HS2012
Copyright 2012 Felix Rohrer                                                   http://hslu.ximit.ch
--------------------------------------------------------------------------------------------------
Name        : Uebung_2-6.c
Author      : Felix Rohrer <felix.rohrer@stud.hslu.ch>
Version     : 2012-09-23v1.0
Description : Fibonacci Zahlen
============================================================================================*/


#include <stdio.h>
#include <stdlib.h>

/**
 * Berechnet die Fibonacci Zahlen.
 * Musterlösung von HSLU, Peter Sollberger
 * @author Peter Sollberger
 */
int main(int argc, char** argv)
{
	int number, i, count;
	int* fibonaccis;

	// Wert einlesen
	printf("Wieviele Fibonacci-Zahlen wollen Sie berechnen? ");
	scanf("%d", &number);
	printf("\n");

	count = number + 1;
	fibonaccis = malloc(sizeof (int) * count);
	if (fibonaccis)
	{
		// Calc Fibonacci-Zahlen
		fibonaccis[0] = 0;
		fibonaccis[1] = 1;
		for (i = 2; i < count; i++)
		{
			fibonaccis[i] = fibonaccis[i-1] + fibonaccis[i-2];
		}

		// Print
		printf("Die Fibonacci Zahlen von 0 bis %d lauten:\n", number);
		for (i = 0; i < count; i++)
		{
			printf("%d\n", fibonaccis[i]);
		}

		// Free Memory
		free(fibonaccis);
	}
	else
	{
		printf("Fehler in Speicherallokation\n");
	}

	fflush(stdin); //flushes the input stream and gets rid of '\n'
	printf("\n\nPress [Enter] to exit...");
	getchar();
	return(EXIT_SUCCESS);
}
```

Übung 3

3.1 – Liste mit Namen/Vornamen

Schreiben Sie ein Programm, welches in einer einfach verketteten Liste eine beliebige Anzahl von Namen/Vornamen festhält. Allozieren Sie den Speicherplatz für Ihre Daten dynamisch. Schreiben Sie ein Testprogramm, wo Sie diese Liste einsetzen, z.B. indem Sie über die Konsole Namen/Vornamen eingeben und anschliessend die Liste ausgeben.

```
/**============================================================================================
Hochschule Luzern - Technik & Architektur                                  Mikrocontroller - HS2012
Copyright 2012 Felix Rohrer                                                   http://hslu.ximit.ch
--------------------------------------------------------------------------------------------------
Name        : Uebung_3-1.c
Author      : Felix Rohrer <felix.rohrer@stud.hslu.ch>
Version     : 2012-09-30v1.0
Description : Liste mit Namen/Vornamen
              Schreiben Sie ein Programm, welches in einer einfach verketteten Liste eine
              beliebige Anzahl von Namen/Vornamen festhält.
              Allozieren Sie den Speicherplatz für Ihre Daten dynamisch.
============================================================================================*/

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

#define MAXLEN 100

typedef struct name* namePtr_t;
typedef struct name {
        char* lastname;
        char* firstname;
        namePtr_t next;
} name_t;

/**
 * Add a new Name to the List
 * @param parent Parent Object
 * @param firstname Firstname
 * @param lastname Lastname
 * @return Pointer to the current Name
 */
namePtr_t addName(namePtr_t parent, char* firstname, char* lastname)
{
        if (parent == NULL) {
                // add new
                parent = (namePtr_t) malloc(sizeof(name_t));     // alloc memory for this new object
                parent->firstname = strdup(firstname);           // copy firstname (alloc mem & copy string)
                parent->lastname = strdup(lastname);             // copy lastname (alloc mem & copy string)
                parent->next = NULL;
        }
        else
        {
                parent->next = addName(parent->next, firstname, lastname); // rekursiv add, save pointer to
next object
        }
        // return parent (root) object
        return parent;
}

/**
 * Print all names
 * @param n Root Object of the Name List
 */
void printName(namePtr_t n)
{
        if (n != NULL)
        {
                printf("FirstName: %s\nLastName:  %s\n\n", n->firstname, n->lastname);
                printName(n->next);
        }
}
```

```c
/**
 * The main procedure to run the application.
 * @param argc Number of command line arguments
 * @param argv The forwarded command line arguments
 * @return Application return (error) code
 */
int main(int argc, char** argv)
{
	namePtr_t root = NULL;

	char firstname[MAXLEN], lastname[MAXLEN];

	printf("'.' as Firstname will abort the input...\n");
	do
	{
		printf("Firstname: ");
		scanf("%s", firstname);
		fflush(stdin); //flushes the input stream and gets rid of '\n'
		if (firstname[0] != '.') {
			printf("Lastname:  ");
			scanf("%s", lastname);
			fflush(stdin); //flushes the input stream and gets rid of '\n'
			//add new name
			root = addName(root, firstname, lastname);
		}
	} while (firstname[0] != '.');


	// PrintOut
	printf("\n\nEntered Names:\n");
	printName(root);



	fflush(stdin); //flushes the input stream and gets rid of '\n'
	printf("\n\nPress [Enter] to exit...");
	getchar();
	return(EXIT_SUCCESS);
}
```

3.2 – Union

Entwerfen Sie eine Übung analog der Aufgabe in Kapitel 3 wo Sie Unions einsetzen. Schreiben Sie eine Aufgabenstellung und erarbeiten Sie die Musterlösung.

```
/**============================================================================================
Hochschule Luzern - Technik & Architektur                                  Mikrocontroller - HS2012
Copyright 2012 Felix Rohrer                                                   http://hslu.ximit.ch
--------------------------------------------------------------------------------------------
Name        : Uebung_3-2.c
Author      : Felix Rohrer <felix.rohrer@stud.hslu.ch>
Version     : 2012-09-30v1.0
Description : Entwurf einer Übung zum Thema Union
============================================================================================*/

#include <stdio.h>
#include <stdlib.h>

union uZahl_union {
        int iZahl;
        float fZahl;
};


/**
* The main procedure to run the application.
* @param argc Number of command line arguments
* @param argv The forwarded command line arguments
* @return Application return (error) code
*/
int main(int argc, char** argv)
{
        union uZahl_union uZahl;
        union uZahl_union* uZahlPtr;
        uZahlPtr = &uZahl;

        printf("Set int-union = 5 ...\n");
        uZahlPtr->iZahl = 5;
        printf("Union int:   %d\n", uZahlPtr->iZahl);
        printf("Union float: %f\n\n", uZahlPtr->fZahl);

        printf("Set float-union = 3.1415 ...\n");
        uZahlPtr->fZahl = 3.1415f;
        printf("Union int:   %d\n", uZahlPtr->iZahl);
        printf("Union float: %f\n\n", uZahlPtr->fZahl);



        printf("\n\nPress [Enter] to exit...");
        getchar();
        return(EXIT_SUCCESS);
}
```

```c
/**
 * Einsatz von Unions
 * Author: Martin Vogel, Dozent HSLU Luzern
 */
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

/**
 * union, welche Zahl als Float oder 4 Byte aufnimmt
 */
union uZahl_union
{
	long int liZahl;

	struct
	{
		unsigned char Byte3;
		unsigned char Byte2;
		unsigned char Byte1;
		unsigned char Byte0;
	}sByte;

	struct
	{
		unsigned int iByte3 : 8;
		unsigned int iByte2 : 8;
		unsigned int iByte1 : 8;
		unsigned int iByte0 : 8;
	}iByte;

	float fZahl;
};

/**
 * uebergebener String umkehren
 */
void reverse(char *str)
{
	int i = 0, j = strlen(str) - 1;
	char c;
	while (i < j)
	{
		c = str[i];
		str[i] = str[j];
		str[j] = c;
		i++;
		j--;
	}
}

/**
 * Gibt einen long int in binäre Zeichenkette von 1 und 0 aus
 */
void bin_print(long int wert)
{
	unsigned int z;
	char s[100];
	int i;
	z = (unsigned int)wert;
	i = 0;
	while (z)
	{
		s[i] = (z & 0x01) ? '1' : '0';
		z >>= 1;
		i++;
	}
	s[i] = '\0'; // String Ende
	reverse(s); // String umkehren
	printf("\nDie Zahl in binaer: %s\n", s);
}
```

```
/**
 * Umwandlung von 4 Byte nach Float oder umgekehrt. Funktionsweise mittels UNION
 */
int main(int argc, char** argv)
{
        char c;
        union uZahl_union uZahl; // Definition union
        do
        {
                printf("\n");
                printf("B --> Eingabe als 4 Bytes\n");
                printf("F --> Eingabe als Float-Wert\n");
                printf("Q --> Quit\n");
                printf("\n");

                // Solange einlesen bis Zahl oder Buchstabe
                while (!isalnum(c = getchar()));
                c = toupper(c);

                switch (c)
                {
                        case 'B':
                                printf("Gegen Sie die 4 Bytes in Hex ein:");
                                scanf("%x", &uZahl.liZahl);
                                printf("\nDie Zahl in Hex: %X %X %X %X\n", uZahl.iByte.iByte0,
uZahl.iByte.iByte1, uZahl.iByte.iByte2, uZahl.iByte.iByte3);
                                printf("\nDie Zahl in Float: %e\n", uZahl.fZahl);
                                bin_print(uZahl.liZahl);
                                break;
                        case 'F':
                                printf("Geben Sie die Zahl als Float-Wert ein: ");
                                scanf("%f", &(uZahl.fZahl));
                                printf("\nDie Zahl in Float: %e\n", uZahl.fZahl);
                                printf("\nDie Zahl in Hex: %X %X %X %X\n", uZahl.iByte.iByte0,
uZahl.iByte.iByte1, uZahl.iByte.iByte2, uZahl.iByte.iByte3);
                                bin_print(uZahl.liZahl);
                                break;
                        default:
                                break;
                }
        } while (c != 'Q');
        return (EXIT_SUCCESS);
}
```

3.3 – Wörter Zählen

Diese Übungen hier lehnen an die Übungen im Leitprogramm (Binärbäume) aus Programmieren 2 an. Entsprechend finden Sie dort zusätzliche hilfreiche Informationen dazu.

Schreiben Sie ein Programm, welches die Häufigkeit von Wörtern in einem Textfile zählt. Verwenden Sie dabei die Datenstruktur eines Binärbaums. Der Knoten soll dabei einen Zeiger auf ein Wort und die Häufigkeit des Wortes festhalten. Sehen Sie dazu auch das Beispiel im Buch K&R Seite 134 ff (im Kapitel 4 hier beigefügt) Es soll eine Funktion programmiert werden, welche ein File ausliest und einen Binärbaum entsprechend aufbaut. Verwenden Sie dazu die im Anhang vorgegebenen Funktionen getword, getch, ungetch.

Programmieren Sie eine Funktion, welche in postorder vom Baum die Schlüssel Wort und Häufigkeit ausgibt z.B. mittels printf() in der Funktion treeprint.

```
/**============================================================================================
Hochschule Luzern - Technik & Architektur                              Mikrocontroller - HS2012
Copyright 2012 Felix Rohrer                                               http://hslu.ximit.ch
--------------------------------------------------------------------------------------------
Name        : Uebung_3-3.c
Author      : Felix Rohrer <felix.rohrer@stud.hslu.ch>
Version     : 2012-09-23v1.0
Description : Wörter Zählen
              a) Binärbaum erzeugen
              b) Ausgabe in PostOrder & InOrder
============================================================================================*/


#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>

#define MAXWORD 100
#define BUFSIZE 100
#define TXTFILE "text.txt"

struct tnode{
        char *word;
        int count;
        struct tnode *left;
        struct tnode *right;
};

/* siehe auch Buch K&R S. 77 */
char buf[BUFSIZE];      // Buffer fuer ungetch()
int bufp = 0;           // naechste freie Position für buf


/* function def */
int getch(void);
void ungetch(int c);
int getword(char *word, int lim);
struct tnode* talloc(void);
struct tnode* addtree(struct tnode *, char *);
void treeprintinorder(struct tnode* p);
void treeprintpostorder(struct tnode* p);
```

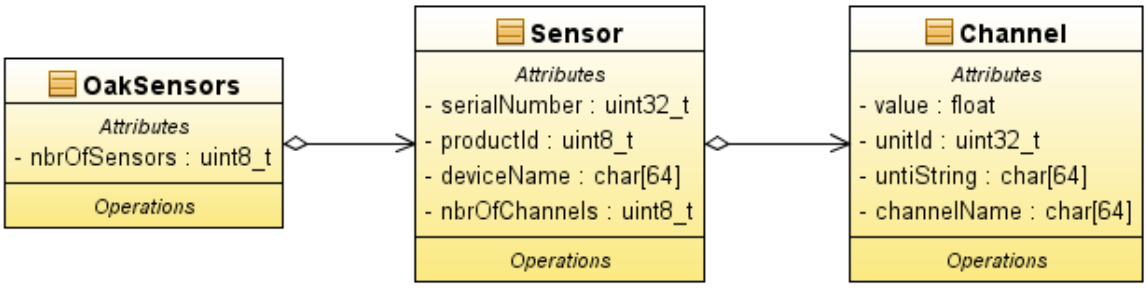
```c
/**
 * The main procedure to run the application.
 * @param argc Number of command line arguments
 * @param argv The forwarded command line arguments
 * @return Application return (error) code
 */
int main(int argc, char** argv)
{
	struct tnode *root;
	char word[MAXWORD];
	root = NULL;

	printf("File '%s' wird eingelesen...\n", TXTFILE);
	if (freopen(TXTFILE, "r", stdin) == NULL) {
		printf("kann File nicht öffnen\n");
	}

	while(getword(word, MAXWORD) != EOF) {
		if(isalpha(word[0])) {
			root = addtree(root, word);
		}
	}

	printf("\nInOrder...\n");
	treeprintinorder(root);

	printf("\nPostOrder...\n");
	treeprintpostorder(root);


	return(EXIT_SUCCESS);
}



int getch(void){ /* naechstes Zeichen holen */
return (bufp > 0 ? buf[--bufp] : getchar());
}


void ungetch(int c){
	if (bufp >= BUFSIZE)
		printf("ungetch: too many characters\n");
	else
		buf[bufp++] = c;
}

