Prof. Franz Schweiggert Abteilung Angewandte Informationsverarbeitung 24. April 2006
Christian Ehrhardt Blatt 1

# Systemnahe Software (SS 2006)

Abgabetermin 2.Mai 2006

## Wa-Tor (10 Punkte)

In diesem Blatt werden einige der in Allgemeine Informatik III erarbeiteten Grundlagen wiederholt und vertieft. Nach der eigentlichen Aufgabenstellung folgt eine ausführliche und hoffentlich hilfreiche Anleitung. Wer das Blatt vollkommen selbständig bearbeiten möchte kann diesen Teil überspringen.

### Das Leben auf Wa-Tor

#### Der Planet

Wa-Tor (Wasser Torus) ist ein vollständig mit Wasser bedeckter Planet. Seine Oberfläche ist in kleine rechteckige Zellen eingeteilt, die in Form eines Rings (Torus) angeordnet sind. Für uns bedeutet das, daß wir die Oberfläche einfach als Rechteck darstellen können. Verläßt man das Rechteck auf einer der Seiten, dann erscheint man einfach an der entsprechenden Stelle auf der gegenüberliegenden Seite. Alle Zellen (auch solche am "Rand") haben also genau vier Nachbarn, je einen im Norden, Süden, Osten und Westen (diagonal liegende Zellen zählen nicht als Nachbarn).

#### Die Tiere

Auf Wa-Tor gibt es nur zwei Arten von Tieren: Haie und Fische. Jede Zelle von Wa-Tor, die nicht leer ist, kann entweder einen Hai oder einen Fisch enthalten. Fische ernähren sich von Plankton, das offenbar im Wasser immer ausreichend vorhanden ist, Haie dagegen ernähren sich von Fischen.

#### Bewegung

In jeder Runde hat nacheinander jedes Tier **genau** einmal die Möglichkeit, sich auf eine seiner Nachbarzellen zu bewegen. Das geschieht nach folgenden Regeln:

- Ein Hai sucht sich unter allen Nachbarzellen, die mit einem Fisch besetzt sind zufällig eine aus und zieht auf diese. Der Fisch wird dabei natürlich gefressen.
- Ein Fisch oder ein Hai, der keinen Fisch fressen konnte, sucht sich aus allen freien Nachbarzellen zufällig eine aus und zieht auf diese.
- Sollten überhaupt keine Bewegung möglich sein, dann bleibt das Tier in dieser Runde einfach stehen.

#### Leben und Sterben auf Wa-Tor

Drei Parameter bestimmen das Leben auf Wa-Tor:

**fasten** Ein Hai, der **fasten** Runden lang keinen Fisch gefressen hat, stirbt ohne sich noch zu bewegen. Seine Zelle ist anschließend unbesetzt.

**hbrut** Ein Hai, der in den letzten **hbrut** Runden keinen Nachwuchs bekommen hat, hinterläßt bei seiner Bewegung auf der verlassenen Zelle einen neuen Junghai.

**fbrut** Ein Fisch, der in den letzten **fbrut** Runden keinen Nachwuchs bekommen hat, hinterläßt bei seiner Bewegung auf der verlassenen Zellen einen jungen Fisch.

Für Haie und Fische gilt: Wenn keine Bewegung stattfinden kann, dann gibt es auch keinen Nachwuchs.

### Aufgabe

Eure Aufgabe ist es, eine Programm zu schreiben, das das Leben auf Wa-Tor simuliert. Nach jeder Runde soll der Planet ausgegeben werden (z.B. Haie als "X", Fische als "*" und leere Zellen als Punkt). Zusätzlich soll ausgegeben werden, wieviele Haie und wieviele Fische gerade auf Wa-Tor existieren. Weiterhin soll es möglich sein, den Namen einer Datei auf der Kommandozeile anzugeben. Wenn eine solche Datei angegeben wurde, dann wird in jeder Runde eine Zeile bestehend aus der Rundennummer, der Anzahl der Haie und der Anzahl der Fische in die Datei geschrieben. Hat man als Namen der Datei STATISTIK angegeben, dann kann man nach dem Aufruf des Kommandos gnuplot folgendermaßen die Entwicklung der Population graphisch darstellen lassen:

```
thales$ gnuplot
...
gnuplot> plot "STATISTIK" using "%lf %*lf %lf" with lines, \
> "STATISTIK" using "%lf %lf %*lf" with lines;
gnuplot> exit
thales$
```

## Anleitung und Hinweise

Ab hier wollen wir versuchen, dem doch etwas umfangreicheren Problem in kleine übersichtlichen Schritten Herr zu werden. Nach jedem Schritt sollte das Programm gespeichert, übersetzt und soweit möglich auch getestet werden.

