PROGRAMMIEREN II

Mechatronik an der HAW

NetBeans

[↑] + [Alt] + [F]	aufräumen
[Strg] + [LEERTASTE]	Autovervollständigung

Ext. Konsole	RM auf Project -> Properties -> Run -> Ext.Terminal

GIT

Repository anlagen

```
git init
```

Zentrales (push-able) Reository anlagen

```
git init --bare
```

Repository als Clone von einem lokalem Repository kopieren

```
git clone –l "c:\temp\Git\repositories\Prog"
```

Repository abrufen

```
git clone git@bitbucket.org:Leo_M/pr2_2.git /c/temp/PRP2_2/gitProject/
```

Grundsätzliche Einstellungen

```
git config --global user.name "Hans_Wurst"
```

```
git config --global user.email hans.wurst@mail.de
```

Auflisten der Einstellungen

```
git config --list
```

Hilfe

```
git help
```

Statusinformationen

```
git status
```

Datei hinzufügen (staging)

```
git add <FILE>      oder            git add .
```

Änderungen mit Nachricht als Version zusammenfassen

```
git commit (-a) –m „Version …"
```

Aktuelle Version der Branches master an das Repository übertragen

```
git push origin master
```

```
Änderungen aus dem Repository holen (ohne merge)
```

```
git fetch origin
```

Unterschiede ermitteln zwischen zwei Branches hier master und dem master der lokalen Kopie

```
git diff master origin/master
```

Angegebenen Branch in den aktuellen mergen

```
git merge master/origin
```

Änderung aus dem Repository holen (mit merge)

```
git pull origin
```

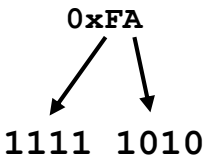
Bit-Operationen

~	NICHT (bitweise)
&	UND
|	ODER
^	XOR
>>	Shift nach rechts
<<	Shift nach links

```
#define LED 0x20
#define MOTOR 0x08

x = 0x28;
x = 0x20 | 0x08;
x = (1<<5) | (1<<3);
x = LED | MOTOR
```

(1 << 4) ⇔ **0x10** ⇔ **0001 0000**



Bit-Manipulation mittels Bitmasken

Bit-Setzen

OR -> 1 in Maske setzt das Bit

Bit löschen

AND -> 0 in Maske löscht das Bit

Bit invertieren

XOR -> 1 in Maske invertiert das Bit

Flankenauswertung

```
pfSPI = (( SPI ^ previousSPI) & SPI );
nfSPI = (( SPI ^ previousSPI ) & ~SPI );
```

Module

`extern` Globale Variable erstellen

`static` schränkt Scope ein und als „MERKER“ (Deklarierung nur beim 1. Durchlauf)

Layering

Funktionen/Module sollen klare Schnittstellen haben. Keine Ebenen überspringen, da sonst die komplexität steigt!

enum

```
enum fsmStates {Idle, Zustand1, Zustand2, Error, Finite};

static enum fsmStates state = Idle;
```

enum-Übergabe Beispiel

```
void fsm(enum coins coinValue) {

if (coinValue … );

}
```

Dateibearbeitung

Aktuelle Position innerhalb der Datei bestimmen

```
long int ftell (FILE * stream );
```

Aktuelle Position innerhalb der Datei neu setzen

```
int fseek ( FILE * stream, long int offset, int origin );
```

SEEK_SET	Dateianfang
SEEK_CUR	Aktuelle Pos.
SEEK_END	Dateiende

Auslesen einzelner Zeichen

```
int fgetc ( FILE * stream );
```

Anzahl von Bytes in Speicher einlesen (Datei -> Speicher)

```
size_t fread ( void * ptr,              <- char * buffer;
               size_t size,             <- Byte-Größe eines Elements
               size_t count,            <- Anzahl der Elemente
               FILE * stream );         <- Datei-Stream
```

Anzahl von Bytes aus dem Speicher schreiben (Speicher -> Datei)

```
size_t fwrite ( void * ptr,
                size_t size,
                size_t count,
                FILE * stream );
```

```c
/* fwrite example : write buffer */
#include <stdio.h>

int main() {
    FILE * pFile;
    char buffer[] = {'x', 'y', 'z'};
    pFile = fopen("c://temp//myfile.bin", "wb");
    fwrite(buffer, sizeof (char), sizeof (buffer), pFile);
    fclose(pFile);
    return 0;
}
```

```c
/* fread example: read an entire file */
#include <stdio.h>
#include <stdlib.h>

int main() {
    FILE * pFile;
    long lSize;
    char * buffer;
    size_t result;

    pFile = fopen("c://temp//myfile.bin", "rb");
    if (pFile == NULL) {
        fputs("File error", stderr);
        exit(1);
    }
    // Datei Größe ermitteln:
    fseek(pFile, 0, SEEK_END);
    lSize = ftell(pFile);
    rewind(pFile);
    // Speicher für die ganze Datei reservieren
    buffer = (char*) malloc(sizeof (char)*lSize);
    if (buffer == NULL) {
        fputs("Memory error", stderr);
        exit(2);
    }
    // Datei in den Puffer kopieren:
    result = fread(buffer, 1, lSize, pFile);
    if (result != lSize) {
        fputs("Reading error", stderr);
        exit(3);
    }
    /* Die ganze Datei wurde nun in den Speicher geladen. */
    // terminate
    fclose(pFile);
    free(buffer);
    return 0;
}
```