/* siehe auch Buch K&R S. 131 */
int getword(char *word, int lim){
	int c;
	char *w = word;
	while(isspace(c = getch()))
		;
	if (c != EOF)
		*w++ = c;
	if (!isalpha(c)) {
		*w = '\0';
		return(c);
	}

	for( ; --lim > 0; w++)
		if(!isalnum(*w = getch())){
			ungetch(*w);
			break;
		}
	*w ='\0';

	return word[0];
}
```

```c
/** Gibt Zeiger auf Speicher zurück. */
struct tnode* talloc(void)
{
	return ((struct tnode*) malloc(sizeof(struct tnode)));
}


struct tnode* addtree(struct tnode* p, char* w)
{
	int cond;
	if (p == NULL) {
		p = talloc(); // einen neuen Knoten erzeugen
		p->word = strdup(w); // Wort kopieren (strdup)
		p->count = 1;
		p->left = NULL;
		p->right = NULL;
	}
	else if ((cond = strcmp(w, p->word)) == 0)
		p->count++; // Wort schon vorgekommen
	else if (cond < 0) // kleiner, links darunter
		p->left = addtree(p->left, w);
	else // grösser, rechts darunter
		p->right = addtree(p->right,w);

	return p; // gibt einen Zeiger zurück, // wo zuletzt eingefügt
}



void treeprintinorder(struct tnode* p)
{
	if (p != NULL){
		treeprintinorder(p->left);
		printf("%4d %s\n", p->count, p->word);
		treeprintinorder(p->right);
	}
}


void treeprintpostorder(struct tnode* p)
{
	if (p != NULL){
		printf("%4d %s\n", p->count, p->word);
		treeprintpostorder(p->left);
		treeprintpostorder(p->right);
	}
}
```

3.4 – Structure Tree

Erstellen Sie eine baumartige, dynamische Datenstruktur zum Verwalten von Sensoren und dessen Messwerte. Dynamisch bedeutet hier, dass die Anzahl Sensoren und Anzahl Kanäle pro Sensor erst zur Laufzeit bekannt sind.

Die Sensoren1 und deren Daten sind in folgendem UML Diagramm ersichtlich.




```
/**============================================================================================================
Hochschule Luzern - Technik & Architektur                                    Mikrocontroller - HS2012
Copyright 2012 Felix Rohrer                                                     http://hslu.ximit.ch
------------------------------------------------------------------------------------------------------------
Name        : Uebung_3-4.c
Author      : Felix Rohrer <felix.rohrer@stud.hslu.ch>
Version     : 2012-09-30v1.0
Description : Structure Tree
              Erstellen Sie eine baumartige, dynamische Datenstruktur zum Verwalten von Sensoren
              und dessen Messwerte. Dynamisch bedeutet hier, dass die Anzahl Sensoren und Anzahl
              Kanäle pro Sensor erst zur Laufzeit bekannt sind.
============================================================================================================*/


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


typedef struct oakSensors_s* oakSensorsPtr_t;
typedef struct sensor_s* sensorPtr_t;
typedef struct channel_s* channelPtr_t;

typedef struct oakSensors_s {
        int nbrOfSensors;
        sensorPtr_t sensor;
} oakSensors_t;

typedef struct sensor_s {
        int serialNumber;
        int productID;
        char* deviceName;
        int nbrOfChannels;
        channelPtr_t channel;
} sensor_t;

typedef struct channel_s {
        float value;
        int unitId;
        char* unitString;
        char* channelName;
} channel_t;


void scanSensors(oakSensorsPtr_t oakSensors);
void printSensors(oakSensorsPtr_t oakSensors);
void setValues(oakSensorsPtr_t oakSensors, int nbrSensor, int nbrChannel, float value);
void setAllValues(oakSensorsPtr_t oakSensors);
float readValues(oakSensorsPtr_t oakSensors, int nbrSensor, int nbrChannel);
void printMeasurements(oakSensorsPtr_t oakSensors);
```

```
/**
 * The main procedure to run the application.
 * @param argc Number of command line arguments
 * @param argv The forwarded command line arguments
 * @return Application return (error) code
 */
int main(int argc, char** argv)
{
        // Scan Sensors
        oakSensors_t oakSensors;
        scanSensors(&oakSensors);

        // PrintOut Sensor Information
        printSensors(&oakSensors);

        printf("-----------------------------------------------------------------\n");
        // setAllValues - fake Data
        setAllValues(&oakSensors);

        // printMeasurements / readValues
        printMeasurements(&oakSensors);



        fflush(stdin); //flushes the input stream and gets rid of '\n'
        printf("\n\nPress [Enter] to exit...");
        getchar();
        return(EXIT_SUCCESS);
}

void printMeasurements(oakSensorsPtr_t oakSensors)
{
        int nbrOfSensors, i;
        int nbrOfChannels, j;
        sensorPtr_t sensor;
        channelPtr_t channel;

        // print Sensor
        nbrOfSensors = oakSensors->nbrOfSensors;
        for (i = 0; i < nbrOfSensors; i++) {
                sensor = oakSensors->sensor;
                nbrOfChannels = sensor[i].nbrOfChannels;
                printf("- %s:\n", sensor[i].deviceName);
                // print Channel, Value
                channel = sensor[i].channel;
                for (j = 0; j < nbrOfChannels; j++) {
                        printf("\t- %s: %f %s\n", channel[j].channelName, channel[j].value,
channel[j].unitString);
                }
        printf("\n");
        }
}


float readValues(oakSensorsPtr_t oakSensors, int nbrSensor, int nbrChannel)
{
        return oakSensors->sensor[nbrSensor].channel[nbrChannel].value;
}



void setAllValues(oakSensorsPtr_t oakSensors)
{
        // Atmospheric preasure
        setValues(oakSensors, 0, 0, 101539.0f);
        setValues(oakSensors, 0, 1, 297.700012f);
        // Humidity
        setValues(oakSensors, 1, 0, 297.299988f);
        setValues(oakSensors, 1, 1, 58.700001f);
        // Acceleration
        setValues(oakSensors, 2, 0, 0.01f);
        setValues(oakSensors, 2, 1, -0.02f);
        setValues(oakSensors, 2, 2, 9.81f);
        // Current
        setValues(oakSensors, 3, 0, 0.0075f);
        // Luminosity
        setValues(oakSensors, 4, 0, 12746.0f);
}
```

```
void setValues(oakSensorsPtr_t oakSensors, int nbrSensor, int nbrChannel, float value)
{
        oakSensors->sensor[nbrSensor].channel[nbrChannel].value = value;
}


void printSensors(oakSensorsPtr_t oakSensors)
{
        int nbrOfSensors, i;
        int nbrOfChannels, j;
        sensorPtr_t sensor;
        channelPtr_t channel;

        // print Sensor
        nbrOfSensors = oakSensors->nbrOfSensors;
        printf("%d Sensors\n", nbrOfSensors);
        for (i = 0; i < nbrOfSensors; i++) {
                sensor = oakSensors->sensor;
                nbrOfChannels = sensor[i].nbrOfChannels;
                printf("- %s (SN %d, PID %d) has %d channels\n", sensor[i].deviceName,
sensor[i].serialNumber, sensor[i].productID, nbrOfChannels);
                // print Channel
                channel = sensor[i].channel;
                for (j = 0; j < nbrOfChannels; j++) {
                        printf("\t- %s measures %s (%d)\n", channel[j].channelName, channel[j].unitString,
channel[j].unitId);
                }
        printf("\n");
        }
}


void scanSensors(oakSensorsPtr_t oakSensors)
{
        const int nbrOfSensors = 5;
        const int nbrOfChannelsSensor0 = 2;
        const int nbrOfChannelsSensor1 = 2;
        const int nbrOfChannelsSensor2 = 3;
        const int nbrOfChannelsSensor3 = 1;
        const int nbrOfChannelsSensor4 = 1;

        // Allocate Memory
        sensorPtr_t sensor = (sensorPtr_t) malloc(sizeof(struct sensor_s) * nbrOfSensors);
        channelPtr_t channelSensor0 = (channelPtr_t) malloc(sizeof(struct channel_s) * nbrOfChannelsSensor0);
        channelPtr_t channelSensor1 = (channelPtr_t) malloc(sizeof(struct channel_s) * nbrOfChannelsSensor1);
        channelPtr_t channelSensor2 = (channelPtr_t) malloc(sizeof(struct channel_s) * nbrOfChannelsSensor2);
        channelPtr_t channelSensor3 = (channelPtr_t) malloc(sizeof(struct channel_s) * nbrOfChannelsSensor3);
        channelPtr_t channelSensor4 = (channelPtr_t) malloc(sizeof(struct channel_s) * nbrOfChannelsSensor4);

        // Add Sensor & Co to oakSensor Object
        oakSensors->nbrOfSensors = nbrOfSensors;
        oakSensors->sensor = sensor;

        // Sensor 0 ------------------------------------------------------------------------
        sensor[0].deviceName    = strdup("Atmospheric preasure");
        sensor[0].serialNumber  = 111111;
        sensor[0].productID     = 1;
        sensor[0].nbrOfChannels = nbrOfChannelsSensor0;
        sensor[0].channel       = channelSensor0;
        // Sensor 0 - Channel 0
        channelSensor0[0].channelName = strdup("Pressure");
        channelSensor0[0].unitString  = strdup("Pa");
        channelSensor0[0].unitId      = 23;
        // Sensor 0 - Channel 1
        channelSensor0[1].channelName = strdup("Temperature");
        channelSensor0[1].unitString  = strdup("K");
        channelSensor0[1].unitId      = 37;

        // Sensor 1 ------------------------------------------------------------------------
        sensor[1].deviceName    = strdup("Humidity");
        sensor[1].serialNumber  = 22222;
        sensor[1].productID     = 2;
        sensor[1].nbrOfChannels = nbrOfChannelsSensor1;
        sensor[1].channel       = channelSensor1;
        // Sensor 1 - Channel 0
        channelSensor1[0].channelName = strdup("Temperatur");
        channelSensor1[0].unitString  = strdup("K");
        channelSensor1[0].unitId      = 37;
        // Sensor 1 - Channel 1
        channelSensor1[1].channelName = strdup("Humidity");
```

```
        channelSensor1[1].unitString  = strdup("% relative");
        channelSensor1[1].unitId      = 49;

        // Sensor 2 ------------------------------------------------------------------------
        sensor[2].deviceName    = strdup("Acceleration");
        sensor[2].serialNumber  = 33333;
        sensor[2].productID     = 3;
        sensor[2].nbrOfChannels = nbrOfChannelsSensor2;
        sensor[2].channel       = channelSensor2;
        // Sensor 2 - Channel 0
        channelSensor2[0].channelName = strdup("x");
        channelSensor2[0].unitString  = strdup("m/s^2");
        channelSensor2[0].unitId      = 55;
        // Sensor 2 - Channel 1
        channelSensor2[1].channelName = strdup("y");
        channelSensor2[1].unitString  = strdup("m/s^2");
        channelSensor2[1].unitId      = 55;
        // Sensor 2 - Channel 2
        channelSensor2[2].channelName = strdup("z");
        channelSensor2[2].unitString  = strdup("m/s^2");
        channelSensor2[2].unitId      = 55;

        // Sensor 3 ------------------------------------------------------------------------
        sensor[3].deviceName    = strdup("Current");
        sensor[3].serialNumber  = 44444;
        sensor[3].productID     = 4;
        sensor[3].nbrOfChannels = nbrOfChannelsSensor3;
        sensor[3].channel       = channelSensor3;
        // Sensor 3 - Channel 0
        channelSensor3[0].channelName = strdup("Current (4..20 mA)");
        channelSensor3[0].unitString  = strdup("A");
        channelSensor3[0].unitId      = 13;

        // Sensor 4 ------------------------------------------------------------------------
        sensor[4].deviceName    = strdup("Luminosity");
        sensor[4].serialNumber  = 55555;
        sensor[4].productID     = 5;
        sensor[4].nbrOfChannels = nbrOfChannelsSensor4;
        sensor[4].channel       = channelSensor4;
        // Sensor 4 - Channel 0
        channelSensor4[0].channelName = strdup("Illuminance");
        channelSensor4[0].unitString  = strdup("Lux");
        channelSensor4[0].unitId      = 87;
}
```

Übung 4

4.1 – Taschenrechner polnische Notation:

a) *Es soll das Programm des Taschenrechners mit polnischer Notation umgesetzt werden (gemäss Beispiel im Buch K&R S. 74 bis S.77 – siehe Anhang). Folgende zusätzliche Informationen hierzu:*
   - *Verwenden Sie das Template main.c aus dem Anhang als Vorlage*
   - *"stdin" ist hier das File "inputfile1.txt" in dem Pfad von „main“ (File mit netbeans als Editor verändern – Win-Text-Editor macht kein EOF wie gewünscht)*
   - *Sie können auch über die Konsole die Eingabe machen – EOF mittels ctrl+d*
   - *Alle Funktionen sind zuerst in einem Source-File („main...“ File) zu halten*
   - *Zeitbedarf 45': codieren, debugging und testen*

b) *Teilen Sie das obige Taschenrechner Programm in Quelldatei, Headerfiles und main auf, so dass Sie eine Aufteilung gemäss Buch S.80 (siehe Anhang) erreichen. Speichern Sie Ihr Projekt unter einem neuen Namen und testen Sie Ihr Programm entsprechend.*

c) *Optional: Erweitern Sie Ihren Taschenrechner mit den Funktionen sin, exp, power*

Lösung 4.1 a)