## Parameter

Folgende Parameter bestimmen offenbar den Planeten Wa-Tor:

- Die Anzahl der Zeilen und Spalten im Rechteck
- Die oben beschriebenen Parameter hbrut, fbrut und fasten.

Definiert für diese 5 Werte Präprozessorkonstanten mit sprechenden Namen. Vernünftige Werte sind: 14 Zeilen, 34 Spalten, hbrut=10, fbrut=3 und fasten=3. Übersetzen nicht vergessen.

## Datenstrukturen

### Tiere

Wir müssen von jedem auf Wa-Tor existierenden Tier wissen:

- Seine Art, also ob es sich um einen Hai oder einen Fisch handelt, z.B. 1 für HAIE und 2 für Fische. Es lohnt sich sicherlich, Makros statt 1 bzw. 2 zu verwenden.
- Sein "Alter", also die Anzahl an Runden, seit denen das Tier keinen Nachwuchs mehr bekommen hat.
- Bei einem HAI müssen wir zusätzlich noch wissen, seit wievielen Runden er keinen Fisch mehr gefressen hat.

Definiert also die oben erwähnten Makros und eine Struktur (`struct animal`), die ein einziges Tier beschreibt. Das heißt die Struktur muß im wesentlichen die oben angegebenen Informationen aufnehmen können. Übersetzen nicht vergessen.

### Der Planet

Der Planet selbst soll durch ein globales zweidimensionales array dargestellt werden. Das Array braucht natürlich genau so viele Zeilen und Spalten wie der Planet Wa-Tor. Jeder Eintrag ist entweder NULL oder er zeigt auf eine `struct animal`, die das Tier beschreibt, das sich in dieser Zelle befindet. Definiert diese globale Variable und schreibt eine passende main-Funktion, die alle Einträge in dem Array mit NULL-Zeigern initialisiert. Übersetzen nicht vergessen.

## Die Ausgabe

Damit wir möglichst bald etwas zu sehen bekommen, schreiben wir als erstes eine Prozedur `show_wator`, die den Zustand des Planeten Wa-Tor auf den Bildschirm ausgibt. Bei einem ersten Test sollte diese Prozedur ein mit leeren Zellen (dargestellt durch je einen Punkt) gefülltes Rechteck auf dem Bildschirm ausgeben.

## Die ersten Tiere

Schreibt eine Prozedur `new_animal`, die ein neues Tier an einer bestimmten Stelle im Planet Wa-Tor plaziert. Der Prozedur wird also die Art des zu erzeugenden Tieres und dessen neue Position in Form von Zeilen- und Spaltennummer übergeben.
Die Prozedur muß natürlich Speicher für eine `struct animal` mit Hilfe von `malloc` allozieren. Vergeßt nicht, die einzelnen Felder der Struktur auch zu initialisieren (am besten alle außer der Art des Tieres mit 0).
Um später Fehler leichter zu bemerken, sollte die Prozedur (z.B. mit Hilfe von `assert`) überprüfen, ob

- Die angegebenen Koordinaten gültig sind.
- Daß die dadurch angegebene Zelle in Wa-Tor leer ist.
- Daß die für das neue Tier angegebene Art entweder Hai oder Fisch ist und nichts anderes.

Versucht die neue Prozedur zu testen, indem Ihr an einigen Stellen (in der Mitte am Rand und in den Ecken) "von Hand" Tiere plaziert. Dabei kann auch gleich die Funktion `show_wator` ausführlicher getestet werden.

