Programmierung 2 - SS19

Projekt 2 – C-Stars

Autoren: Niklas Medinger, Jessica A. Schmidt, Kallistos Weis

08. Mai 2019

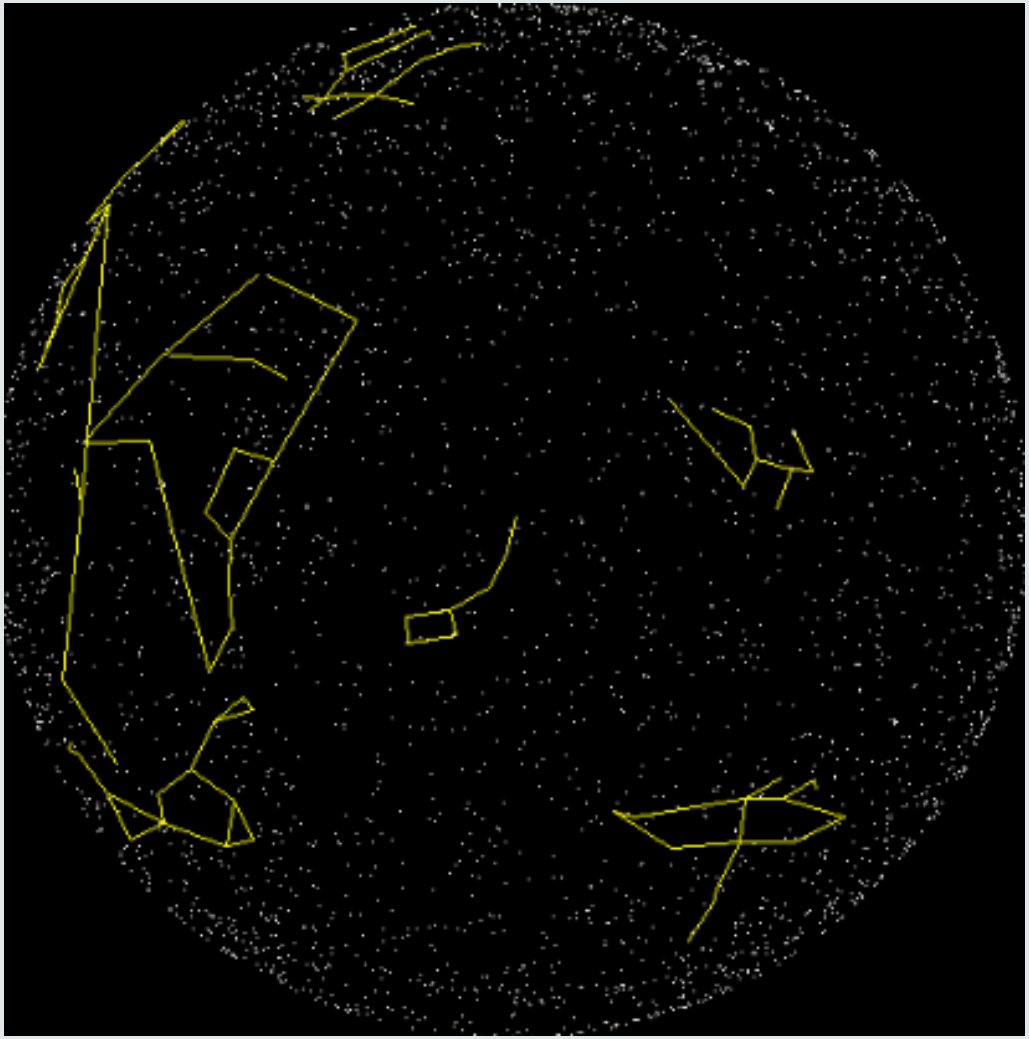
Universität des Saarlandes

Überblick

1. Einführung
2. Aufgaben
3. Testing
4. C — Ein Beispiel

Einführung

Sterne und Sternenbilder

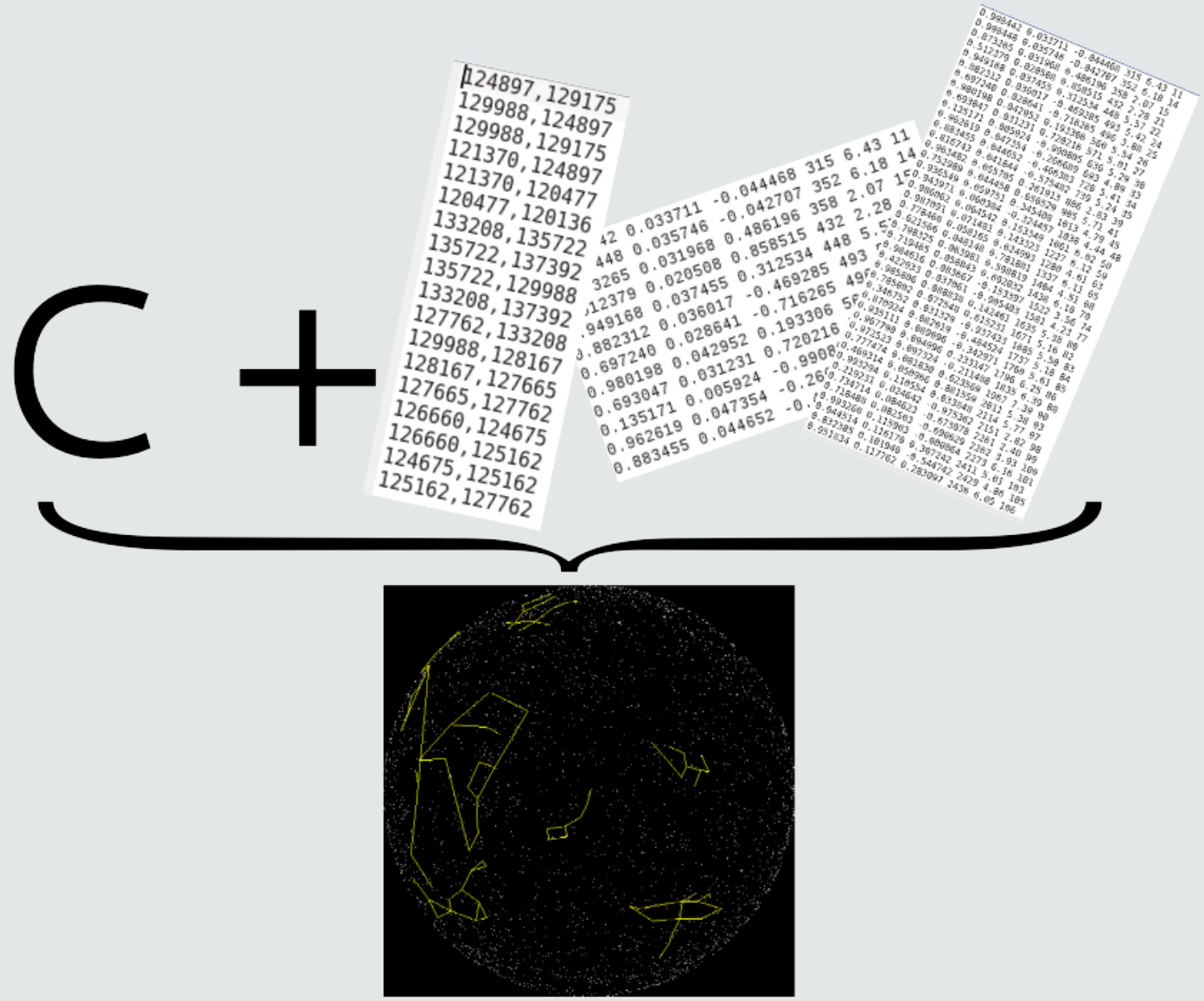


Sterne und Sternenbilder

```
0.998442 0.033711 -0.044468 315 6.43 11
0.998448 0.035746 -0.042707 352 6.18 14
0.873265 0.031968 0.486196 358 2.07 15
0.512379 0.020508 0.858515 432 2.28 21
0.949168 0.037455 0.312534 448 5.57 22
0.882312 0.036017 -0.469285 493 5.42 24
0.697240 0.028641 -0.716265 496 3.88 25
0.980198 0.042952 0.193306 560 5.54 26
0.693047 0.031231 0.720216 571 5.01 27
0.135171 0.005924 -0.990805 636 5.29 30
0.962619 0.047354 -0.266689 693 4.89 33
0.883455 0.044652 -0.466383 720 5.41 34
```

```
124897,129175
129988,124897
129988,129175
121370,124897
121370,120477
120477,120136
133208,135722
135722,137392
135722,129988
133208,137392
127762,133208
129988,128167
128167,127665
127665,127762
126660,124675
126660,125162
124675,125162
125162,127762
```

Eure Aufgabe:



Git Projekt-Repository

Wir können das Projekt mit `git clone` unter folgender URL beziehen:

```
https://prog2scm.cdl.uni-saarland.de/git/project2/$username
```

$username = Euer Benutzername auf der Prog2-Website

Git Projekt-Repository

Wir können das Projekt mit `git clone` unter folgender URL beziehen:

```
https://prog2scm.cdl.uni-saarland.de/git/project2/$username
```

$username = Euer Benutzername auf der Prog2-Website

Beispiel

```
git clone https://.../project2/s8konrad project2
```

Git Projekt-Repository

Wir können das Projekt mit `git clone` unter folgender URL beziehen:

```
https://prog2scm.cdl.uni-saarland.de/git/project2/$username
```

