Game of Life

Agenda

- Wikipedia Artikel zu GOL
- Demo der fertigen Applikation
- Demo GOL NaCl

Interface 1

Brainstorming

Interface 2

Commandline Interface

- height(-h), width,(-w) filename(-f)
- man 3 getopt

DYI

Datenstrukturen

Brainstorming

Datenstrukturen 2

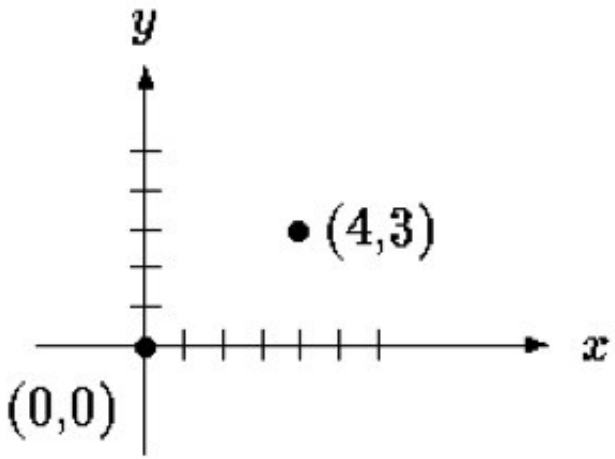
Remember structs?

Datenstrukturen 3

Strukturen (1)

- eine Stuktur ist eine Sammlung einer oder mehrerer Variable
- können von unterschiedlichen Datentypen sein
- gruppiert unter einem neuen eindeutigen Namen
  - *einfacherer Handhabe*

Beispiel Koordinatendarstellung



Datenstrukturen 4

Strukturen (2)

- Die zwei Komponenten werden in einer Struktur definiert

```
1. struct point_t {
2.   int x;
3.   int y;
4. };
```

```
1. // Syntaktlisch analog zu: int x;
2. struct point_t pt;
```

- Ein Mitglied einer Struktur ist wie folgt zu erreichen via

```
1. struct_name.member_name;
```

- *typedef struct* definiert einen neuen Datentyp

```
1. typedef struct point_t point;
```

Erstelle die Welt

- Welt enthaelt Array aus Zellen
  - *Brainstorming*

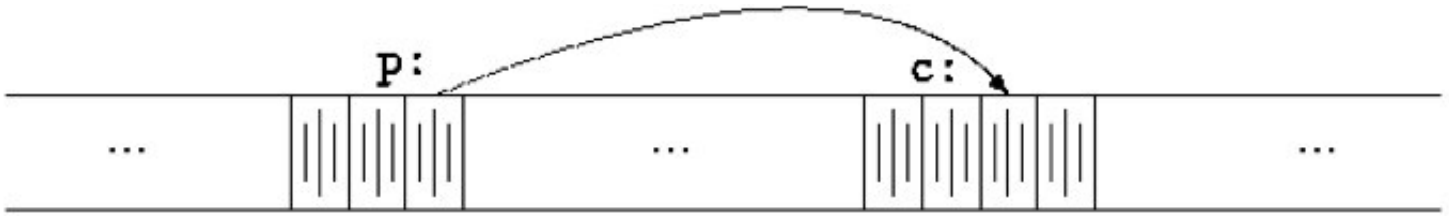
Erstelle die Welt 2

Remember Pointers?

Zeiger und Arrays (1)

- Zeiger(Pointer) ist
  - *eine Variable und enthält Adresse einer anderen Variable*
  - *eine Gruppe aus Speicherzellen*

Beispiel: *c* ist ein 'char' und *p* zeigt darauf



Zeiger und Arrays (2)

- Der unäre Operator & gibt die Adresse eines Objektes zurück
  - *nur für Variable und Array Elemente. Nicht für Ausdrücke, Konstanten und 'register' Variable.*

```
1. p = &c;
```

- Der unäre Operator * ist die Umkehrung (Dereferencing).
  - *er gibt den Wert des Objektes auf das gezeigt wird zurück*

```
1. int x = 1, y = 2, z[10];
2. int *ip;    /* ip is a pointer to int */
3. ip = &x;    /* ip now points to x */
4. y = *ip;    /* y is now 1 */
5. *ip = 0;    /* x is now 0 */
6. ip = &z[0]; /* ip now points to z[0] */
```

- Zeiger und Funktionsargumente
  - *Call by Reference vs Call by Value*

Zeiger und Arrays (3)