```
/**============================================================================================================
Hochschule Luzern - Technik & Architektur                                    Mikrocontroller - HS2012
Copyright 2012 Felix Rohrer                                                     http://hslu.ximit.ch
------------------------------------------------------------------------------------------------------------
Name        : Uebung_4-1.c
Author      : Felix Rohrer <felix.rohrer@stud.hslu.ch>
Version     : 2012-10-07v1.0
Description : Taschenrechner polnische Notation:
              o Verwenden Sie das Template main.c aus dem Anhang als Vorlage
              o "stdin" ist hier das File "inputfile1.txt" in dem Pfad von „main“ (File mit
                NetBeans als Editor verändern – Win-Text-Editor macht kein EOF wie gewünscht)
              o Sie können auch über die Konsole die Eingabe machen – EOF mittels ctrl+d
              o Alle Funktionen sind zuerst in einem Source-File („main...“ File) zu halten
              o Zeitbedarf 45': codieren, debugging und testen
============================================================================================================*/

#include <stdio.h>
#include <stdlib.h>             /* fuer atof() */
#include <ctype.h>              /* fuer isdigit() */

/* Konstanten */
#define MAXOP 100               /* max Laenge von Operand und/oder Operator */
#define BUFSIZE 100             /* Buffer Size */
#define MAXVAL 100              /* Stack Size */
#define NUMBER '0'              /* Anzeige, Zahl */

/* Globale Variablen */
char buf[BUFSIZE];              /* Buffer */
int bufp = 0;                   /* Buffer-Pointer */
double val[MAXVAL];             /* Stack */
int sp = 0;                             /* Stack-Pointer */

/* Funktions-Deklaration */
void push(double f);    /* push: f auf den Stack bringen */
double pop(void);               /* pop: Wert von Stack holen und liefern */
int getop(char s[]);    /* getop: naechsten Operator und /oder numerischen Operanden holen */
int getch(void);        /* getch: Zeichen holen mit Buffer */
void ungetch(int c);    /* ungetch: Zeichen wieder zurueckstellen wenn zuviel geholt */
```

```c
/**
 * The main procedure to run the application.
 * @param argc Number of command line arguments
 * @param argv The forwarded command line arguments
 * @return Application return (error) code
 */
int main(int argc, char** argv)
{
	/* Taschenrechner in umgekehrter polnischer Notation - Beispiel S. 74 K&R */

	int type;
	double op2;
	char s[MAXOP];

	//Oeffnet das File "inputfile.txt" in dem Pfad von main.c und nimmt das File als
	//Standard Eingabe anstelle Tastatur
	freopen("inputfile1.txt", "r", stdin);
	while ((type = getop(s)) != EOF) {
		switch (type) {
		case NUMBER:	// Zahl auf Stack pushen
			push(atof(s));
			break;
		case '+':		// + Plus
			push(pop() + pop());
			break;
		case '-':		// - Minus
			op2 = pop();
			push(pop() - op2);
			break;
		case '*':		// * Multiplikation
			push(pop() * pop());
			break;
		case '/':		// / Division
			op2 = pop();
			if (op2 != 0.0) {
				push(pop() / op2);
			} else {
				printf("Error: Division durch 0!\n");
			}
			break;
		case '\n':		// Resultat Ausgeben
			printf("Result:\t%.8g\n", pop());
			break;
		default:		// Unbekannter Befehl
				printf("Error: Unbekannter Befehle: %s\n", s);
				break;
		} // switch (type)
	} // while ( != EOF)

	fclose(stdin); // Schliesst die Standard-Eingabe stdin (das File von freopen)

	return(EXIT_SUCCESS);
}


/* push: f auf den Stack bringen */
void push(double f) {
	if (sp < MAXVAL) {
		val[sp++] = f;
	} else {
		printf("Error: Stack ist voll! %g kann nicht eingefuegt werden...\n", f);
	}
}

/*pop: Wert von Stack holen und liefern */
double pop(void) {
	if (sp > 0) {
		return val[--sp];
	} else {
		printf("Error: Stack ist leer!\n");
		return 0.0;
	}
}
```

```c
/*getop: naechsten Operator und /oder numerischen Operanden holen */
int getop(char s[]){
    int i, c;

    while ((s[0] = c = getch()) == ' ' || c == '\t')
                ; // whitespaces entfernen - tue nichts bei diese Zeichen

    s[1] = '\0'; // String terminieren
    if (!isdigit(c) && c != '.') {
                return c; /* keine Zahl --> Befehl */
        }
    i = 0;
    if (isdigit(c)) { /* ganzzahligen Teil sammeln */
        while (isdigit(s[++i] = c = getch())) // solange es eine Zahl ist
            ;
        }
    if (c == '.') { /* Dezimalstellen */
        while (isdigit(s[++i] = c = getch())) // solange es eine Zahl ist
            ;
        }
    s[i] = '\0'; // String terminieren
    if (c != EOF) { // solange nicht am Ende von der Zeile
        ungetch(c); // Zeichen zurück auf Stack
        }

    return NUMBER; // default return wert 0
}

/* getch: Zeichen holen mit Buffer */
int getch(void) {
        // return (bufp > 0) ? buf[--bufp] : getchar();
        if (bufp > 0) {
                return buf[--bufp];
        } else {
                return getchar();
        }
}

/* ungetch: Zeichen wieder zurueckstellen wenn zuviel geholt */
void ungetch(int c){
        if (bufp >= BUFSIZE) {
        printf("Error ungetch(): Zu viele Zeichen\n");
        } else {
        buf[bufp++] = c;
        }
}
```

Lösung 4.1 b) & c)

main.c

```
/**============================================================================================
Hochschule Luzern - Technik & Architektur                                  Mikrocontroller - HS2012
Copyright 2012 Felix Rohrer                                                   http://hslu.ximit.ch
--------------------------------------------------------------------------------------------
Name        : main.c
Author      : Felix Rohrer <felix.rohrer@stud.hslu.ch>
Version     : 2012-10-07v1.0
Description : Taschenrechner polnische Notation
============================================================================================*/

#include <stdio.h>
#include <stdlib.h>             /* fuer atof() */
#include <math.h>               /* fuer sin() und co */
#include "calc.h"               /* Eigene Methoden einbinden */

/* Konstanten */
#define MAXOP 100               /* max Laenge von Operand und/oder Operator */
#define PI 3.14159265   /* PI */

/**
 * The main procedure to run the application.
 * @param argc Number of command line arguments
 * @param argv The forwarded command line arguments
 * @return Application return (error) code
 */
int main(int argc, char** argv)
{
        /* Taschenrechner in umgekehrter polnischer Notation - Beispiel S. 74 K&R */

        int type;
        double op2;
        char s[MAXOP];

        //Oeffnet das File "inputfile.txt" in dem Pfad von main.c und nimmt das File als
        //Standard Eingabe anstelle Tastatur
        freopen("inputfile1.txt", "r", stdin);
        while ((type = getop(s)) != EOF) {
                switch (type) {
                case NUMBER:    // Zahl auf Stack pushen
                        push(atof(s));
                        break;
                case '+':               // + Plus
                        push(pop() + pop());
                        break;
                case '-':               // - Minus
                        op2 = pop();
                        push(pop() - op2);
                        break;
                case '*':               // * Multiplikation
                        push(pop() * pop());
                        break;
                case '/':               // / Division
                        op2 = pop();
                        if (op2 != 0.0) {
                                push(pop() / op2);
                        } else {
                                printf("Error: Division durch 0!\n");
                        }
                        break;
                case FN_SIN:    // Sinus
                        push(sin(pop() * PI / 180.0));
                        break;
                case FN_COS:    // Cosinus
                        push(cos(pop() * PI / 180.0));
                        break;
                case FN_TAN:    // Tangens
                        push(tan(pop() * PI / 180.0));
                        break;
                case FN_EXP:    // Exp, e^x
                        push(exp(pop()));
                        break;
                case FN_POWER:  // pow, x^y
                        op2 = pop();
                        push(pow(pop(), op2));
                        break;
                case '\n':              // Resultat Ausgeben
                        printf("Result:\t%.8g\n", pop());
```

```
                        break;
                default:        // Unbekannter Befehl
                        printf("Error: Unbekannter Befehl: %s\n", s);
                        break;
                } // switch (type)
        } // while ( != EOF)

        fclose(stdin); // Schliesst die Standard-Eingabe stdin (das File von freopen)

        return(EXIT_SUCCESS);
}
```

calc.h

```
/**=============================================================================================
Hochschule Luzern - Technik & Architektur                                   Mikrocontroller - HS2012
Copyright 2012 Felix Rohrer                                                    http://hslu.ximit.ch
-----------------------------------------------------------------------------------------------
Name        : calc.h
Author      : Felix Rohrer <felix.rohrer@stud.hslu.ch>
Version     : 2012-10-07v1.0
Description : Taschenrechner polnische Notation
=============================================================================================*/
/* Konstanten */
#define NUMBER '0'              /* Anzeige, Zahl */
#define FN_SIN '-1'             /* Sinus */
#define FN_COS '-2'             /* Cosinus */
#define FN_TAN '-3'             /* Tangens */
#define FN_EXP '-4'             /* Exponent */
#define FN_POWER '-5'   /* Power */

/* Funktions-Deklaration */
void push(double f);    /* push: f auf den Stack bringen */
double pop(void);               /* pop: Wert von Stack holen und liefern */
int getop(char s[]);    /* getop: naechsten Operator und /oder numerischen Operanden holen */
int getch(void);        /* getch: Zeichen holen mit Buffer */
void ungetch(int c);    /* ungetch: Zeichen wieder zurueckstellen wenn zuviel geholt */
```

getch.ch

```
/**=============================================================================================
Hochschule Luzern - Technik & Architektur                                   Mikrocontroller - HS2012
Copyright 2012 Felix Rohrer                                                    http://hslu.ximit.ch
-----------------------------------------------------------------------------------------------
Name        : getch.c
Author      : Felix Rohrer <felix.rohrer@stud.hslu.ch>
Version     : 2012-10-07v1.0
Description : Taschenrechner polnische Notation
              getch: Zeichen holen mit Buffer
              ungetch: Zeichen wieder zurueckstellen wenn zuviel geholt
=============================================================================================*/
#include <stdio.h>

/* Konstanten */
#define BUFSIZE 100             /* Buffer Size */

/* Globale Variablen */
char buf[BUFSIZE];              /* Buffer */
int bufp = 0;                   /* Buffer-Pointer */


/* getch: Zeichen holen mit Buffer */
int getch(void) {
        // return (bufp > 0) ? buf[--bufp] : getchar();
        if (bufp > 0) {
                return buf[--bufp];
        } else {
                return getchar();
        }
}

/* ungetch: Zeichen wieder zurueckstellen wenn zuviel geholt */
void ungetch(int c){
        if (bufp >= BUFSIZE) {
        printf("Error ungetch(): Zu viele Zeichen\n");
        } else {
        buf[bufp++] = c;
        }
}
```

getop.c

```
/**============================================================================================
Hochschule Luzern - Technik & Architektur                                   Mikrocontroller - HS2012
Copyright 2012 Felix Rohrer                                                    http://hslu.ximit.ch
--------------------------------------------------------------------------------------------------
Name        : getop.c
Author      : Felix Rohrer <felix.rohrer@stud.hslu.ch>
Version     : 2012-10-07v1.0
Description : Taschenrechner polnische Notation
              naechsten Operator und /oder numerischen Operanden holen
==============================================================================================*/

#include <stdio.h>
#include <stdlib.h>             /* fuer atof() */
#include <ctype.h>              /* fuer isdigit() */
#include <string.h>             /* fuer strcmp() */
#include "calc.h"               /* Eigene Methoden einbinden */

/*getop: naechsten Operator und /oder numerischen Operanden holen */
int getop(char s[]){
    int i, c, cOld;

    while ((s[0] = c = getch()) == ' ' || c == '\t')
                ; // whitespaces entfernen - tue nichts

    s[1] = '\0'; // String terminieren

    if (!isdigit(c) && c != '.') { // Befehl Sammeln
                cOld = c;
                i = 0;

                while (isalpha(s[++i] = c = getch())) { // solange es ein char ist
            ; // do nothing
                }
                s[i] = '\0'; // String terminieren
                if (!isalpha(c)) {
                        ungetch(c);
                }

                if (strcmp(s, "sin") == 0) {
                        return FN_SIN; // Sinus
                } else if (strcmp(s, "cos") == 0) {
                        return FN_COS; // Cosinus
                } else if (strcmp(s, "tan") == 0) {
                        return FN_TAN; // Tangens
                } else if (strcmp(s, "exp") == 0) {
                        return FN_EXP; // Exponent
                } else if (strcmp(s, "power") == 0) {
                        return FN_POWER; // Power
                } else {
                        return cOld; // keine Zahl --> Befehl
                }
        }
    i = 0;
    if (isdigit(c)) { /* ganzzahligen Teil sammeln */
        while (isdigit(s[++i] = c = getch())) // solange es eine Zahl ist
            ;
        }
    if (c == '.') { /* Dezimalstellen */
        while (isdigit(s[++i] = c = getch())) // solange es eine Zahl ist
            ;
        }
    s[i] = '\0'; // String terminieren
    if (c != EOF) { // solange nicht am Ende von der Zeile
        ungetch(c); // Zeichen zurück auf Stack
        }

    return NUMBER; // default return wert 0
}
```

stack.c

```
/**=============================================================================================
Hochschule Luzern - Technik & Architektur                                   Mikrocontroller - HS2012
Copyright 2012 Felix Rohrer                                                    http://hslu.ximit.ch
-----------------------------------------------------------------------------------------------
Name        : stack.c
Author      : Felix Rohrer <felix.rohrer@stud.hslu.ch>
Version     : 2012-10-07v1.0
Description : Taschenrechner polnische Notation
              Implementiert den Stack
=============================================================================================*/

#include <stdio.h>
#include "calc.h"               /* Eigene Methoden einbinden */

/* Konstanten */
#define MAXVAL 100              /* Stack Size */

/* Globale Variablen */
double val[MAXVAL];             /* Stack */
int sp = 0;                             /* Stack-Pointer */

/* push: f auf den Stack bringen */
void push(double f) {
        if (sp < MAXVAL) {
                val[sp++] = f;
        } else {
                printf("Error: Stack ist voll! %g kann nicht eingefuegt werden...\n", f);
        }
}

/*pop: Wert von Stack holen und liefern */
double pop(void) {
        if (sp > 0) {
                return val[--sp];
        } else {
                printf("Error: Stack ist leer!\n");
                return 0.0;
        }
}
```

4.2 – Makros

Schreiben Sie Makros:

- *MIN(x, y) liefert den kleineren der beiden Werte*
- *MAX(x, y) liefert den grösseren der beiden Werte*
- *SWAP(t,x,y) vertauscht die zwei Argumente x und y vom Typ t.*

Testen Sie Ihr Programm mit verschiedenen Testfällen.