$username = Euer Benutzername auf der Prog2-Website

Beispiel

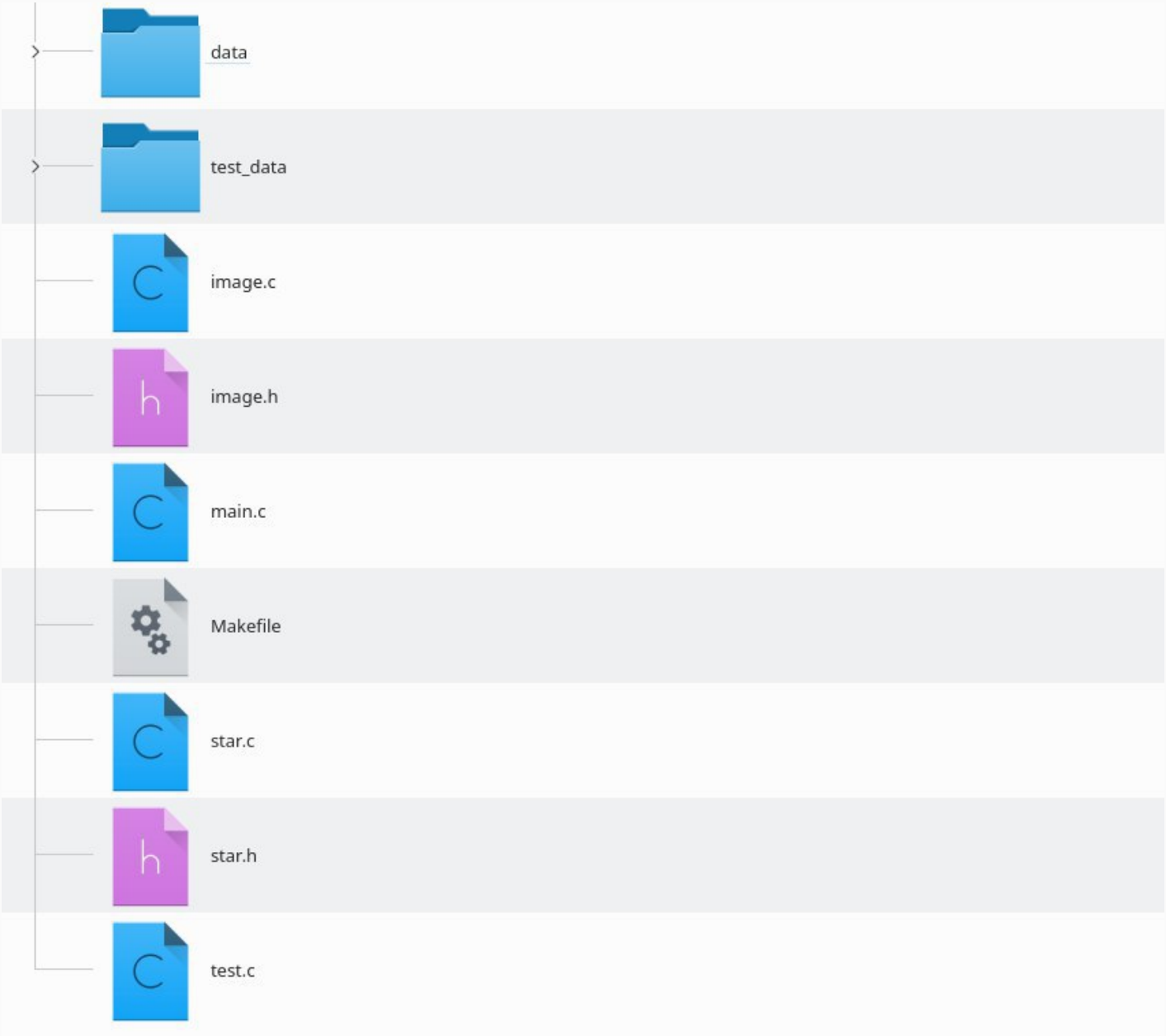
```
git clone https://.../project2/s8konrad project2
```

Achtung!

Die Repositories sind nur innerhalb des Uninetzes erreichbar. Von außerhalb kann man eine VPN-Verbindung zum Uninetz einrichten.

Eine Anleitung steht auf der Website unter Software.