- 'int *ip;' ist eine Eselsbrücke und zeigt an, dass die Variable auf die gezeigt wird einen 'int' enthält
  - *Ein Zeiger ist dadurch aber auch beschränkt auf eine Variable des richtigen Typs zu zeigen*
  - *Alternativ 'void *': Zeigt auf jeden Typ, aber kann selbst nicht dereferenziert werden*

Wenn ip auf einen Integer x zeigt, dann kann *ip an jeder Stelle vorkommen an der sonst x stünde:

```
1. *ip = *ip + 10;
2. y = *ip + 1;
3. *ip += 1;
4. ++*ip;
5. (*ip)++; // Parentheses are needed. Otherwise the pointer would be increased,
6.          // not the value behind it's variable.
```

- Da Zeiger Variable sind, können sie auch ohne Dereferenzierung genutzt werden.

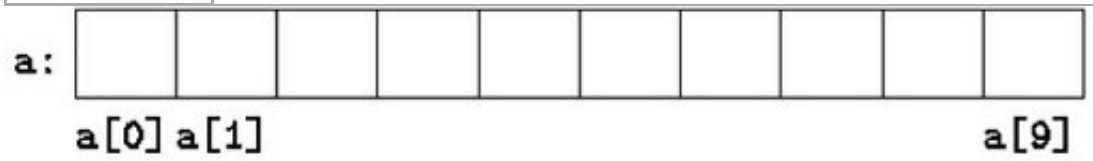
Wenn 'iq' ein anderer Zeiger auf einen int ist:

```
1. iq = ip;
```

Zeiger und Arrays (4)

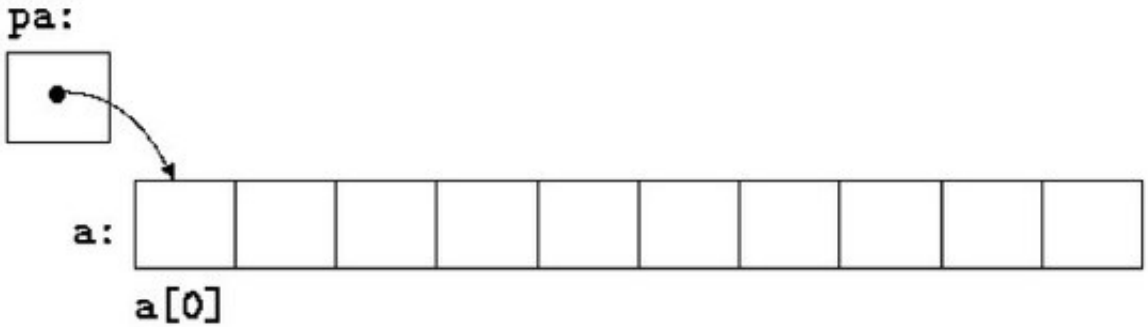
- Zeiger und Arrays miteinander sind eng verwandt
  - *Jede Array-Operation kann auch als Zeiger Operation umgesetzt werden*
  - *Die Zeiger Variante ist in der Regel schneller, aber für Anfänger schwerer zu verstehen*

```
1. int a[10];
```



- Wenn pa ein Zeiger auf einen Integer ist

```
1. int *pa;
2. // sets pa to point to element zero of a. pa has the address of
   a[0].
3. pa = &a[0];
```



Zeiger und Arrays (5)

Forführung Beispiel letzter Slide.

```
1. x = *pa; // will copy the contents of a[0] into x
2. *(pa+1); // refers to the contents of a[i]
```

- Indexierung und Zeiger Arithmetik ist sehr eng verwandt. Daher gibt es folgende Abkürzung:

```
1. pa = &a[0];
2. // is the same as
3. pa = a;
```

- Ebenso:

```
1. *(pa+i);
2. // is the same as
3. pa[i];
```

TODO

- Datenstrukturen für Welt und Zellen
- Speicher bereitstellen fuer alle Zellen
  - *man 3 malloc*
- Zufaellige Welt erstellen
  - *man 3 rand*
  - *man 3 time fuer seed*
- Welt von Datei einlesen
  - *man 3 fseek fuer dateigroesse*
  - *man 3 fread*
- Nachbarn zaehlen
- Naechstes Intervall berechnen

Ziel

- Vorgehen
  - *Schritt für Schritt*
  - *Theorie -> Slides*
  - *Eigenständiges Programmieren*
  - *Codefreigabe auf Github für aktuellen Schritt*
  - *Code Walkthrough*
  - *Q&A*
  - *Repeat*

Datenstrukturen für Welt und Zellen

- Eine Welt hat viele Zellen
- Einzelne Zellen haben einen Zustand 'lebend'
  - *für die aktuelle und nächste Iteration*

