Übung 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);
}
```

Übung 3.2 - Entwurf einer Übung zum Thema 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);
}
```

Übung 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);
```

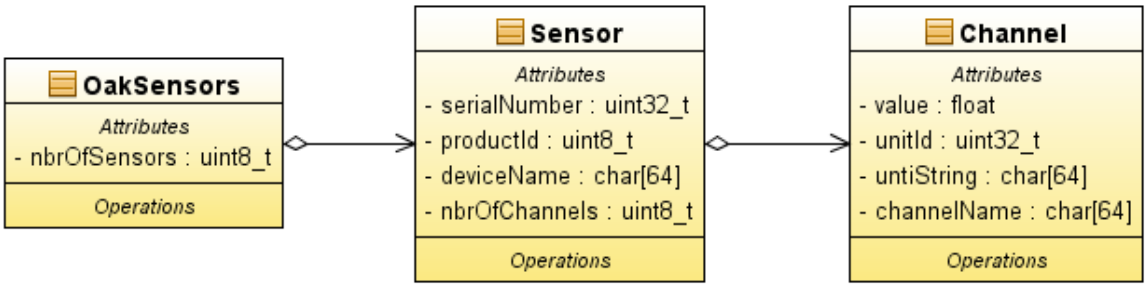
```c
/**
 * The main procedure to run the application.
 * @param argc Number of command line arguments
 * @param argv The forwarded command line arguments
 * @return Application return (error) code
 */
int main(int argc, char** argv)
{
        struct tnode *root;
        char word[MAXWORD];
        root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

Übung 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;
}
```