Game of Life

Agenda

- Reschedule 14.01.
- Wikipedia
- Demo

Interface 1

Brainstorming

Interface 2

Commandline Interface

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

DYI

Datenstrukturen

Brainstorming

Datenstrukturen 2

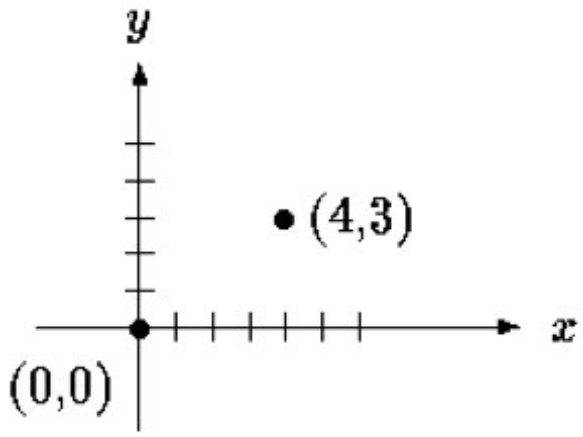
Remember structs?

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;
```

```
1. point c;
2. c.x;
3. c.y;
```

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.

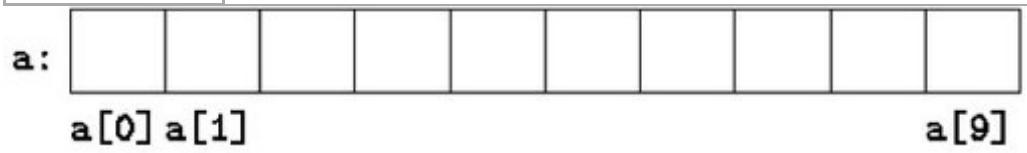
Wenn 'iq' ein anderer Zeiger auf einen int ist:

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

Zeiger und Arrays (4)

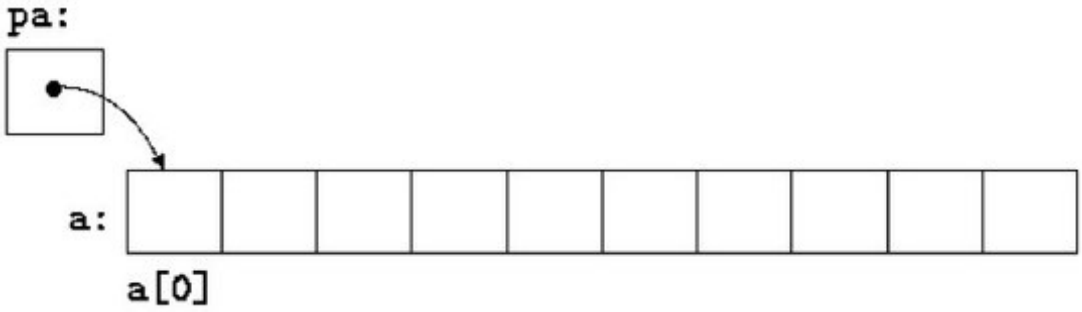
- Zeiger und Arrays miteinander sind eng verwandt
  - *Jede Array-Operation kann auch als Zeiger Operation umgesetzt werden*
  - *Die Zeiger Variante ist in der Regel schneller, aber für Anfänger schwerer zu verstehen*

```
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];
```

- 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*

Welt ausgeben

Game of Life

- Nachbarn zaehlen
- Naechstes Intervall berechnen