Dynamische Speicherreservierung

Das Betriebssystem verwaltet den Hauptspeicher eines Computers. Aus einem speziellen Bereich (Heap) kann sich ein Programm Speicherplatz ausleihen.

```
void * malloc ( size_t size );
```

Benutzung

- Benötigte Größe in Byte bestimmen
- Typ muss an gewünschten Typ angepasst werden (casting)
- Bereich muss spätestens am Programmende zurückgegeben werden

Hinweise

- bei Fehlschlag wird Null-Pointer (NULL) zurückgegeben. Dieser darf nicht benutzt werden!
- Der Speicherbereich wird nicht initialisiert

Bsp:

```
int* values = 0;
values = (int*) malloc(sizeof (int)*10);

if (values != NULL) {
    for (int i = 0; i < 10; i++) {
        values[i] = i*i
}
free(values);
}
```

Speicher wieder freigeben

```
void free ( void * ptr );
```

Speicher vergrößern und Inhalt beibehalten

```
void * realloc ( void * ptr, size_t size );
```

Speicherbereich mit Wert x-mal füllen

```
void * memset ( void * ptr, int value, size_t num );
```

Pointer

Referenz herstellen: Adress-Operator &

```
<TYPE> * <NAME> = 0           // Pointer initialisieren
<NAME> = &<VARIABLE>          // dem Pointer eine Adresse zuweisen
```

Zugriff auf Datenelemente: Dereferenzierungsoperator *

```
<VARIABLE> = *<POINTER>
```

Zugriff auf Elemente in Strukturen ->

```
a = ref->x;
```

Dynamische verkettete Listen

```
typedef struct tableRow {
    int ID;
    char* term;
    struct tableRow* next;
} tableRow_T;
```

main.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "table_test.h"
#include "table.h"

int main(int argc, char** argv) {
    test_table();
    FILE *myFileStream = 0;
    myFileStream = fopen("c:\\tmp\\words.txt", "r");
    if (myFileStream != 0) {
        int result = 0;
        int ID = 0;
        int rownumber = 0;
        char buffer[256];

        tableRow_T* head = 0;

        while (!feof(myFileStream)) {
            result = fscanf(myFileStream, "%i;%s\n", &ID, buffer);
            if (result == 2) {
                tableRow_T* node = 0;
                node = createNode(ID, buffer);
                append(&head, node);
                printf("Zeile %i: %i,%s\n", rownumber, ID, buffer);
            }
            rownumber++;
        }

        fclose(myFileStream);
    }
    return (EXIT_SUCCESS);
}
```

table.h

```c
typedef struct tableRow {
    int ID;
    char* term;
    struct tableRow* next;
} tableRow_T;

tableRow_T* createNode(int ID, char* term );
void append( tableRow_T** start, tableRow_T* node );
```

table.c

```c
#include "table.h"
#include <stdlib.h>
#include <string.h>

tableRow_T* createNode(int ID, char* term) {
    tableRow_T* node = 0;
    node = (tableRow_T*) malloc(sizeof (tableRow_T));
    if (node != 0) {
        node->next = 0;
        node->ID = ID;
        node->term = (char*) malloc(sizeof (char)*strlen(term));
        strcpy(node->term, term);
    }
    return node;
}

void append(tableRow_T** start, tableRow_T* node) {
    tableRow_T* iterator = *start;
    if (iterator == 0) {
        *start = node;
    } else {
        while (iterator->next != 0) {
            iterator = iterator->next;
        }
        iterator->next = node;
    }
}
```

table_test.h

```c
#include "table.h"
void test_table();
void test_createNode();
void test_append();
```

time()

```c
/* time example */
#include <stdio.h>      /* printf */
#include <time.h>       /* time_t, struct tm, difftime, time, mktime */

int main ()
{
  time_t timer;
  struct tm y2k;
  double seconds;

  y2k.tm_hour = 0;   y2k.tm_min = 0; y2k.tm_sec = 0;
  y2k.tm_year = 100; y2k.tm_mon = 0; y2k.tm_mday = 1;

  time(&timer);  /* get current time; same as: timer = time(NULL)  */
  seconds = difftime(timer,mktime(&y2k));
  printf ("%.f seconds since January 1, 2000 in the current timezone\n", seconds);

  // Ausgabe: 442082505 seconds since January 1, 2000 in the current timezone
  return 0;
}
```

ctime()

```c
/* ctime example */
#include <stdio.h>      /* printf */
#include <time.h>       /* time_t, time, ctime */

int main ()
{
  time_t rawtime;

  time (&rawtime);
  printf ("The current local time is: %s", ctime (&rawtime));

  // Ausgabe: The current local time is: Fri Jan 03 15:44:24 2014
  return 0;
}
```

localtime()

```c
/* localtime example */
#include <stdio.h>      /* puts, printf */
#include <time.h>       /* time_t, struct tm, time, localtime */

int main ()
{
  time_t rawtime;
  struct tm * timeinfo;

  time (&rawtime);
  timeinfo = localtime (&rawtime);
  printf ("Current local time and date: %s", asctime(timeinfo));

  return 0;
}
```