```
/**============================================================================================
Hochschule Luzern - Technik & Architektur                                  Mikrocontroller - HS2012
Copyright 2012 Felix Rohrer                                                   http://hslu.ximit.ch
--------------------------------------------------------------------------------------------
Name        : Uebung_4-2.c
Author      : Felix Rohrer <felix.rohrer@stud.hslu.ch>
Version     : 2012-10-07v1.0
Description : Schreiben Sie Makros
              o MIN(x, y) liefert den kleineren der beiden Werte
              o MAX(x, y) liefert den grösseren der beiden Werte
              o SWAP(t,x,y) vertauscht die zwei Argumente x und y vom Typ t.
============================================================================================*/

#include <stdio.h>
#include <stdlib.h>

#define MIN(x, y)               ((x) < (y) ? (x) : (y))
#define MAX(x, y)               ((x) > (y) ? (x) : (y))
#define SWAP(t, x, y)   { t tmp; tmp = x; x = y; y = tmp; }

/**
 * The main procedure to run the application.
 * @param argc Number of command line arguments
 * @param argv The forwarded command line arguments
 * @return Application return (error) code
 */
int main(int argc, char** argv)
{
        int x, y;
        char* a, *b;

        x = 5;
        y = 10;
        a = "Hallo";
        b = "Du";

        printf("Value x: %i\n", x);
        printf("Value y: %i\n", y);
        printf("> Min: %i\n", MIN(x,y));
        printf("> Max: %i\n", MAX(x,y));
        printf("> Swap...\n");
        SWAP(int, x, y);
        printf("  > Value x: %i\n", x);
        printf("  > Value y: %i\n", y);

        printf("----------------------------------------\n");
        printf("String a: %s\n", a);
        printf("String b: %s\n", b);
        printf("> Min: %s\n", MIN(a,b));
        printf("> Max: %s\n", MAX(a,b));
        printf("> Swap...\n");
        SWAP(char*, a, b);
        printf("  > Value a: %s\n", a);
        printf("  > Value b: %s\n", b);



        fflush(stdin); //flushes the input stream and gets rid of '\n'
        printf("\n\nPress [Enter] to exit...");
        getchar();
        return(EXIT_SUCCESS);
}
```

4.3 – Ampel

Schreiben Sie eine kleine Ampel-Simulation, welche der Reihe nach die Zustände RED – GREEN – GREEN_BLINKING – YELLOW – RED – etc. annimmt. Teilen Sie das Programm in drei Files main.c, ampel.c und ampel.h auf und halten Sie den Zustand als statische, lokale Variable im File ampel.c fest. Das File ampel.c soll die drei Funktionen getState(), nextState() und printState() zur Verfügung stellen, welche Sie im main.c testen.

main.c

```
/**=========================================================================================================
Hochschule Luzern - Technik & Architektur                                   Mikrocontroller - HS2012
Copyright 2012 Felix Rohrer                                                    http://hslu.ximit.ch
---------------------------------------------------------------------------------------------------------
Name        : main.c
Author      : Felix Rohrer <felix.rohrer@stud.hslu.ch>
Version     : 2012-10-07v1.0
Description : Ampel
              Schreiben Sie eine kleine Ampel-Simulation, welche der Reihe nach die Zustände
              RED – GREEN – GREEN_BLINKING – YELLOW – RED – etc. annimmt.
=========================================================================================================*/

#include <stdio.h>
#include <stdlib.h>
#include <windows.h> /* Sleep() */
#include "ampel.h"

/**
 * The main procedure to run the application.
 * @param argc Number of command line arguments
 * @param argv The forwarded command line arguments
 * @return Application return (error) code
 */
int main(int argc, char** argv)
{
        int i;

        for(i=0; i<3; i++) // dreimal wiederholen
        {
                // Rot
                printState();
                printf("  State: %i\n", getState());
                Sleep(5000); // 6sec delay
                nextState();

                // Gruen
                printState();
                printf("  State: %i\n", getState());
                Sleep(3000); // 4sec delay
                nextState();

                // Gruen-Blinkend
                printState();
                printf("  State: %i\n", getState());
                Sleep(1000); // 1sec delay
                nextState();

                // Gelb
                printState();
                printf("  State: %i\n", getState());
                Sleep(1000); // 1sec delay
                nextState();

                //printf("\n----------------------\n\n");
        } // dreimal wiederholen


        fflush(stdin); //flushes the input stream and gets rid of '\n'
        printf("\n\nPress [Enter] to exit...");
        getchar();
        return(EXIT_SUCCESS);
}
```

ampel.h

```
/**============================================================================================================
Hochschule Luzern - Technik & Architektur                                        Mikrocontroller - HS2012
Copyright 2012 Felix Rohrer                                                         http://hslu.ximit.ch
------------------------------------------------------------------------------------------------------------
Name        : ampel.h
Author      : Felix Rohrer <felix.rohrer@stud.hslu.ch>
Version     : 2012-10-07v1.0
Description : Ampel
============================================================================================================*/

/**
 * Gibt den aktuellen Zustand zurück
 * @return Int-Wert für den aktuellen Zustand
 */
int getState(void);

/**
 * Setz die Ampel auf den nächsten Status, Zustand
 */
void nextState(void);

/**
 * Gibt den aktuellen Zustand auf stdout aus
 */
void printState(void);
```

ampel.c

```
/**=========================================================================================================
Hochschule Luzern - Technik & Architektur                                   Mikrocontroller - HS2012
Copyright 2012 Felix Rohrer                                                    http://hslu.ximit.ch
---------------------------------------------------------------------------------------------------------
Name        : ampel.c
Author      : Felix Rohrer <felix.rohrer@stud.hslu.ch>
Version     : 2012-10-07v1.0
Description : Ampel
=========================================================================================================*/

#include <stdio.h>
#include "ampel.h"

typedef enum State_ {
        RED, // 0
        GREEN, // 1
        GREEN_BLINKING, // 2
        YELLOW, // 3
        STATE_COUNT
} State;

static int currState = 0;


int getState(void)
{
        return currState;
}

void nextState(void)
{
        if (currState < (STATE_COUNT - 1)) {
                currState++;
        } else {
                currState = 0;
        }
}


void printState(void)
{
        switch (currState) {
        case RED:
                printf("Rot\n");
                break;
        case GREEN:
                printf("Gruen\n");
                break;
        case GREEN_BLINKING:
                printf("Gruen blinkend\n");
                break;
        case YELLOW:
                printf("Gelb\n");
                break;
        } // switch
}
```

4.4 – Message Queue

Machen Sie eine einfache „Message Queue“, welche mind. die drei Funktionen initialize, enqueue and dequeue enthält. Die „Messages“ sollen aus einem Integerwert bestehen. enqueue schreibt die neue „Message“ an das Ende der „Queue“, dequeue liest das nächste Element aus der „Queue“. Falls keine „Messages“ vorhanden sind soll die Funktion dequeue blockieren. (Tipp: siehe verkettete Liste) Achten Sie auf ein sauberes „Allozieren und Freigeben“ der Speicherplätze und testen Sie Ihr Programm entsprechend!

main.c

```
/**=============================================================================================
Hochschule Luzern - Technik & Architektur                                   Mikrocontroller - HS2012
Copyright 2012 Felix Rohrer                                                    http://hslu.ximit.ch
-----------------------------------------------------------------------------------------------
Name        : main.c
Author      : Felix Rohrer <felix.rohrer@stud.hslu.ch>
Version     : 2012-10-07v1.0
Description : Message Queue
              Machen Sie eine einfache „Message Queue“, welche mind. die drei Funktionen
              initialize, enqueue and dequeue enthält. Die „Messages“ sollen aus einem Integerwert
              bestehen. enqueue schreibt die neue „Message“ an das Ende der „Queue“, dequeue liest
              das nächste Element aus der „Queue“. Falls keine „Messages“ vorhanden sind soll die
              Funktion dequeue blockieren.
=============================================================================================*/

#include <stdio.h>
#include <stdlib.h>
#include "queue.h"


/**
 * The main procedure to run the application.
 * @param argc Number of command line arguments
 * @param argv The forwarded command line arguments
 * @return Application return (error) code
 */
int main(int argc, char** argv)
{

        // init MessageQueue
        MessageQueue* aQueue = initialize();
        int i;

        // Write Queue
        for(i=0; i<10; i++) {
                printf("in:  %d\n",enqueue(aQueue,i));
        }
        printf("-----------------------------------------------------\n");

        // Read Queue
        for(i=0; i<5; i++) {
                printf("out: %d\n",dequeue(aQueue));
        }
        printf("-----------------------------------------------------\n");

        // Write Queue
        for(i=500; i>490; i--) {
                printf("in:  %d\n",enqueue(aQueue,i));
        }
        printf("-----------------------------------------------------\n");

        // Read Queue
        for(i=0; i<15; i++) {
                printf("out: %d\n",dequeue(aQueue));
        }
        printf("-----------------------------------------------------\n");


        // DEAD LOCK
        // Read Queue --> Queue is empty!
        printf("out: %d\n",dequeue(aQueue));

        return(EXIT_SUCCESS);
}
```

queue.h

```
/**============================================================================================================
Hochschule Luzern - Technik & Architektur                                        Mikrocontroller - HS2012
Copyright 2012 Felix Rohrer                                                          http://hslu.ximit.ch
------------------------------------------------------------------------------------------------------------
Name        : queue.h
Author      : Felix Rohrer <felix.rohrer@stud.hslu.ch>
Version     : 2012-10-07v1.0
Description : Message Queue
============================================================================================================*/


typedef struct Node {
  struct Node* next;            // next node pointer
  int data;                             // value
} Node;

typedef struct MessageQueue {
  Node* head;                           // first node pointer
  Node* rear;                           // rear node pointer
  int size;                                     // queue size
} MessageQueue;


/**
 * Allocate Memory for the Queue
 * @return Pointer for the new Queue
 */
MessageQueue* initialize();

/**
 * Add a new Value to the Queue
 * @param aQueue Pointer to the Queue
 * @param aValue Value to add
 * @return Value that has been added
 */
int enqueue(MessageQueue *aQueue, int aValue);

/**
 * Get a Value from the Queue
 * @param aQueue Pointer to the Queue
 * @return Value from the Queue
 */
int dequeue(MessageQueue *aQueue);
```

queue.c

```
/**=============================================================================================
Hochschule Luzern - Technik & Architektur                                  Mikrocontroller - HS2012
Copyright 2012 Felix Rohrer                                                   http://hslu.ximit.ch
-----------------------------------------------------------------------------------------------
Name        : queue.c
Author      : Felix Rohrer <felix.rohrer@stud.hslu.ch>
Version     : 2012-10-07v1.0
Description : Message Queue
==============================================================================================*/

#include <stdio.h>
#include <stdlib.h>
#include <windows.h>
#include "queue.h"


MessageQueue* initialize() {
        //create a new Queue
        MessageQueue* aQueue = (MessageQueue*) malloc(sizeof(MessageQueue));     /* allocate Memory */
        aQueue->size=0;                                                          /* init Size 0 */
        return aQueue;
}

int enqueue(MessageQueue *aQueue, int aValue) {
        //create a new Node
        Node* newNode = (Node*) malloc(sizeof(Node));    /* allocate Memory */
        newNode->data = aValue;                          /* set the data of the Node */
        newNode->next = NULL;                            /* newNode is the last Element (no next) */

        if (aQueue->size == 0) {                         /* check if the Queue is empty */
                aQueue->head=newNode;                    /* if the Queue is empty set the HEAD */
        } else {
                aQueue->rear->next = newNode;            /* if the Queue is not empty set the next node */
        }

        aQueue->rear=newNode;                            /* set the new Node as the new REAR */
        aQueue->size++;                                  /* increase the size of the Queue */
        return aValue;
}

int dequeue(MessageQueue *aQueue) {
        int tmpData;
        Node* removeNode;

        while (aQueue->size == 0) {             /* wait if the queue is empty --> DEAD LOCK !!! */
                printf("Queue is empty...\n");
                Sleep(2000);
        }

        tmpData = aQueue->head->data;           /* get Data */
        removeNode = aQueue->head;              /* temp var. used for free() memory */
        aQueue->head = aQueue->head->next;      /* remove Node from Queue */

        free(removeNode);                       /* release memory */
        aQueue->size--;                         /* decrease Queue size */
        return tmpData;
}
```

5.1 – Matrix erstellen

Schreiben Sie die Funktion createMatrix(…), welche eine Matrix erzeugt und die Matrix mit einem vorgegebenen Werte initialisiert (der Wert wird als dritter Übergabeparameter übergeben).

main.c