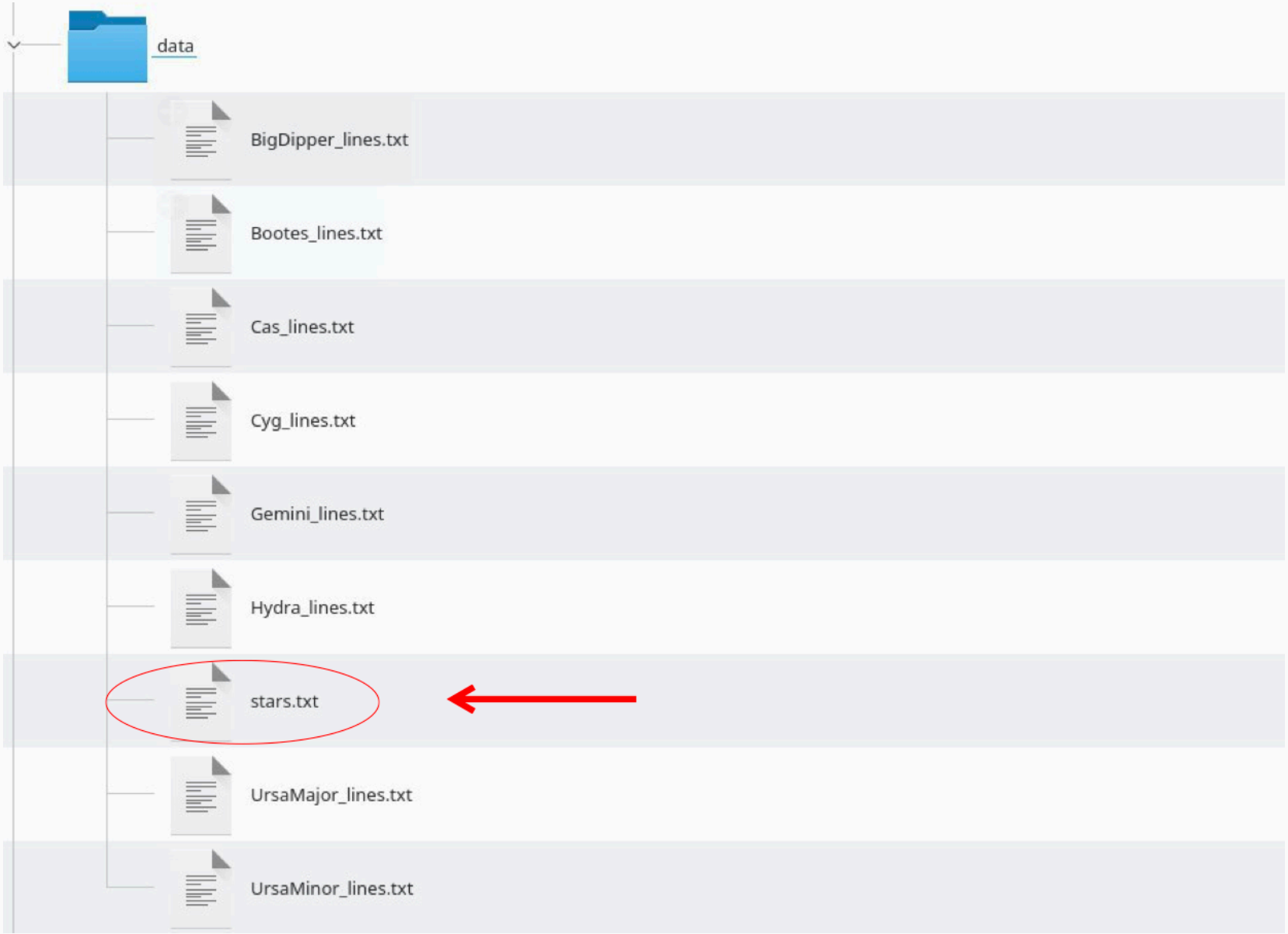
Verzeichnisstruktur



data


data
BigDipper_lines.txt
Bootes_lines.txt
Cas_lines.txt
Cyg_lines.txt
Gemini_lines.txt
Hydra_lines.txt
stars.txt
UrsaMajor_lines.txt
UrsaMinor_lines.txt

data


data
BigDipper_lines.txt
Bootes_lines.txt
Cas_lines.txt
Cyg_lines.txt
Gemini_lines.txt
Hydra_lines.txt
stars.txt
UrsaMajor_lines.txt
UrsaMinor_lines.txt

data


data
BigDipper_lines.txt
Bootes_lines.txt
Cas_lines.txt
Cyg_lines.txt
Gemini_lines.txt
Hydra_lines.txt
stars.txt
UrsaMajor_lines.txt
UrsaMinor_lines.txt

