CSBas Final Exam - Notizen - Kaderli Severin

ASM

C

Compiling and linking

```
gcc -c main.c
gcc -o main main.o
```

IO

```
#inclde <stdio.h>
#define PI 3.1415
#define area(x, y) x * y

int main(int argc, char **argv) {
        int inputValue;
        printf("Please enter a number");
        scanf("%d", &inputValue);
        printf("Your value is %d and PI is %f", inputValue, PI);

        return 0;
}
```

Random Numbers

```
#include <time.h>
#include <stdlib.h>

srand(time(NULL));
// Creates random 0 <= r <= 19
int random = rand % 20;
```

Pointer

```
int* pv;
int v = 3;
pv = &v;

// Wert bzw. Adresse erhalten
int val = *(pv);
int *ptr = &(val);

// Function pointer
void printSq(int n, void (*fp)(int square)) {
        int sq = n * n;
        (*pf)(sq);
}
printSq(3, printA);
```

Arrays

```
int n = 5;
double* marks = (double*) malloc(n * sizeof(double));
double marks[n];

// String array
char* arr[n];
arr[0] = (char *) malloc(10 * sizeof(char));

// Speicher freigeben
free(marks);

// Speicher neu vergeben
realloc(marks, size_t);
```

Struct

```
// Struct als Typo definieren
typedef struct rec {
        int id;
        float PI;
} Record;

// Struct initialisieren
Record* structPtr;
structPtr = (Record*) malloc(sizeof(Record));

// Auf Struct Werte zugreifen
(*structPtr).id = 10;
structPtr -> id = 10;
```

Enum

```
enum month {JAN, FEB, MAR, APR, MAY, JUN, ...};
enum month rmonth;
rmonth = JUN;
```

Command Line Parameters

```
int main(int argc, char** argv) {
        for(int i = 0; i < argc; i++) {
                printf("argv[%d] %s\", i, argv[i]);
        }
}
```

File Handling

```
// Open file pointer
FILE* fp;
fp = fopen("file.txt", "a");

// Write to file
fprintf(fp, "this is a new line\n");

// Close file pointer
```

```
fclose(fp);
```

Threads

```
#include <pthread.h>

pthread_mutex_t mutex;

void* printOut1(void *ch) {
        pthread_mutex_lock(&mutex);
        printf("%c1\n", *(char*)ch)
        pthread_mutex_unlock(&mutex);
        return NULL;
}

void* printOut2(void *ch) {
        pthread_mutex_lock(&mutex);
        printf("%c2\n", *(char*)ch)
        pthread_mutex_unlock(&mutex);
        return NULL;
}

int main(int argc, char **argv) {
        pthread_t p1, p2;
        char ch1 = '1', ch2 = '2';

        // Initialise the mutex (Mutual Exclusion)
        pthread_mutex_init(&mutex, NULL);

        // Create the threads
        pthread_create(&p1, NULL, printOut, &ch1);
        pthread_create(&p2, NULL, printOut, &ch2);

        // Wait for threads to finish
        pthread_join(p1, NULL);
        pthread_join(p2, NULL);

        return 0;
}
```