## Zum ersten Mal Zufall

Die Funktion `rand()` (siehe Manual) liefert (gleichverteilte) Pseudozufallszahlen zwischen 0 und `RAND_MAX`. Die erzeugte Folge sieht zwar recht zufällig aus, hängt aber in Wirklichkeit nur vom ersten Wert ab. Damit wir nicht jedesmal die selbe Folge bekommen, empfiehlt es sich, zu Beginn den Zufallszahlengenerator (einmal!) mit dem Aufruf von `srand (time(NULL))` zu initialisieren. Anschließend kann immer `rand()` verwendet werden.
Nachdem die Initialisierung im Hauptprogramm ergänzt wurde, brauchen wir später noch eine Funktion `myrand(int max)`, die eine (gleichverteilte) Zufallszahl zwischen 0 (einschließlich) und dem Parameter `max` (ausschließlich) zurückliefert. Übersetzen nicht vergessen.

## Die Initialisierung von Wa-Tor

Vor der ersten Runde soll Wa-Tor zufällig so initialisiert werden, daß ca. die Hälfte aller Zellen mit einem Fisch und ca. ein Zehntel aller Zellen mit einem Hai belegt ist. Schreibt also eine Funktion `init_wator`, die für jede Zelle eine Zufallszahl zwischen 0 und 100 bestimmt. Wenn diese Zahl kleiner als 50 ist, wird in der Zelle ein Fisch plaziert, wenn die Zahl zwischen 50 und 60 liegt, wird ein Hai plaziert, sonst bleibt die Zelle leer. Übersetzen und testen nicht vergessen.

## Bewegungsrichtungen

### Konstanten

Es gibt vier Bewegungsrichtungen: Norden, Süden, Osten und Westen. Definiert zunächst Konstanten für diese vier Richtungen.

### Zeile und Spalte von Nachbarzellen

Schreibt eine Funktion `ni`, die Zeilennummer und Spaltennummer einer Zelle und eine der vier Bewegungsrichtungen übergeben bekommt. Die Funktion soll dann die Zeilennummer der Zelle zurückliefern, die von der Zelle aus in der angegebenen Richtung liegt. Achtet darauf, daß der Rückgabewert auch dann richtig ist, wenn die ursprüngliche Zelle am "Rand", also ganz oben oder ganz unten liegt. Ihr werdet feststellen, daß auf die Angabe der Spaltennummer sogar verzichtet werden kann.
Schreibt weiterhin analog dazu eine Funktion `nj`, die das selbe für die Spaltennummer erledigt. Bei dieser Funktion kann auf die Angabe der Zeilennummer als Parameter verzichtet werden. Übersetzen und Testen nicht vergessen.

## Nachbarzellen

Als nächstes benötigen wir eine Funktion, die alle Nachbarzellen einer gegebenen Zelle daraufhin überprüft, ob sie einen Fisch enthält. Wenn das der Fall ist, soll eine dieser Nachbarzellen zufällig ausgewählt werden. Die Funktion muß also alle vier Bewegungsrichtungen durchgehen und die jeweilige Nachbarzelle untersuchen. Wenn diese ein Tier enthält und dieses Tier ein Fisch ist, dann kommt diese Bewegungsrichtung in Frage.
Alle in Frage kommenden Bewegungsrichtungen werden gezählt und in einem lokalen Array gespeichert. Aus den in Frage kommenden Richtungen wird dann mit Hilfe von `myrand` eine ausgesucht und zurückgeliefert. Die Funktion muß auch einen speziellen Rückgabewert vorsehen, falls keine Nachbarzelle einen Fisch enthält.
Wo wir gerade dabei sind schreiben wir ganz analog noch eine weitere Funktion, die aus allen leeren Nachbarzellen zufällig eine aussucht und die zugehörige Bewegungsrichtung zurückliefert. Auch hier muß natürlich berücksichtigt werden, daß es eventuell gar keine leere Nachbarzelle gibt. Übersetzen und testen nicht vergessen.

## Bewegung eines Tieres

### Bewegung auf eine leere Zelle

Jetzt können wir uns daran machen, eine Prozedur zu schreiben, die tatsächlich ein Tier bewegt. Zunächst beschränken wir uns dabei auf den Fall, daß ein Hai oder ein Fisch auf eine leere Nachbarzelle bewegt werden soll. Dazu wird eine entsprechende Nachbarzelle mit Hilfe der oben geschriebenen Funktion ausgesucht. Falls es eine solche Zelle gibt, wird der Hai oder der Fisch in diese Zelle bewegt, d.h. der Zeiger für die neue Zelle muß jetzt auf unser Tier zeigen, der Zeiger der alten jetzt leeren Zelle wird zunächst mit NULL belegt.
Sollte das Alter des Tieres den Wert `hbrut` für Haie oder `fbrut` für Fische erreicht oder überschritten haben, dann wird das Alter zurück auf 0 gesetzt und in der alten Zelle ein neues Tier der selben Art (Hai oder Fisch) erzeugt. Eine passende Funktion gibt es bereits. Der Rückgabewert unserer neuen Funktion sollte anzeigen, ob eine Bewegung möglich war oder ob keine leere Nachbarzelle gefunden wurde. Übersetzen nicht vergessen.