```
/**============================================================================================================
Hochschule Luzern - Technik & Architektur                                        Mikrocontroller - HS2012
Copyright 2012 Felix Rohrer                                                          http://hslu.ximit.ch
------------------------------------------------------------------------------------------------------------
Project     : Uebung_5-1
Name        : main.c
Author      : Felix Rohrer <felix.rohrer@stud.hslu.ch>
Version     : 2012-10-27v1.0
Description : Matrix erstellen
              Schreiben Sie die Funktion createMatrix(…), welche eine Matrix erzeugt und die
              Matrix mit einem vorgegebenen Werte initialisiert (der Wert wird als dritter
              Übergabeparameter übergeben).
============================================================================================================*/
#include <stdio.h>
#include <stdlib.h>
#include "matrix.h"

/**
 * The main procedure to run the application.
 * @param argc Number of command line arguments
 * @param argv The forwarded command line arguments
 * @return Application return (error) code
 */
int main(int argc, char** argv) {

  // set number of rows, cols and the initialization value
  const int rows = 4;
  const int cols = 3;
  const int value = 3;

  // create 3x4 matrix with initialization value 3
  int **m = createMatrix(rows,cols,value);

  if(m != NULL) {
    // print matrix to standard output
    printMatrix(m, rows, cols);

    // free memory
    freeMatrix(m, rows);
  }

    return (EXIT_SUCCESS);
}
```

matrix.h

```
/**=============================================================================================
Hochschule Luzern - Technik & Architektur                                  Mikrocontroller - HS2012
Copyright 2012 Felix Rohrer                                                   http://hslu.ximit.ch
-----------------------------------------------------------------------------------------------
Project     : Uebung_5-1
Name        : matrix.h
Author      : Felix Rohrer <felix.rohrer@stud.hslu.ch>
Version     : 2012-10-27v1.0
Description : Matrix erstellen
==============================================================================================*/
#ifndef MATRIX_H
#define MATRIX_H

/**
 * Creates a matrix by a given number of rows and a given number of columns.
 * The matrix is filled with the values given by the third parameter.
 * Note: Don't forget to dealloc space after using!
 *
 * @param rows Number of rows for the matrix
 * @param cols Number of columns for the matrix
 * @param value Initialization value of the cells
 * @return <rows> x <cols> matrix filled with <value>
 * @see freeMatrix(int **matrix, int rows, int cols)
 */
int **createMatrix(int rows, int cols, int value);

/**
 * Filles a previously created matrix with a given value.
 *
 * @param matrix The matrix to be filled
 * @param rows Number of rows of the matrix
 * @param cols Number of columns of the matrix
 * @param value Initialization value of the cells
 * @return nothing
 */
void fillMatrix(int **matrix, int rows, int cols, int value);

/**
 * Prints a previously created matrix to the standard output
 *
 * @param matrix The matrix to be printed
 * @param rows Number of rows for the matrix
 * @param cols Number of columns for the matrix
 * @return nothing
 */
void printMatrix(int *const *matrix, int rows, int cols);

/**
 * Deallocates previously reserved memory
 *
 * @param matrix The matrix to deallocate
 * @param rows Number of rows for the matrix
 * @return <rows> x <cols> matrix filled with <value>
 */
void freeMatrix(int **matrix, int rows);

#endif  /* MATRIX_H */
```

matrix.c

```
/**=============================================================================================
Hochschule Luzern - Technik & Architektur                                  Mikrocontroller - HS2012
Copyright 2012 Felix Rohrer                                                   http://hslu.ximit.ch
-----------------------------------------------------------------------------------------------
Project     : Uebung_5-1
Name        : matrix.c
Author      : Felix Rohrer <felix.rohrer@stud.hslu.ch>
Version     : 2012-10-27v1.0
Description : Matrix erstellen
=============================================================================================*/
#include <stdio.h>
#include <stdlib.h>
#include "matrix.h"

int** createMatrix(int rows, int cols, int value) {
  int row;               // used for iteration
  int** matrix = NULL;    // placeholder for matrix

  // allocate memory
  matrix = (int**) malloc(rows*sizeof(int*));
  if(matrix != NULL) {    // if succesfull allocated

    // allocate rows
    for(row=0; row<rows; row++) {
      matrix[row] = (int*) malloc(cols*sizeof(int));
      if(matrix[row] == NULL) {
        printf("Unable to allocate memory!\n");
        // free if unsuccesfull allocation
        freeMatrix(matrix, row);
        return NULL;
      }
    }

    // fill matrix with values
    fillMatrix(matrix, rows, cols, value);
  }
  else printf("Unable to allocate memory!\n");

  return matrix;
}

void fillMatrix(int** matrix, int rows, int cols, int value) {
  int row, col;   // used for iteration

  // fill matrix
  for(row = 0; row < rows; row++)
    for(col = 0; col < cols; col++)
      matrix[row][col] = value;
}

void printMatrix(int* const* matrix, int rows, int cols) {
  int row, col;   // used for iteration

  for(row = 0; row < rows; row++) {
    printf("{");   // print row boundary
    for(col = 0; col < cols; col++) {
      // print cell. separator only if first cell of row
      printf(col == 0 ? "%6d" : ",%6d", matrix[row][col]);
    }
    printf("}\n");   // print row boundary
  }
}

void freeMatrix(int** matrix, int rows) {
  int row;        // used for iteration

  // free columns first
  for (row = 0; row < rows; row++){
    free(matrix[row]);
  }

  // free rows
  free(matrix);
}
```

Übung 5

5.2 – Zeigerarithmetik

Entwerfen Sie eine Übung analog der Aufgabe in Kapitel 3 wo Sie Zeigerarithmetik einsetzen.

main.c

```
/**=============================================================================================
Hochschule Luzern - Technik & Architektur                                  Mikrocontroller - HS2012
Copyright 2012 Felix Rohrer                                                    http://hslu.ximit.ch
-----------------------------------------------------------------------------------------------
Project     : Uebung_5-2
Name        : main.c
Author      : Felix Rohrer <felix.rohrer@stud.hslu.ch>
Version     : 2012-10-27v1.0
Description : Entwurf einer Übung zum Thema Zeigerarithmetik
              Entwerfen Sie eine Übung analog der Aufgabe in Kapitel 3 wo Sie Zeigerarithmetik
              einsetzen.
=============================================================================================*/
#include <stdio.h>
#include <stdlib.h>
#include "zeigerarithmetic.h"

/*
 * The main procedure to run the application.
 * @param argc Number of command line arguments
 * @param argv The forwarded command line arguments
 * @return Application return (error) code
 */
int main(int argc, char** argv)
{
  int field[SIZE][SIZE];
  int posX = 0;
  int posY = 0;
  char input;

  /* fill the entire field with zero */
  memset(field, 0, sizeof(field));

  /* set a one in the first field */
  field[posY][posX] = 1;

  /* loop until a 'x' has been pressed (see below) */
  do
  {
    /* clear the screen - this makes the "matrix" appear at the
       same position all the time (very nice ;-) ) */
    //system("cls");
    system("cmd.exe /c cls"); // Cygwin hack

    /* prompt */
    printf("Zug [w hoch, a links, s runter, " \
           "d rechts] eingeben; beenden mit x\n");

    /* print the entire field on the console */
    printField(&field[0][0]);

    /* wait until a key has been pressed */
    while (!isalnum(input = (char)getchar()));

    /* move the "one" in the field, according to input */
    move(&field[0][0], &posX, &posY, input);
  } while(input != 'x');

  /* exit */
        return(EXIT_SUCCESS);
}
```

zeigerarithmetic.h

```
/**=============================================================================================
Hochschule Luzern - Technik & Architektur                                  Mikrocontroller - HS2012
Copyright 2012 Felix Rohrer                                                   http://hslu.ximit.ch
-----------------------------------------------------------------------------------------------
Project     : Uebung_5-2
Name        : zeigerarithmetic.h
Author      : Felix Rohrer <felix.rohrer@stud.hslu.ch>
Version     : 2012-10-27v1.0
Description : Matrix erstellen
              Schreiben Sie die Funktion createMatrix(…), welche eine Matrix erzeugt und die
              Matrix mit einem vorgegebenen Werte initialisiert (der Wert wird als dritter
              Übergabeparameter übergeben).
=============================================================================================*/

#ifndef ZEIGERARITHMETIC_H
#define ZEIGERARITHMETIC_H

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

/* function prototypes */
void printField(const int *field);
void move(int *field, int *posX, int *posY, char in);

/* define the size of the field */
#define SIZE 7

/* print an error message for mysterious SIZE values */
#if SIZE < 2
#error "SIZE IS SENSELESS"
#elif SIZE > 39
#error "SIZE IS LARGER THAN CONSOLE :-)"
#endif



#endif  /* ZEIGERARITHMETIC_H */
```

zeigerarithmetic.c

```
/**=============================================================================================
Hochschule Luzern - Technik & Architektur                                  Mikrocontroller - HS2012
Copyright 2012 Felix Rohrer                                                   http://hslu.ximit.ch
-----------------------------------------------------------------------------------------------
Project     : Uebung_5-2
Name        : zeigerarithmetic.c
Author      : Felix Rohrer <felix.rohrer@stud.hslu.ch>
Version     : 2012-10-27v1.0
Description : Matrix erstellen
              Schreiben Sie die Funktion createMatrix(…), welche eine Matrix erzeugt und die
              Matrix mit einem vorgegebenen Werte initialisiert (der Wert wird als dritter
              Übergabeparameter übergeben).
=============================================================================================*/
#include <stdio.h>
#include "zeigerarithmetic.h"

/* print the field */
void printField(const int *field)
{
  int row, col;

  /* newline */
  printf("\n");

  /* loop through all the rows */
  for(row = 0; row < SIZE; row++)
  {
    /* loop through all the columns */
    for(col = 0; col < SIZE; col++)
    {
      printf("%d ", *(field + row * SIZE + col));
    }
    printf("\n");
  }
}


/* moves the "one" around */
void move(int *field, int *posX, int *posY, char in)
{
  /* overwrite the old position with zero */
  *(field + *posY * SIZE + *posX) = 0;

  /* calculate the new position */
  switch(in)
  {
    case 'w':
    {
      (*posY)--;
    }
    break;

    case 'a':
    {
      (*posX)--;
    }
    break;

    case 's':
    {
      (*posY)++;
    }
    break;

    case 'd':
    {
      (*posX)++;
    }
    break;

    /* should not happen, normally :-) */
    default:
      return;
  }
```

```
  /* check each of the coordinates against upper / lower bounds */
  if(*posX < 0)
  {
    *posX = SIZE - 1;
  }

  if(*posX > SIZE - 1)
  {
    *posX = 0;
  }

  if(*posY < 0)
  {
    *posY = SIZE - 1;
  }

  if(*posY > SIZE - 1)
  {
    *posY = 0;
  }

  /* set the new position */
  *(field + *posY * SIZE + *posX) = 1;
}
```

5.3 – Zeiger auf Funktion

Implementieren Sie in der vorgegebenen Ampelsteuerung eine Callback-Funktion, damit die Zeiten der einzelnen Phasen ‚von aussen' eingestellt werden können.

Studieren Sie den vorgegebenen Code und schauen Sie sich an, wie das Programm bis jetzt läuft. Kopieren Sie dazu den Code von unten in die drei Dateien semaphore.h, semaphore.c und main.c. Anschliessend machen Sie die Änderungen.

main.c

```
/**============================================================================================================
Hochschule Luzern - Technik & Architektur                                        Mikrocontroller - HS2012
Copyright 2012 Felix Rohrer                                                         http://hslu.ximit.ch
------------------------------------------------------------------------------------------------------------
Project     : Uebung_5-3
Name        : main.c
Author      : Felix Rohrer <felix.rohrer@stud.hslu.ch>
Version     : 2012-10-27v1.0
Description : Zeiger auf Funktion / Threads
              Implementieren Sie in der vorgegebenen Ampelsteuerung eine Callback-Funktion, damit
              die Zeiten der einzelnen Phasen ‚von aussen' eingestellt werden können.
============================================================================================================*/
#include <stdio.h>
#include <stdlib.h>
#include "semaphore.h"

/**
* Determins the length of each phase of the semaphor.
* @param phase actuel phase
* @return Time in ms for the corresponding phase
*/
int getPhaseTime(SemaphorStates_t phase)
{
        int rc = 1;
        switch (phase) {
        case GREEN:          rc = 6;  break;
        case GREEN_BLINKING: rc = 2;  break;
        case YELLOW:         rc = 2;  break;
        case RED:            rc = 10; break;
        default:             rc = 2;  break;
        }
        return rc;
}

/**
* Starts the semaphor and defines with the the implementation of a callback
* function the length of the various periods.
*/
int main(int argc, char** argv)
{
        //Set callback function
        setSemaphorCallbackFunction(getPhaseTime);

        printf("Starting the semaphor...\n");
        startSemaphore();

        // Let the semaphor run for 1 minute
        sleep(60);
        printf("Stopping the semaphor...\n");
        stopSemaphor();
        printf("Semaphor stopped...\n");

        return (EXIT_SUCCESS);
}
```

semaphore.h

```
/**=========================================================================================================
Hochschule Luzern - Technik & Architektur                                   Mikrocontroller - HS2012
Copyright 2012 Felix Rohrer                                                     http://hslu.ximit.ch
-----------------------------------------------------------------------------------------------------------
Project     : Uebung_5-3
Name        : semaphore.h
Author      : Felix Rohrer <felix.rohrer@stud.hslu.ch>
Version     : 2012-10-27v1.0
Description : Zeiger auf Funktion / Threads
=========================================================================================================*/
#ifndef SEMAPHORE_H
#define SEMAPHORE_H