Verzeichnisstruktur

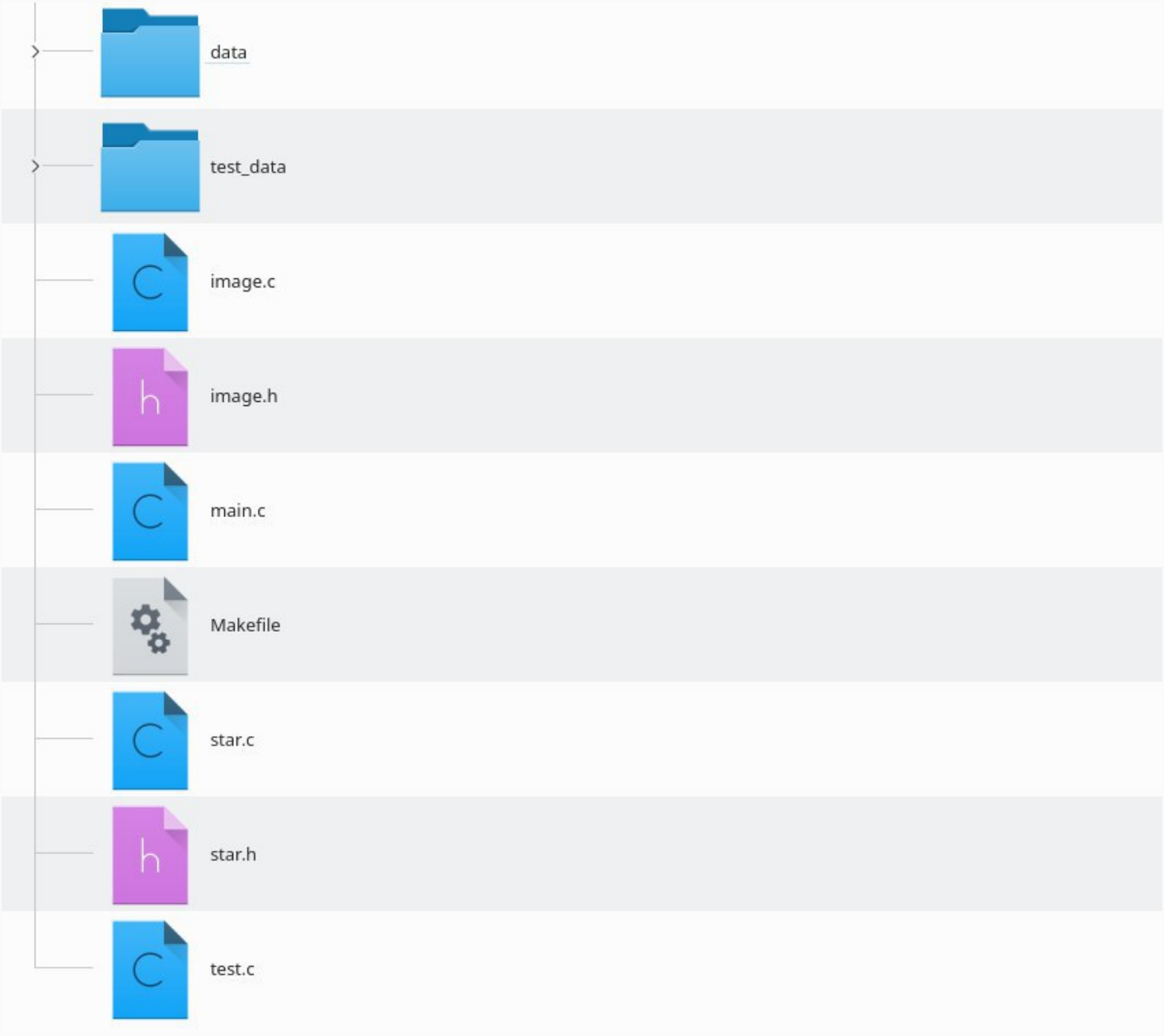


image.h

```c
#ifndef IMAGE_H
#define IMAGE_H

#include <stdio.h>

/**
 * An image data structure with width @p w and height @p h.
 * Use @p image_init to allocate the @p data field and @p image_destroy to deallocate it again.
 */
struct image {
    int w, h;
    int* data;
};

/**
 * Initialize an image structure with a width and a height.
 * We also allocate w * h ints to hold the pixels of the image.
 * Every pixel is an int that contains the red, green, and blue value of the pixel.
 * red is in bits 16-23, green in bits 8-15, blue in bits 0-7.
 */
void image_init(struct image* img, int w, int h);

/**
 * Destroys the image @p img by freeing its data field.
 * Don't use @p img afterwards.
 */
void image_destroy(struct image* img);

/**
 * Draw a pixel to the image.
 * The color is encoded as described in image_init().
 * This function *needs* to clip the pixel, that is, if x and y are outside the image's boundary,
 * the pixel must not be drawn.
 */
void image_draw_pixel(struct image* img, int color, int x, int y);

/**
 * Draws a line from (x0,y0) to (x1,y1).
 */
void image_draw_line(struct image* img, int color, int x0, int y0, int x1, int y1);

/**
 * Writes an image to a portable pixmap (P3) file.
 * See http://en.wikipedia.org/wiki/Netpbm_format for details on the file format.
 */
void image_write_to_file(struct image* img, FILE* f);

#endif
```

image.c

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

#include "image.h"

void image_init(struct image* img, int w, int h)
{
»   abort(); // TODO implement
}

void image_destroy(struct image* img)
{
»   abort(); // TODO implement
}

void image_draw_pixel(struct image* img, int color, int x, int y)
{
»   abort(); // TODO implement
}
```

Projekt Komplieren

Der Programmcode kann mit `make` kompiliert werden:

```
prog2@prog2vm ~/project2 $ make
cc -O0 -g -Wall -pedantic -fsanitize=address --std=c99 -o star.o -c star.c
cc -O0 -g -Wall -pedantic -fsanitize=address --std=c99 -o main.o -c main.c
cc -O0 -g -Wall -pedantic -fsanitize=address --std=c99 -o image.o -c image.c
cc -lasan -o stars star.o main.o image.o
cc -O0 -g -Wall -pedantic -fsanitize=address --std=c99 -o test.o -c test.c
cc -lasan -o test test.o image.o star.o
```

Resultat


image.c
image.h
image.o
main.c
main.o
Makefile
star.c
star.h
star.o
stars
test
test.c
test.o

Ausführung

Das kompilierte Programm und die Tests können nun ausgeführt werden.

Ausführung

Das kompilierte Programm und die Tests können nun ausgeführt werden.

Programm:

`./stars <breite> <sternedatei> [<sternbilddatei>...]` z. B.:

```
./stars 200 data/stars.txt
```

Ausführung

Das kompilierte Programm und die Tests können nun ausgeführt werden.

Programm:

`./stars <breite> <sternedatei> [<sternbilddatei>...]` z. B.:

```
./stars 200 data/stars.txt
```

Tests:

```
./test
```

Ausführung

Das kompilierte Programm und die Tests können nun ausgeführt werden.

Programm:

`./stars <breite> <sternedatei> [<sternbilddatei>...]` z. B.:

```
./stars 200 data/stars.txt
```

Tests:

```
./test
```

Achtung!

Aufruf von abort() bricht die Programmausführung ab

Technische Fragen?

Aufgaben

Bild Daten

`image.c -> image_init(image, w, h)`

`image.c -> image_destroy(image)`

`image.c -> image_draw_pixel(image, color, x, y)`

```
struct image {
    int w, h;
    int* data;
};
```

`image.h`

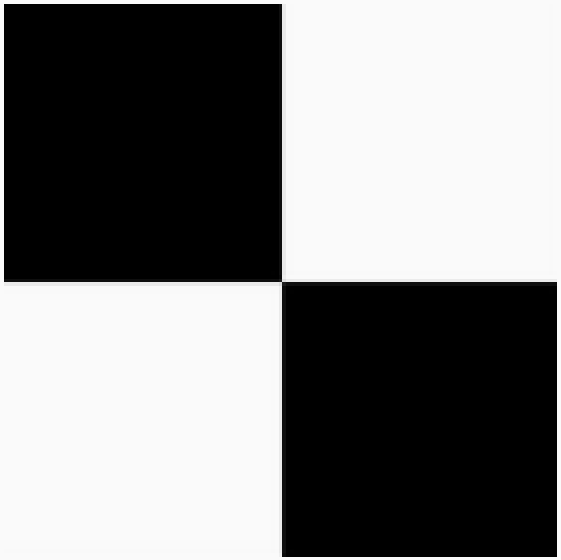
Beispiel

Breite: 7, Höhe: 5, Pixel an Stelle (3, 2)

Array Element: 2 * 7 + 3

Bild schreiben

`image.c -> image_write_to_file(image, file)`



```
P3
2 2
255
0   0   0   255 255 255
255 255 255 0   0   0
```

R-G-B Werte

```
image.c -> image_write_to_file(image, file)
```

Farbe

In einer Ganzzahl stehen 3 Farbwerte!

R-G-B Werte