Musterlösung

- git pull
- git branch
- git merge 1_world_and_cells

Speicher allozieren für alle Zellen

- Demo calloc_demo.c

```
phorce-mba:~/src/gol$./a.out
Enter the number of type int to allocate: 9999
Memory allocation was successful.
phorce-mba:~/src/gol$./a.out
Enter the number of type int to allocate: 999999999999999999999999
Memory allocation failed.
```

- man 3 malloc

```
     The malloc(), calloc(), valloc(), realloc(), and reallocf() functions allocate memory.  The allocated
     memory is aligned such that it can be used for any data type, including AltiVec- and SSE-related
     types.  The free() function frees allocations that were created via the preceding allocation func-
     tions.

     The malloc() function allocates size bytes of memory and returns a pointer to the allocated memory.

     ...

     The free() function deallocates the memory allocation pointed to by ptr. If ptr is a NULL pointer, no
     operation is performed.
```

Welt zufällig mit Leben füllen

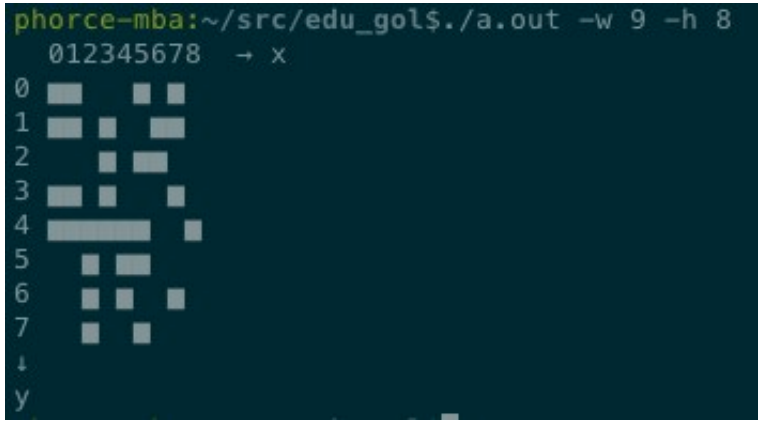
- man 3 rand

```
The rand() function computes a sequence of pseudo-random integers in the range of 0 to RAND_MAX (as
defined by the header file ).

The srand() function sets its argument seed as the seed for a new sequence of pseudo-random numbers
to be returned by rand().  These sequences are repeatable by calling srand() with the same seed
value.

If no seed value is provided, the functions are automatically seeded with a value of 1.
```

* man 3 time fuer seed

```
The time() function returns the value of time in seconds since 0 hours, 0 minutes, 0 seconds, January
1, 1970, Coordinated Universal Time, without including leap seconds.  If an error occurs, time()
returns the value (time_t)-1.
```

Welt ausgeben



Welt aus Datei einlesen

- Interface: ./gol -f [path_to_file]
- man 3 fread

```
The function fread() reads nitems objects, each size bytes long, from the stream pointed to by
stream, storing them at the location given by ptr.
```

- man 3 fseek fuer dateigroesse

```
The fseek() function sets the file position indicator for the stream pointed to by stream.  The new
position, measured in bytes, is obtained by adding offset bytes to the position specified by whence.
If whence is set to SEEK_SET, SEEK_CUR, or SEEK_END, the offset is relative to the start of the file,
the current position indicator, or end-of-file, respectively.  A successful call to the fseek() func-
tion clears the end-of-file indicator for the stream and undoes any effects of the ungetc(3) and
ungetwc(3) functions on the same stream.
```

Game of Life Logik

Nachbarn zählen

- Ziel

```
phorce-mba:~/src/edu_gol$./gol -f seed/blinker
  012  → x
0  ■
1  ■
2  ■
↓
y
neighbours of y(0):x(0) -> 2
neighbours of y(0):x(1) -> 1
neighbours of y(0):x(2) -> 2
neighbours of y(1):x(0) -> 3
neighbours of y(1):x(1) -> 2
neighbours of y(1):x(2) -> 3
neighbours of y(2):x(0) -> 2
neighbours of y(2):x(1) -> 1
neighbours of y(2):x(2) -> 2
```

Game of Life Logik

Nächste Iteration berechnen (next tick)

1. Any live cell with fewer than two live neighbours dies, as if caused by under-population.
2. Any live cell with two or three live neighbours lives on to the next generation.
3. Any live cell with more than three live neighbours dies, as if by overcrowding.
4. Any dead cell with exactly three live neighbours becomes a live cell, as if by reproduction.

Game of Life Logik

Refaktorieren

Game of Life Logik

Done!