/**
 * Implementation of a semaphor with a thread.
 * Uses a callback function to define the various times.
 */
typedef enum SemaphorStates_ {
    GREEN = 1,
    GREEN_BLINKING,
    YELLOW,
    RED
} SemaphorStates_t;

/**
 * Set the callback function that defines the time of each phase.
 * @param b pointer to function that returns the phase length in seconds.
 *        The function gets as Parameter the current semaphore state.
 */
void setSemaphorCallbackFunction(int (*b) (SemaphorStates_t phase));

/**
 * Starts the semaphor.
 * Within a thread, the semaphor changes its state from GREEN to GREEN_BLINKING
 * to YELLOW to RED to GREEN to ... .
 * After each state change, the callbackfunction defined by setCallbackFunction(...)
 * will be called to get the length of the corresponding phase.
 */
void startSemaphore();

/**
 * Stop the semaphor.
 */
void stopSemaphor();

/**
 * Definiert die Semaphorezeiten
 */
int getPhaseTime(SemaphorStates_t phase);

#endif  /* SEMAPHORE_H */
```

semaphore.c

```
/**=============================================================================================
Hochschule Luzern - Technik & Architektur                                  Mikrocontroller - HS2012
Copyright 2012 Felix Rohrer                                                   http://hslu.ximit.ch
-----------------------------------------------------------------------------------------------
Project     : Uebung_5-3
Name        : semaphore.c
Author      : Felix Rohrer <felix.rohrer@stud.hslu.ch>
Version     : 2012-10-27v1.0
Description : Zeiger auf Funktion / Threads
==============================================================================================*/
#include <stdio.h>
#include <string.h>
#include <pthread.h>
#include "semaphore.h"

static int threadRunning = 0;
static SemaphorStates_t state = RED;
static pthread_t threadId; // Thread stuff
static pthread_attr_t threadAttr;
int (*f)(SemaphorStates_t phase);
int sleepTime;

/**
 * Printout current thread state
 */
static void printThreadState(int sleepTime) {
    char stateStr[64];

    switch (state) {
        case GREEN: strcpy(stateStr, "GREEN");
            break;
        case GREEN_BLINKING: strcpy(stateStr, "GREEN BLINKING");
            break;
        case YELLOW: strcpy(stateStr, "YELLOW");
            break;
        case RED: strcpy(stateStr, "RED");
            break;
        default: strcpy(stateStr, "UNDEFINED");
            break;
    }
    printf("The semaphor is %-14s for %2d seconds\n", stateStr, sleepTime);
    fflush(stdout);
}

static void* threadFunction(void* args) {
    while (threadRunning) {
        switch (state) {
            case GREEN:
                state = GREEN_BLINKING;
                break;
            case GREEN_BLINKING:
                state = YELLOW;
                break;
            case YELLOW:
                state = RED;
                break;
            case RED:
                state = GREEN;
                break;
            default: // Should not happen, but added error recovery
                state = RED;
                break;
        }

        if (f != NULL) {
            sleepTime = f(state);
        }

        printThreadState(sleepTime);
        sleep(sleepTime);
    }
    printf("Semaphor thread stopped\n");

    return NULL;
}
```

```
void setSemaphorCallbackFunction(int (*b) (SemaphorStates_t phase)) {
    f = b;
}

void startSemaphore() {
    threadRunning = 1;

    // Thread starten
    pthread_attr_init(&threadAttr);
    // Hier wird als drittes Argument ein Zeiger auf die Thread-Funktion übergeben.
    pthread_create(&threadId, &threadAttr, threadFunction, NULL);
}

void stopSemaphor() {
    threadRunning = 0;

    // Wait until thread is finished
    pthread_join(threadId, NULL);
    pthread_attr_destroy(&threadAttr);
}
```

5.4 – Mehrdimensionales Array mit Druckmesswerten

Ein Messfeld von Drucksensoren in einer 2D-Ebene liefert Ihnen Druckmesswerte, welche Sie in einem 2-dimensionalen Array speichern sollen. Die Messwerte bestehen jeweils aus einem Druckwert (float) und einem Einheitsvorsatz (Mega, Kilo, Zero, Milli, Mikro).

Definieren Sie eine passende Datenstruktur für den Druckmesswert.

Kreieren Sie dynamisch ein variables 2-dimensionales Array mit diesen Druckmesswerten und initialisieren diese allgemein mit den Werten wie im Beispiel unten angegeben. Verwenden Sie bitte Pointer-auf-Pointer, Enum und Strukturen wie in der Vorlesung gezeigt.

Schreiben Sie zusätzlich eine Funktion, welche das Array entsprechend formatiert auf der Konsole ausgibt.

main.c

```
/**============================================================================================
Hochschule Luzern - Technik & Architektur                              Mikrocontroller - HS2012
Copyright 2012 Felix Rohrer                                               http://hslu.ximit.ch
--------------------------------------------------------------------------------------------
Project     : Uebung_5-4
Name        : main.c
Author      : StalderJ
Version     : 2012-10-27v1.0
Description : Mehrdimensionales Array mit Druckmesswerten
              Ein Messfeld von Drucksensoren in einer 2D-Ebene liefert Ihnen Druckmesswerte, welche
              Sie in einem 2-dimensionalen Array speichern sollen. Die Messwerte bestehen jeweils
              aus einem Druckwert (float) und einem Einheitsvorsatz (Mega, Kilo, Zero, Milli,Mikro).
              Definieren Sie eine passende Datenstruktur für den Druckmesswert.
              Kreieren Sie dynamisch ein variables 2-dimensionales Array mit diesen Druckmesswerten
              und initialisieren diese allgemein mit den Werten wie im Beispiel unten angegeben.
              Verwenden Sie bitte Pointer-auf-Pointer, Enum und Strukturen wie in der Vorlesung
              gezeigt. Schreiben Sie zusätzlich eine Funktion, welche das Array entsprechend
              formatiert auf der Konsole ausgibt.
============================================================================================*/
#include <stdlib.h>
#include "sensorgrid.h"

/*
 * Program control
 */
int main(int argc, char** argv) {

    // Sensordata Struct
    sdata_t** grid;

    const int rows = 3;
    const int cols = 4;

    // Create the array
    createGrid(&grid, rows, cols);

    // fill in some sensless data
    writeToGrid(grid, 0, 0, 0.00001);
    writeToGrid(grid, 0, 1, 0.1);
    writeToGrid(grid, 0, 2, 0.2);
    writeToGrid(grid, 0, 3, 0.3);

    writeToGrid(grid, 1, 0, 1.0);
    writeToGrid(grid, 1, 1, 1.1);
    writeToGrid(grid, 1, 2, 1.2);
    writeToGrid(grid, 1, 3, 1.3);

    writeToGrid(grid, 2, 0, 2.0);
    writeToGrid(grid, 2, 1, 2000.1);
    writeToGrid(grid, 2, 2, 2000000.2);
    writeToGrid(grid, 2, 3, 2.3);

    // put it out to console
    displayGrid(grid, rows, cols);

    // Free what would be free-ed anyway
    free(grid);
    return (EXIT_SUCCESS);
}
```

sensorgrid.h

```
/**============================================================================================================
Hochschule Luzern - Technik & Architektur                                       Mikrocontroller - HS2012
Copyright 2012 Felix Rohrer                                                        http://hslu.ximit.ch
------------------------------------------------------------------------------------------------------------
Project     : Uebung_5-4
Name        : sensorgrid.h
Author      : StalderJ
Version     : 2012-10-27v1.0
Description : Mehrdimensionales Array mit Druckmesswerten
============================================================================================================*/
#ifndef SENSORGRID_H
#define SENSORGRID_H

/**
 * Defines the fraction.
 */
typedef enum fraction {
    Mega = 'M',
    Kilo = 'K',
    Zero = ' ',
    Milli = 'm',
    Mikro = '\xb5'
} fraction_t;

/**
 * Defines the data structure.
 */
typedef struct sensordata {
    float pressure;
    fraction_t frac;
} sdata_t;

/**
 * Creates a grid of sensor data containers.
 * @param grid Reference to the grid that will be filled.
 * @param x First Dimension of the grid.
 * @param y Second Dimensions of the grid.
 */
void createGrid(sdata_t ***grid, unsigned int x, unsigned int y);

/**
 * Displays the grid on console.
 * @param grid Reference to the grid.
 * @param x Size of the grid in x direction.
 * @param y Size of the grid in y direction.
 */
void displayGrid(sdata_t **grid, unsigned int x, unsigned int y);

/**
 * Writes a value to the grid at position [x,y].
 * @param grid Reference to the grid.
 * @param x X-Dimension parameter.
 * @param y Y-Dimension parameter.
 * @param val Measure value.
 */
void writeToGrid(sdata_t **grid, unsigned int x, unsigned int y, float val);

#endif  /* SENSORGRID_H */
```

sensorgrid.c

```
/**=============================================================================================================
Hochschule Luzern - Technik & Architektur                                        Mikrocontroller - HS2012
Copyright 2012 Felix Rohrer                                                         http://hslu.ximit.ch
-------------------------------------------------------------------------------------------------------------
Project     : Uebung_5-4
Name        : sensorgrid.c
Author      : StalderJ
Version     : 2012-10-27v1.0
Description : Mehrdimensionales Array mit Druckmesswerten
=============================================================================================================*/
#include "sensorgrid.h"

void createGrid(sdata_t ***g1, unsigned int x, unsigned int y){
    int i;
    sdata_t **grid;

    // Create the x dimension of the grid
    grid = (sdata_t**) malloc(x* sizeof(sdata_t*));
    // Create an y-dimension in every x direction
    for(i=0;i < x;i++){
        *(grid+i) = (sdata_t*) malloc(y * sizeof(sdata_t));
    }

    // Write the address where the first grid node is stored
    // directly to the pointer-var in the caller function
    (*g1) = grid;
}

void displayGrid(sdata_t **grid, unsigned int x, unsigned int y){
    int i, j;

    // Walk through every row
    for(i=0;i < x;i++){
        printf("|");
        // Walk through every element in the row
        for(j=0;j < y;j++){
            printf("%10g%c |", grid[i][j].pressure, grid[i][j].frac);
        }
        printf("\n");
    }
}

void writeToGrid(sdata_t **grid, unsigned int x, unsigned int y, float val) {
    // Searches for a matching fraction and store value with fraction
    if (val > 1000000) {
        grid[x][y].pressure = val / (1000 * 1000);
        grid[x][y].frac = Mega;
    } else if(val > 1000) {
        grid[x][y].pressure = val / 1000;
        grid[x][y].frac = Kilo;
    } else if(val < 0.001) {
        grid[x][y].pressure = val * (1000 * 1000);
        grid[x][y].frac = Mikro;
    } else if(val < 1) {
        grid[x][y].pressure = val * 1000;
        grid[x][y].frac = Milli;
    } else {
        grid[x][y].pressure = val;
        grid[x][y].frac = Zero;
    }
}
```

Übung 6

6.1 – Projekt Adressverwaltung

In dieser Übung vertiefen Sie noch einmal viele Elemente der Programmiersprache C, so zum Beispiel Strukturen und Zeiger auf Strukturen, und Sie können viele Funktionen zu Eingabe, Ausgabe und Dateiverwaltung anwenden.

Auftrag:

Schreiben Sie ein Programm zur Adressverwaltung. Über die Konsole können Sie neue Adressen eingeben, die Adressliste ausgeben und nach verschiedenen Kriterien sortieren. Erweitern Sie das Ihre Adressverwaltung, sodass die Adressliste in eine Datei abgespeichert und wieder gelesen werden kann.

main.c

```
/**============================================================================================================
Hochschule Luzern - Technik & Architektur                                    Mikrocontroller - HS2012
Copyright 2012 Felix Rohrer                                                      http://hslu.ximit.ch
------------------------------------------------------------------------------------------------------------
Project     : Adressverwaltung
Name        : main.c
Author      : Felix Rohrer <felix.rohrer@stud.hslu.ch>
Version     : 2012-10-19v1.0
Description : Adressverwaltung
              Schreiben Sie ein Programm zur Adressverwaltung. Über die Konsole können Sie neue
              Adressen eingeben, die Adressliste ausgeben und nach verschiedenen Kriterien
              sortieren. Erweitern Sie das Ihre Adressverwaltung, sodass die Adressliste in eine
              Datei abgespeichert und wieder gelesen werden kann.
============================================================================================================*/
#include <stdlib.h>
#include "address.h"
#include "addressConsoleIO.h"

/*
 * The main procedure to run the application.
 * @param argc Number of command line arguments
 * @param argv The forwarded command line arguments
 * @return Application return (error) code
 */
int main(int argc, char** argv) {
    err_t errCode = 0; // errorcode handler

    // init AddressList
    errCode = initialize();
    if (errCode > 0) {
        //printf("Init: Successfully...\n");
        showMenu(TRUE);
    } else {
        //printf("Error: Init AddrList: (Err: %d)\n", errCode);
        return (EXIT_FAILURE);
    }

    return (EXIT_SUCCESS);
}
```

address.h

```
/**============================================================================================================
Hochschule Luzern - Technik & Architektur                                        Mikrocontroller - HS2012
Copyright 2012 Felix Rohrer                                                          http://hslu.ximit.ch
------------------------------------------------------------------------------------------------------------
Project     : Adressverwaltung
Name        : address.h
Author      : Felix Rohrer <felix.rohrer@stud.hslu.ch>
Version     : 2012-10-19v1.0
Description : Adressverwaltung
============================================================================================================*/
#ifndef ADDRESS_H
#define ADDRESS_H

#define PATH_SIZE 255 // File Path Buffer Size
#define TRUE      1   // Boolean True
#define FALSE     0   // Boolean False