```
image.c -> image_write_to_file(image, file)
```

Farbe

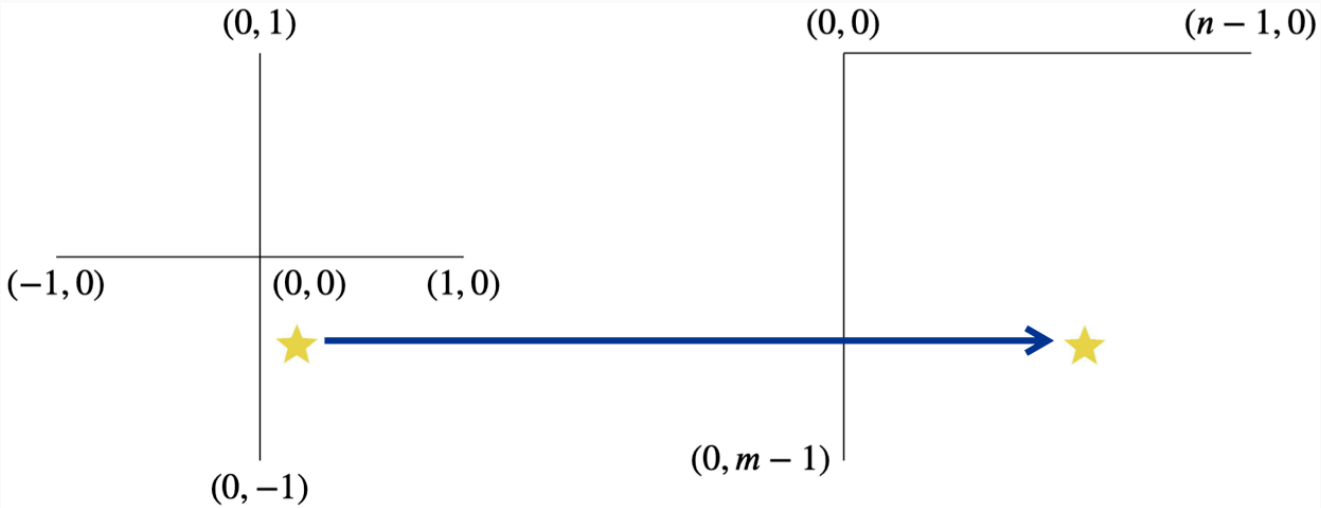
In einer Ganzzahl stehen 3 Farbwerte!

Beispiel

$$0x00\underbrace{FF}_{\text{rot}}\underbrace{A5}_{\text{grün}}\underbrace{00}_{\text{blau}} = \text{orange}$$

Koordinaten transformieren

```
star.c -> star_coord_to_pixel(star, image, x, y)
```



Sterne einlesen

`star.c -> read_star_from_file(star, file)`

```
0.994 0.023 -0.099 28  4.61 3
0.972 0.024 0.231  87  5.55 4
0.435 0.012 0.900  144 5.57 7
0.998 0.033 -0.044 315 6.43 11
0.998 0.035 -0.042 352 6.18 14
0.873 0.031 0.486  358 2.07 15
0.512 0.020 0.858  432 2.28 21
...
```

`data/stars.txt`

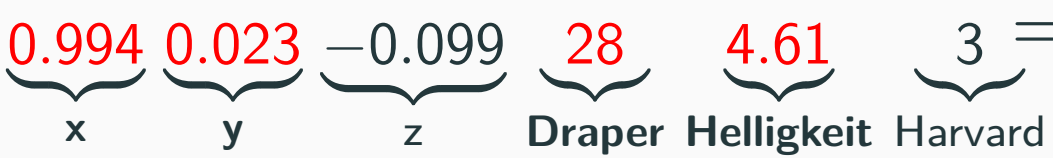
$\Rightarrow$

```
struct star {
    double x, y,
        magnitude;
    int draper;
};
```

`star.h`

Werte der Sterne

`star.c -> read_star_from_file(star, file)`



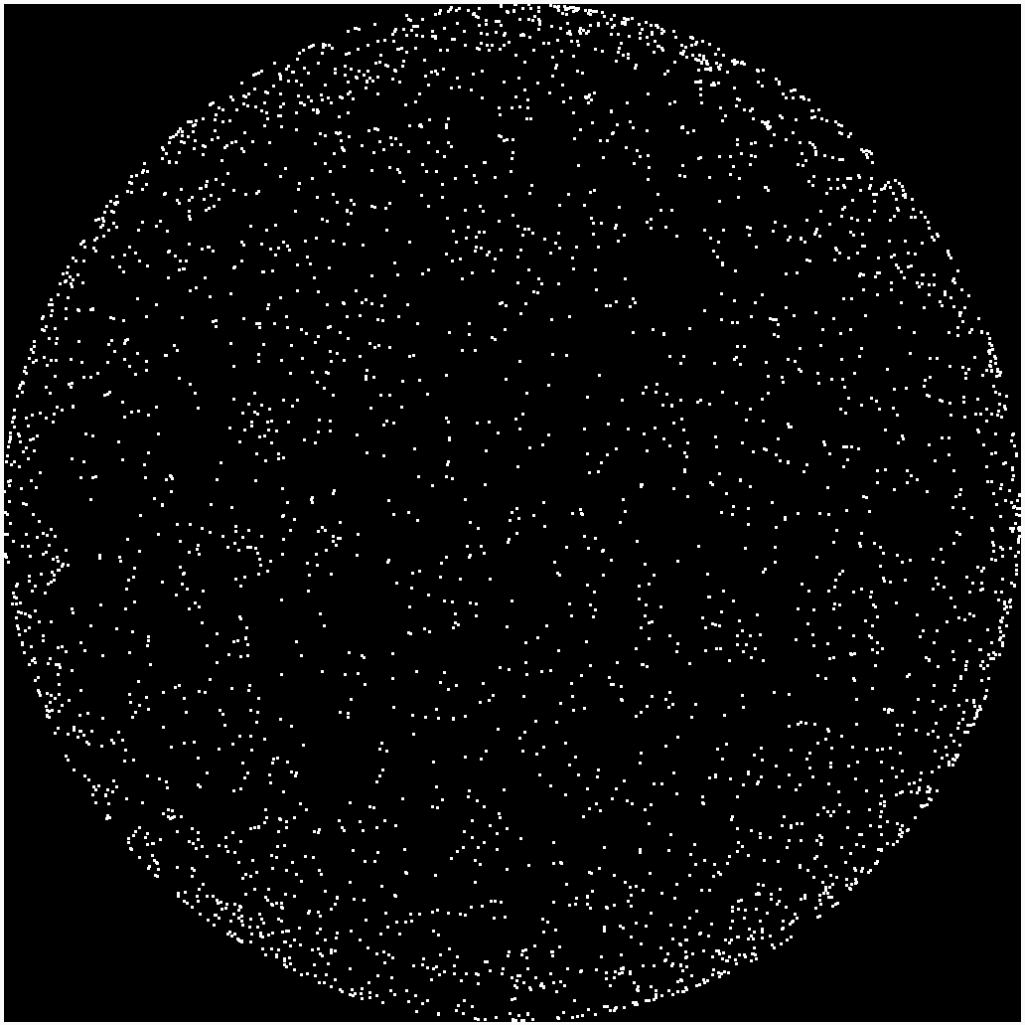