/**
 * Custom Error Return Code
 */
typedef enum errorCode {
    SUCCESS           = 1,   // OK
    UNKNOW            = 0,   // undef status
    ERROR             = -1,  // general error
    NOMEMORY          = -2,  // alloc, realloc failed
    FILECREATEERROR   = -11, // fopen failed
    FILECLOSEERROR    = -12, // fclose failed
    FILENOTFOUNDERROR = -13, // file not found
    FILEREADERROR     = -14  // file read failed
} err_t;

/**
 * Defines address struct
 */
#define MAXSTRLEN 45 // Max String Len
typedef struct address_ {
    char firstname[MAXSTRLEN+1];
    char lastname[MAXSTRLEN+1];
    char street[MAXSTRLEN+1];
    unsigned int zip;
    char city[MAXSTRLEN+1];
    unsigned int index;
} address_t, address_p;

/**
 * Defines address List
 */
typedef struct addressList_ {
    address_t* addr;
    int size;
} addressList_t, addressList_p;

/**
 * Init AddressList (Allocate Memory)
 * @return Success / Error Status Code
 */
err_t initialize(void);

/**
 * List the current Address List (with CallBack Function for Output)
 * @param showAddrEntryFN CallBack Function for Output
 */
void listAddress(void (*showAddrEntryFN)(int, address_t*));

/**
 * Add a new Address to the List (Values)
 * @param lastname Lastname of the new entry
 * @param firstname Firstname of the new entry
 * @param street Street of the new entry
 * @param zip Zip of the new entry
 * @param city City of the new entry
 * @return Success / Error Status Code
 */
err_t addAddressValues(char* lastname, char* firstname, char* street, int zip, char* city);

/**
 * Add a new Address to the list (Address Object Pointer)
 * @param newAddress Pointer to the new Address
 * @return Success / Error Status Code
```

```
 */
err_t addAddress(address_p newAddress);

/**
 * Clear Address List in the Memory (free & re-malloc)
 * @return Success / Error Status Code
 */
err_t clearAddress(void);

/**
 * Save the current AddressList to a File
 * @param path Path to File for save
 * @param num Number or Addresses Loaded
 * @return Success / Error Status Code
 */
err_t saveAddrFile(char* path, int* num);

/**
 * Load Addresses from a File and them to the current AddressList
 * @param path Path to File for reading
 * @param num Number or Addresses Loaded
 * @return Success / Error Status Code
 */
err_t readAddrFile(char* path, int* num);

/**
 * Defines SortMode
 */
typedef enum { NAME=1, STREET, ZIP, CITY} sortmode;  // SortMode

/**
 * Sort Address by Sortmode with a modified (stable!) QuickSort
 * @param mode Sortmode, i.e. NAME, STREET, ZIP or CITY
 * @return Success / Error Status Code
 */
err_t sortAddress(sortmode mode);

/**
 * Compare Function pointer
 */
typedef int (*compfn)(const void *a, const void *b);

/**
 * Comparator by Name
 * @param a Address 1
 * @param b Address 2
 * @return Compare Result, >0 if Addr2 is bigger than Addr1
 */
int compareByName(const void *a, const void *b);

/**
 * Comparator by Street
 * @param a Address 1
 * @param b Address 2
 * @return Compare Result, >0 if Addr2 is bigger than Addr1
 */
int compareByStreet(const void *a, const void *b);

/**
 * Comparator by Zip
 * @param a Address 1
 * @param b Address 2
 * @return Compare Result, >0 if Addr2 is bigger than Addr1
 */
int compareByZip(const void *a, const void *b);

/**
 * Comparator by City
 * @param a Address 1
 * @param b Address 2
 * @return Compare Result, >0 if Addr2 is bigger than Addr1
 */
int compareByCity(const void *a, const void *b);

#endif  /* ADDRESS_H */
```

address.c

```
/**============================================================================================================
Hochschule Luzern - Technik & Architektur                                    Mikrocontroller - HS2012
Copyright 2012 Felix Rohrer                                                     http://hslu.ximit.ch
------------------------------------------------------------------------------------------------------------
Project     : Adressverwaltung
Name        : address.c
Author      : Felix Rohrer <felix.rohrer@stud.hslu.ch>
Version     : 2012-10-19v1.0
Description : Adressverwaltung
============================================================================================================*/
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include "address.h"
#include "addressFileIO.h"

// Global VAR: Addresslist
addressList_t* addrList = NULL;

err_t initialize(void)
{
    //create a new List
    addrList = (addressList_t*) malloc(sizeof(addressList_t));
    if (addrList) { // allocate Memory successfully?
        addrList->addr = NULL;
        addrList->size = 0;
        return SUCCESS;
    } else {
        return NOMEMORY;
    };
}

void listAddress(void (*showAddrEntryFN)(int, address_t*))
{
    int i; // tmp counter

    for (i=0; i<addrList->size; i++) {
        showAddrEntryFN(i+1, &(addrList->addr[i])); // Use CallBack Function for Output
    }
}

err_t addAddressValues(char* lastname, char* firstname, char* street, int zip, char* city)
{
    address_p newAddr; // Address Object to fill and add

    strcpy(newAddr.firstname, firstname);
    strcpy(newAddr.lastname, lastname);
    strcpy(newAddr.street, street);
    newAddr.zip = zip;
    strcpy(newAddr.city, city);
    return addAddress(newAddr);
}

err_t addAddress(address_p newAddress)
{
    int count; // Address Counter

    count = addrList->size;
    if (count == 0) {
        // alloc new memory
        addrList->addr = (address_t*) malloc(sizeof(address_t));
    } else {
        // realloc memory
        addrList->addr = (address_t*) realloc(addrList->addr, sizeof(address_t) * (count + 1));
    }
    if (addrList->addr) {
        // Add new Node
        strcpy(addrList->addr[count].firstname, newAddress.firstname);
        strcpy(addrList->addr[count].lastname, newAddress.lastname);
        strcpy(addrList->addr[count].street, newAddress.street);
        addrList->addr[count].zip = newAddress.zip;
        strcpy(addrList->addr[count].city, newAddress.city);
        addrList->addr[count].index = count;
        addrList->size = ++count;
        return SUCCESS;
    } else {
        return NOMEMORY;
    }
```

```
}

err_t clearAddress(void)
{
    // release memory
    free(addrList);
    // alloc new memory
    return initialize();
}

err_t saveAddrFile(char* path, int* num)
{
    return writeFile(path, addrList, formatCSVTab, num);
}

err_t readAddrFile(char* path, int* num)
{
    return readFile(path, addrList, lineParser, num);
}

err_t sortAddress(sortmode mode)
{
    int i; // temp counter, used for stable sort

    // qsort(Beginning address of array, Number of elements, Size of each element, Pointer to compare
function
    switch (mode) {
        case NAME:
            qsort(addrList->addr,addrList->size, sizeof(address_t), (compfn)compareByName);
            break;
        case STREET:
            qsort(addrList->addr,addrList->size, sizeof(address_t), (compfn)compareByStreet);
            break;
        case ZIP:
            qsort(addrList->addr,addrList->size, sizeof(address_t), (compfn)compareByZip);
            break;
        case CITY:
            qsort(addrList->addr,addrList->size, sizeof(address_t), (compfn)compareByCity);
            break;
    }
    // update index - used for stable sort
    for (i=0; i<addrList->size; i++) {
        addrList->addr[i].index = i;
    }
    return SUCCESS;
}

int compareByName(const void *a, const void *b)
{
    int res;                            // errorcode handler
    char name1[2*MAXSTRLEN + 1];        // Buffer for address 1 Name
    char name2[2*MAXSTRLEN + 1];        // Buffer for address 2 Name
    address_t *addr1 = (address_t*) a; // address 1, cast param
    address_t *addr2 = (address_t*) b; // address 2, cast param

    // Copy Lastname and Firstname into one String
    strcpy(name1, addr1->lastname);
    strcat(name1, addr1->firstname);
    strcpy(name2, addr2->lastname);
    strcat(name2, addr2->firstname);

    // Compare String Value 'Lastname Firstname'
    res = strcasecmp(name1, name2); // Compare String (ignore case)
    if (res == 0) { // stable sort
        res = addr1->index - addr2->index;
    }
    return res;
}

int compareByStreet(const void *a, const void *b)
{
    int res;                            // errorcode handler
    address_t *addr1 = (address_t*) a; // address 1, cast param
    address_t *addr2 = (address_t*) b; // address 2, cast param

    // Compare String Value 'Street'
    res = strcasecmp(addr1->street, addr2->street); // ignore case
    if (res == 0) { // stable sort
        res = addr1->index - addr2->index;
    }
```

```
    return res;
}

int compareByZip(const void *a, const void *b)
{
    int res;                            // errorcode handler
    address_t *addr1 = (address_t*) a; // address 1, cast param
    address_t *addr2 = (address_t*) b; // address 2, cast param

    // Compare Int value 'Zip'
    res = addr1->zip - addr2->zip;
    if (res == 0) { // stable sort
        res = addr1->index - addr2->index;
    }
    return res;
}

int compareByCity(const void *a, const void *b)
{
    int res;                            // errorcode handler
    address_t *addr1 = (address_t*) a; // address 1, cast param
    address_t *addr2 = (address_t*) b; // address 2, cast param

    // Compare String Value 'City'
    res = strcasecmp(addr1->city, addr2->city);
    if (res == 0) { // stable sort
        res = addr1->index - addr2->index;
    }
    return res;
}
```

addressConsoleIO.h

```
/**=============================================================================================
Hochschule Luzern - Technik & Architektur                                  Mikrocontroller - HS2012
Copyright 2012 Felix Rohrer                                                    http://hslu.ximit.ch
-----------------------------------------------------------------------------------------------
Project     : Adressverwaltung
Name        : addressConsoleIO.h
Author      : Felix Rohrer <felix.rohrer@stud.hslu.ch>
Version     : 2012-10-19v1.0
Description : Adressverwaltung
              Adressen aus der Konsole einlesen, bestehende ausgeben
=============================================================================================*/
#ifndef ADDRESSCONSOLEIO_H
#define ADDRESSCONSOLEIO_H

/**
 * Main Menu
 * @param showMainMenu Use Large or Small Menu
 */
void showMenu(int showMainMenu);

/**
 * Show Main Menu Desc: Large
 */
void showMainMenuLarge(void);

/**
 * Show Main Menu Desc: Small
 */
void showMainMenuSmall(void);

/**
 * GUI Add a new Address
 */
void addAddressGUI(void);

/**
 * GUI Save the current AddressList to a File
 */
void saveAddrFileGUI(void);

/**
 * GUI Load Addresses from a File and them to the current AddressList
 */
void readAddrFileGUI(void);

/**
 * GUI Sort Address by Sortmode with a modified (stable!) QuickSort
 * @param mode Sortmode, i.e. NAME, STREET, ZIP or CITY
 */
void sortAddressGUI(sortmode mode);

/**
 * GUI Clear Address List in the Memory (free & re-malloc)
 */
void clearAddressGUI(void);

/**
 * GUI Show one Address (CallBack Function)
 * @param count Index #Number
 * @param addr Addres to display
 */
void displayAddr(int count, address_t* addr);

/**
 * Show the current Address List
 */
void listAddressGUI(void);

#endif  /* ADDRESSCONSOLEIO_H */
```

addressConsoleIO.c

```
/**=============================================================================================
Hochschule Luzern - Technik & Architektur                                  Mikrocontroller - HS2012
Copyright 2012 Felix Rohrer                                                   http://hslu.ximit.ch
-----------------------------------------------------------------------------------------------
Project     : Adressverwaltung
Name        : addressConsoleIO.c
Author      : Felix Rohrer <felix.rohrer@stud.hslu.ch>
Version     : 2012-10-19v1.0
Description : Adressverwaltung
              Adressen aus der Konsole einlesen, bestehende ausgeben
=============================================================================================*/
#include <stdio.h>
#include <string.h>
#include "address.h"
#include "addressConsoleIO.h"

void showMenu(int displargeMenu)
{
    char menu = ' '; // buffer for key, user menu choice
    int listAfterSort = TRUE;

    do // Menu (until Q pressed)
    {
        if (displargeMenu) {
            // show Main Menu Desc (large)
            showMainMenuLarge();
            displargeMenu = FALSE;
        } else {
            // show Main Menu Desc (small)
            showMainMenuSmall();
        }
        printf(" » Please choose: ");

        while (!isalnum(menu = getchar())); // Solange einlesen bis Zahl oder Buchstabe
        menu = toupper(menu);

        switch(menu){
        case 'N': // Insert a new address
            addAddressGUI();
            break;
        case 'L': // List stored addresses
            listAddressGUI();
            break;
        case 'C': // Clear stored addresses
            clearAddressGUI();
            break;
        case 'R': // Read addresses from file
            readAddrFileGUI();
            break;
        case 'S': // Save addresses to file
            saveAddrFileGUI();
            break;
        case 'M': // Show Main Menu
            displargeMenu = TRUE;
            break;
        case '1': // Sort addresses by name & List them
            sortAddressGUI(NAME);
            break;
        case '2': // Sort addresses by street & List them
            sortAddressGUI(STREET);
            break;
        case '3': // Sort addresses by zip & List them
            sortAddressGUI(ZIP);
            break;
        case '4': // Sort addresses by city & List them
            sortAddressGUI(CITY);
            break;
        }
    } while(menu != 'Q');
}
```