```
struct star {
    double x, y,
           magnitude;
    int draper;
};
```

`data/stars.txt`

`star.h`

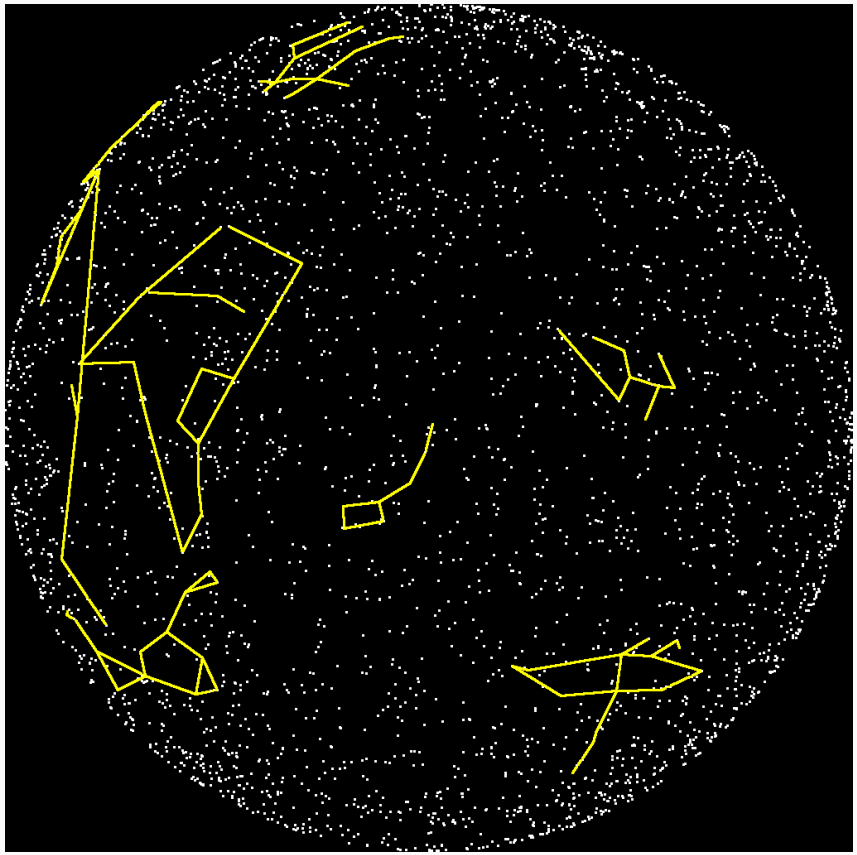
Sterne zeichnen

```
star.c -> star_plot(star, image)
```



Sternbilder zeichnen

```
star.c -> draw_constellation_from_file()
```



Sternbild Dateien

```
120315,116656
116656,112185
112185,106591
106591,103287
103287,95418
95418,95689
106591,95689
95689,71369
85235,84999
...
```

Programme aufrufen

main.c

```c
int main(int argc, char *argv[])
{
    if (argc < 3) {
        fprintf(stderr, "syntax: %s size starlist [lines...]\n", argv[0]);
        return EXIT_FAILURE;
    }

    int size = atoi(argv[1]);
    if (size <= 0) {
        fprintf(stderr, "size of image cannot be <= 0\n");
        return EXIT_FAILURE;
    }

    struct image img;
    image_init(&img, size, size);

    // TODO: Read in the stars from the file with name argv[2]
    // save them in an array in the order they are read in and draw them to the image.
    abort();

    // open every constellation file
    for (int i = 3; i < argc; i++) {
        FILE* f = fopen(argv[i], "r");
        // if the file could not be opened, skip to the next one.
        if (f == NULL) {
            fprintf(stderr, "cannot open line file \"%s\"\n", argv[i]);
            continue;
        }
        abort(); // TODO draw the constellation to the image
        fclose(f);
    }
```

Testing

test.c

```
bool test_transform0()
{
    struct star validstar;
    int x;
    int y;
    int data[100] = {0,0,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0,0};
    struct image img ={10,10, data};

    validstar.x = 1.0;
    validstar.y = 1.0;
    star_coord_to_pixel(&validstar, &img, &x, &y);
    if (x != 9 || y != 0) {
        return FAIL("coordinate transformation incorrect");
    }

    return PASS();
}
```

test.c

```
bool test_read()
{
»   FILE* f = fopen("test_data/validstar.txt", "r");

»   struct star s;
»   fseek(f, 0, SEEK_SET);
»   int res = star_read_from_file(&s, f);
»   fclose(f);
»   if (res != 1)
»   »   return FAIL("wrong result code");
»   if (s.draper != 28)
»   »   return FAIL("wrong draper number read");
»   if (s.x < 0.994771 || s.x > 0.994773)
»   »   return FAIL("wrong x coord read");
»   if (s.y < 0.023163 || s.y > 0.023165)
»   »   return FAIL("wrong y coord read");
»   if (s.magnitude < 4.6 || s.magnitude > 4.62)
»   »   return FAIL("wrong magnitude read");

»   return PASS();
}
```

test.c

```
int main(int argc, char *argv[])
{
    // all tests in an array of function pointers
    static bool (*const all_tests[])() = {
        test_round,
        test_read,
        test_transform0,
        test_pixel_out_of_bounds0,
        test_draw_pixel_colors0,
        test_transfrom_valid_coords_nonquadratic0,
        test_image,
        // feel free to add your own tests here
    };

    static const int num_tests = sizeof(all_tests)/sizeof(*all_tests);

    if (argc == 1) {
        int num_failed = 0;
        for (int i = 0; i != num_tests; ++i) {
            if (!all_tests[i]())
                ++num_failed;
        }
        printf("ran %i tests; %i failed\n", num_tests, num_failed);

        return num_failed ? EXIT_FAILURE : EXIT_SUCCESS;
    }

    if (argc == 2) {
        int i = atoi(argv[1]);
```

test.c

```
int main(int argc, char *argv[])
{
    // all tests in an array of function pointers
    static bool (*const all_tests[])() = {
        test_round,
        test_read,
        test_transform0,
        test_pixel_out_of_bounds0,
        test_draw_pixel_colors0,
        test_transfrom_valid_coords_nonquadratic0,
        test_image,
        // feel free to add your own tests here
    };

    static const int num_tests = sizeof(all_tests)/sizeof(*all_tests);

    if (argc == 1) {
        int num_failed = 0;
        for (int i = 0; i != num_tests; ++i) {
            if (!all_tests[i]())
                ++num_failed;
        }
        printf("ran %i tests; %i failed\n", num_tests, num_failed);

        return num_failed ? EXIT_FAILURE : EXIT_SUCCESS;
    }

    if (argc == 2) {
        int i = atoi(argv[1]);
```

C — Ein Beispiel

Man-Pages

- Einträge für (fast) alle C-Befehle
- "Nutzerhandbuch"
- Aufruf mit: `man <Befehlsname>`
- Bsp: `man scanf`
- Rückkehr in die Terminalansicht: q

man scanf

```
SCANF(3)                          Linux Programmer's Manual                          SCANF(3)

NAME
       scanf, fscanf, sscanf, vscanf, vsscanf, vfscanf - input format conversion

SYNOPSIS
       #include <stdio.h>

       int scanf(const char *format, ...);
       int fscanf(FILE *stream, const char *format, ...);
       int sscanf(const char *str, const char *format, ...);

       #include <stdarg.h>

       int vscanf(const char *format, va_list ap);
       int vsscanf(const char *str, const char *format, va_list ap);
       int vfscanf(FILE *stream, const char *format, va_list ap);

   Feature Test Macro Requirements for glibc (see feature_test_macros(7)):

       vscanf(), vsscanf(), vfscanf():
           _ISOC99_SOURCE || _POSIX_C_SOURCE >= 200112L

DESCRIPTION
       The  scanf() family of functions scans input according to format as described below.  This format may con-
       tain conversion specifications; the results from such conversions, if any, are  stored  in  the  locations
       pointed  to  by the pointer arguments that follow format.  Each pointer argument must be of a type that is
       appropriate for the value returned by the corresponding conversion specification.

       If the number of conversion specifications in format exceeds the number of pointer arguments, the  results
       are  undefined.   If the number of pointer arguments exceeds the number of conversion specifications, then
       the excess pointer arguments are evaluated, but are otherwise ignored.

       The scanf() function reads input from the standard input stream  stdin,  fscanf()  reads  input  from  the
       stream pointer stream, and sscanf() reads its input from the character string pointed to by str.

       The  vfscanf() function is analogous to vfprintf(3) and reads input from the stream pointer stream using a
       variable argument list of pointers (see stdarg(3).  The vscanf() function scans a variable  argument  list
       from  the  standard  input  and  the vsscanf() function scans it from a string; these are analogous to the
       vprintf(3) and vsprintf(3) functions respectively.

       The format string consists of a sequence of directives which describe how to process the sequence of input
       characters.   If processing of a directive fails, no further input is read, and scanf() returns.  A "fail-
       ure" can be either of the following: input failure, meaning that input  characters  were  unavailable,  or
       matching failure, meaning that the input was inappropriate (see below).

       A directive is one of the following:

       ·      A  sequence  of white-space characters (space, tab, newline, etc.; see isspace(3)).  This directive
:
```

C — Ein Beispiel

Aufgabe:
Schreiben Sie ein Programm, dass aus einer gegebenen Datei Daten in folgendem Format einließt:

```
ID Day Month Year Grade, ..., ID Day Month Year Grade
```

Wobei `ID`, `Tag`, `Monat` und `Jahr` Ganzzahlen sind und `Note` eine Gleitkommazahl ist. Das Programm soll den Durchschnitt der Noten berechnen. Die Ausgabe des Programms soll die Liste der Daten und Noten sein, gefolgt vom Durchschnitt.

C — Ein Beispiel

Eingabedatei:

```
1 21 09 1995 2.0, 2 5 06 1994 2.0
```

Ausgabe:

```
Day-Month-Year: 21.9.1995
Grade: 1.00
Day-Month-Year: 5.8.1994
Grade: 2.00
Average: 1.50
```

Demo

Noch zwei Hinweise

Nicht mit uninitialisierten Variablen rechnen (z. B. `int a;`),
sondern diesen **Werten zuordnen** (z.B. `int a = 0;`), da diese
nicht initial auf 0 gesetzt sind !!

In der Projektbeschreibung in Absatz 6 stehen hilfreiche Hinweise zum Projekt!

Vielen Dank für eure Aufmerksamkeit!