```
void showMainMenuLarge(void)
{
        printf("\n");
        printf("╔═══════════════════════════════════════════════════════════════════════════════╗\n");
        printf("║                                                                               ║\n");
        printf("║       _       _   _                  __  __                                   ╔\n");
        printf("║      /_\\  __| |__| |_ _ ___ ______  |  \\/  |___ _ _  _  __ _ __ _ ___ _ _    ╔\n");
        printf("║     / _ \\/ _` / _` | '_/ -_|_-<_-< | |\\/| / _ \\ ' \\ V / _` / _` / -_) '_|   ╔\n");
        printf("║    /_/ \\_\\__,_\\__,_|_| \\___/__/__/ |_|  |_\\___/_||_\\_/\\__,_\\__, \\___|_|     ╔\n");
        printf("║                                                              |___/            ╔\n");
        printf("╠═══════════════════════════════════════════════════════════════════════════════╣\n");
        printf("╠═══════════════════════════════════════════════════════════════════════════════╣\n");
        printf("║   Select an option                                                            ╔\n");
        printf("║     N : Insert a new address         1 : Sort addresses by name               ╔\n");
        printf("║     L : List stored addresses        2 : Sort addresses by street             ╔\n");
        printf("║     C : Clear stored addresses       3 : Sort addresses by zip                ╔\n");
        printf("║                                      4 : Sort addresses by city               ╔\n");
        printf("║     R : Read addresses from file                                              ╔\n");
        printf("║     S : Save addresses to file       Q : Quit           © 2012 Felix Rohrer   ╔\n");
        printf("╚═══════════════════════════════════════════════════════════════════════════════╝\n");
}

void showMainMenuSmall(void)
{
        printf("\n");
        printf("╔═══════════════════════════════════════════════════════════════════════════════╗\n");
        printf("║ N: New │ L: List │ C: Clear │ R: ReadFile │ S: SaveFile │ M: Menu │ Q: Quit   ╔\n");
        printf("╠═══════════════════════════════════════════════════════════════════════════════╣\n");
        printf("║ 1: sort by name │ 2: sort by street │ 3: sort by zip │ 4: sort by city         ╔\n");
        printf("╚═══════════════════════════════════════════════════════════════════════════════╝\n");
}

void addAddressGUI(void)
{
    int res;                    // errorcode handler
    char firstname[MAXSTRLEN+1]; // Buffer for Firstname
    char lastname[MAXSTRLEN+1];  // Buffer for Lastname
    char street[MAXSTRLEN+1-5];  // Buffer for Street
    char streetNbr[5+1];         // Buffer for Street Number
    unsigned int zip;            // Buffer for Zip
    char city[MAXSTRLEN+1];      // Buffer for City
    size_t length;               // Lenght of Street, combine Street + StreetNbr

    printf("\n\n┌──────────────────────────────────────────────────────────────────────────────┐\n");
    printf("│ Add new address                                                              │\n");
    printf("├──────────────────────────────────────────────────────────────────────────────┘\n");
    printf("│ Firstname:  ");
    scanf("%s", firstname);
    printf("│ Lastname:   ");
    scanf("%s", lastname);
    printf("│ Street:     ");
    scanf("%s", street);
    printf("│ Street Nbr: ");
    scanf("%s", streetNbr);
    printf("│ Zip:        ");
    scanf("%i", &zip);
    printf("│ City:       ");
    scanf("%s", city);
    printf("├──────────────────────────────────────────────────────────────────────────────\n");

    // combine street & streetnumber
    length = strlen(street);
    street[length] = ' ';
    street[length+1] = '\0';
    strcat(street, streetNbr);
    // add new addr.
    res = addAddressValues(lastname, firstname, street, zip, city);
    if (res != SUCCESS) {
        printf("│ ERROR: Could not add new address! (Err: %i)\n", res);
    } else {
        printf("│ New address added successfully.\n");
    }
    printf("└──────────────────────────────────────────────────────────────────────────────\n");
}

void saveAddrFileGUI(void)
{
    err_t res;            // errorcode handler
    char path[PATH_SIZE]; // filepath, filename buffer
    int count = 0;        // counter of addresses

    printf("\n\n┌──────────────────────────────────────────────────────────────────────────────\n");
```

```
    printf("| Save to file                                                                 |\n");
    printf("|------------------------------------------------------------------------------|\n");
    printf("| Please enter filename: ");
    scanf("%s", path);
    res = saveAddrFile(path, &count); // Save AddressList to a File
    if (res != SUCCESS) {
        printf("| ERROR: Could not write file! (Err: %i)\n", res);
    } else {
        printf("| Save done. %i addresses saved into file '%s'\n", count, path);
    }
    printf("└──────────────────────────────────────────────────────────────────────────────\n");
}

void readAddrFileGUI(void)
{
    err_t res;            // errorcode handler
    char path[PATH_SIZE]; // filepath, filename buffer
    int count = 0;        // counter of addresses

    printf("\n\n┌──────────────────────────────────────────────────────────────────────────────\n");
    printf("| Load from file                                                               |\n");
    printf("|------------------------------------------------------------------------------|\n");
    printf("| Please enter filename: ");
    scanf("%s", path);
    res = readAddrFile(path, &count); // Read Addresses from a File and add them to the current AddressList
    if (res != SUCCESS) {
        printf("| ERROR: Could not read file '%s'! (Err: %i)\n", path, res);
    } else {
        printf("| Load done. %i addresses loaded from file '%s'.\n", count, path);
    }
    printf("└──────────────────────────────────────────────────────────────────────────────\n");
}

void sortAddressGUI(sortmode mode)
{
    err_t res; // errorcode handler

    res = sortAddress(mode); // SortAddress by SortMode
    if (res == SUCCESS) {
        printf(" » Address List successfully sorted...\n");
    } else {
        printf(" » ERROR: Address List could not be sorted! (Err: %i)\n", res);
    }
}

void clearAddressGUI(void)
{
    err_t res; // errorcode handler

    res = clearAddress(); // free & re-malloc memory
    if (res == SUCCESS) {
        printf(" » Address List cleared...\n");
    } else {
        printf(" » ERROR: Address List could not be cleared! (Err: %i)\n", res);
    }
}

void displayAddr(int count, address_t* addr)
{
    char nameBuff[2*MAXSTRLEN+1]; // Buffer for Lastname + Firstname

    sprintf(nameBuff, "%s %s", addr->lastname, addr->firstname);
    printf("| %4i %-25.24s%-25.24s%-5d%.16s\n",
            count,
            nameBuff,
            addr->street,
            addr->zip,
            addr->city);
}

void listAddressGUI(void)
{
    printf("\n\n┌──────────────────────────────────────────────────────────────────────────────\n");
    printf("| Address List                                                                 |\n");
    printf("|------------------------------------------------------------------------------|\n");
    printf("| %4s %-25s%-25s%-5s%s\n", "#", "Name", "Street", "Zip", "City");
    listAddress(displayAddr); // ListAddress with CallBack Function displayAddr -> Console Stdout
    printf("└──────────────────────────────────────────────────────────────────────────────\n");
}
```

addressFileIO.h

```
/**=============================================================================================
Hochschule Luzern - Technik & Architektur                                   Mikrocontroller - HS2012
Copyright 2012 Felix Rohrer                                                    http://hslu.ximit.ch
-----------------------------------------------------------------------------------------------
Project     : Adressverwaltung
Name        : addressFileIO.h
Author      : Felix Rohrer <felix.rohrer@stud.hslu.ch>
Version     : 2012-10-19v1.0
Description : Adressverwaltung
              Adressen in eine Datei (csv) speichern oder aus einer Datei importieren.
=============================================================================================*/
#ifndef ADDRESSFILEIO_H
#define ADDRESSFILEIO_H

/**
 * Parse the line to get the Address Details
 * @param line Addres Line from CSV Export
 * @return Success / Error Status Code
 */
err_t lineParser(char* line);

/**
 * Export AddressList to a CSV_File
 * @param path Filepath, Filename for the csv-File
 * @param addrList Pointer to the current AddressList
 * @param formatCSVfn CallBack-Function to format the Output
 * @param num Return value how many Addresses are exporet
 * @return Success / Error Status Code
 */
err_t writeFile(char* path, addressList_t* addrList, err_t (*formatCSVfn)(address_t*, char*), int* num);

/**
 * Import AddressList from a CSV-File
 * @param path Filepath, Filename for the csv-File
 * @param addrList Pointer to the current AddressList
 * @param lineParserFn CallBack-Function for the LineParser (Format)
 * @param num Returnh value how many Addresses are imported
 * @return Success / Error Status Code
 */
err_t readFile(char* path, addressList_t* addrList, err_t (*lineParserFn)(char*), int* num);

/**
 * CallBack-Function to Format the Export: CSV by Tab
 * @param addr Pointer to one Address
 * @param line Formated Output String
 * @return Success / Error Status Code
 */
err_t formatCSVTab(address_t* addr, char* line);

/**
 * CallBack-Function to Format the Export: CSV by Semicolon
 * @param addr Pointer to one Address
 * @param line Formated Output String
 * @return Success / Error Status Code
 */
err_t formatCSVSemicolon(address_t* addr, char* line);

#endif  /* ADDRESSFILEIO_H */
```

addressFileIO.c

```
/**=============================================================================================
Hochschule Luzern - Technik & Architektur                                  Mikrocontroller - HS2012
Copyright 2012 Felix Rohrer                                                    http://hslu.ximit.ch
-----------------------------------------------------------------------------------------------
Project     : Adressverwaltung
Name        : addressFileIO.c
Author      : Felix Rohrer <felix.rohrer@stud.hslu.ch>
Version     : 2012-10-19v1.0
Description : Adressverwaltung
              Adressen in eine Datei (csv) speichern oder aus einer Datei importieren.
=============================================================================================*/
#include <stdio.h>
#include <string.h>
#include "address.h"
#include "addressFileIO.h"

#define BUFFER_SIZE 2048 // File Line Buffer Size

err_t lineParser(char* txtLine)
{
    char lastname[MAXSTRLEN+1];  // buffer for lastname
    char firstname[MAXSTRLEN+1]; // buffer for firstname
    char street[MAXSTRLEN+1];    // buffer for street
    int zip;                     // buffer for zip
    char city[MAXSTRLEN+1];      // buffer for city
    char* txtPart;               // buffer for text token

    // init, set default to empty
    strcpy(firstname, "\0");
    strcpy(lastname, "\0");
    strcpy(street, "\0");
    zip = -1;
    strcpy(city, "\0");

    // Reads all address tokens out of txtLine
    // lastname
    txtPart = strtok(txtLine, "\t"); // only first read needs the txtLine
    if (txtPart != NULL && (strlen(txtPart)<MAXSTRLEN)) {
        strcpy(lastname, txtPart);
    } else { return FILEREADERROR; }

    // firstname
    txtPart = strtok(NULL, "\t");
    if (txtPart != NULL && (strlen(txtPart)<MAXSTRLEN)) {
        strcpy(firstname, txtPart);
    } else { return FILEREADERROR; }

    // street
    txtPart = strtok(NULL, "\t");
    if (txtPart != NULL && (strlen(txtPart)<MAXSTRLEN)) {
        strcpy(street, txtPart);
    } else { return FILEREADERROR; }

    // zip
    txtPart = strtok(NULL, "\t");
    if (txtPart != NULL && (strlen(txtPart)<MAXSTRLEN)) {
        zip = atoi(txtPart);
    } else { return FILEREADERROR; }

    // city
    txtPart = strtok(NULL, "\r"); // \r to remove trailing cr / newline
    if (txtPart != NULL && (strlen(txtPart)<MAXSTRLEN)) {
        strcpy(city, txtPart);
    } else { return FILEREADERROR; }

    // add new Address
    return addAddressValues(lastname, firstname, street, zip, city);
}

err_t writeFile(char* path, addressList_t* addrList, err_t (*formatCSVfn)(address_t*, char*), int* num)
{
    int res;                // errorcode handler
    int count;              // counter of addresses
    int i;                  // tmp counter
    FILE *file = NULL;      // file handler
    char line[BUFFER_SIZE]; // buffer for one line

    // create new file (overwrite existing one)
    file = fopen(path, "w+");
```

```
    if (file == NULL) {
        return FILECREATEERROR;
    }
    // write file
    count = addrList->size;
    *num = count; // save number of addresses for output
    for (i=0; i<count; i++) {
        res = (*formatCSVfn)(&(addrList->addr[i]), line); // Use CallBack-Func. to Format the Export
        if (res != SUCCESS || fputs(line, file) == EOF) {
            free(line);
            fclose(file);
            return res;
        }
    }
    // close the file, define return status
    if (fclose(file) != 0) {
        res = FILECLOSEERROR;
    } else {
        res = SUCCESS;
    }
    return res;
}

err_t readFile(char* path, addressList_t* addrList, err_t (*lineParserFn)(char*), int* num)
{
    err_t res;              // errorcode handler
    FILE *file = NULL;      // file handler
    char line[BUFFER_SIZE]; // buffer file reading
    int count = 0;          // Counter for addresses

    // Open the file (read-only)
    file = fopen(path, "r");
    if (file == NULL) {
        return FILENOTFOUNDERROR;
    }

    // Reads until no more lines are available in the csv
    while(fgets(line, sizeof(line), file) != NULL) {
        count++;
        // try to parse and process the file (Use CallBack-Func.)
        res = (*lineParserFn)(line);
        // in case of error, close the file and abort with error code
        if (res != SUCCESS) {
            fclose(file);
            file = NULL;
            return res;
        }
    }

    // close the file
    if (fclose(file) != 0) {
        res = FILECLOSEERROR;
    } else {
        res = SUCCESS;
    }
    *num = count; // save back counter value
    return res;
}

err_t formatCSVTab(address_t* addr, char* line){
    // format string for csv export
    sprintf(line, "%s\t%s\t%s\t%i\t%s\r\n",
            addr->lastname,
            addr->firstname,
            addr->street,
            addr->zip,
            addr->city);
    return SUCCESS;
}

err_t formatCSVSemicolon(address_t* addr, char* line){
    // format string for csv export
    sprintf(line, "%s;%s;%s;%i;%s\r\n",
            addr->lastname,
            addr->firstname,
            addr->street,
            addr->zip,
            addr->city);
    return SUCCESS;
}
```