### Bewegung auf eine Zelle mit einem Fisch

Ganz analog benötigen wir eine Funktion, die einen Hai auf eine Nachbarzelle mit einem Fisch bewegt. Die Funktion sollte natürlich überprüfen, ob das zu bewegende Tier tatsächlich ein Hai ist. Außerdem ist darauf zu achten, daß der Speicherplatz für den gerade gefressenen Fisch freigegeben wird. Wenn ein Fisch gefressen werden konnte, dann muß der Hungerzähler des Hais auf 0 zurückgesetzt werden. Auch diese Funktion sollte ggf. einen neuen Hai in der alten Zelle erzeugen und über ihren Rückgabewert mitteilen, ob die Bewegung möglich war oder nicht.

### Die komplette Bewegung eines Tieres

Jetzt können wir die beiden gerade geschriebenen Funktionen verwenden, um die Bewegung eines einzelnen Tieres zu implementieren. Zunächst wird natürlich das Alter und im Falle eines Hais auch der Hungerzähler erhöht. Sollte der Hungerzähler eines Hais den Wert `fasten` erreichen, wird der Hai entfernt, seine Zelle ist also anschließend leer und wir sind bereits fertig (Speicher freigeben nicht vergessen). Wenn das Tier ein Hai ist und überlebt, muß zunächst versucht werden, ihn auf eine Nachbarzelle mit einem Fisch zu bewegen. Die passende Funktion dafür haben wir gerade eben geschrieben. War die Bewegung möglich (sieht man ja am Rückgabewert), dann sind wir mit diesem Tier bereits fertig.
Wenn keine Bewegung möglich war oder wenn das Tier ein Fisch ist, dann muß zuletzt versucht werden, das Tier auf eine leere Nachbarzelle zu bewegen. Auch hierfür gibt es ja eine passende Funktion.

### Bewegung aller Tiere

Als nächstes brauchen wir noch eine Funktion, die jedes Tier auf Wa-Tor (falls es nicht vorher gefressen wird) genau einmal bewegt. Wir müssen uns dazu für jedes Tier merken, ob es in dieser Runde schon bewegt wurde. Dazu wird der Datenstruktur `struct animal` ein zusätzliches Feld hinzugefügt, das diese Information aufnimmt. Zu beginn der Runde wird es für alle Tiere auf 1 gesetzt. Anschließend werden die Tiere der Reihe nach mit Hilfe der oben geschriebenen Funktion bewegt, allerdings nur, wenn sie in dieser Runde noch nicht bewegt wurden.

## Ein Hauptprogramm

Im Hauptprogramm wird nach der Initialisierung in einer Endlosschleife die Funktion zum Bewegen aller Tiere und die Funktion `show_wator` im Wechsel aufgerufen. Eventuell ist es hilfreich, zwischen zwei Runden auf eine Tastatureingabe zu warten.

## Statistik in einer Datei

Wenn als Kommandozeilenargument ein Dateiname angegeben wurde, dann sollte die Datei zu Beginn des Hauptprogramms geöffnet und dabei ggf. neu angelegt oder auf 0 verkürzt werden. Wurden keine Kommandozeilenargumente angegeben, dann ist diesbezüglich natürlich auch nichts zu tun.
Der von `open` zurückgelieferte Filedeskriptor wird in einer globalen Variable gespeichert, die wenn keine Datei angegeben wurde mit -1 initialisiert wird. Zusätzlich wird die Funktion `show_wator` so ergänzt, daß sie die aktuelle Runde, die Zahl der Haie und die Zahl der Fische in die geöffnete Datei schreibt, falls der Filedeskriptor nicht -1 ist. Sollte regelmäßig Haie oder Fische aussterben, dann ist irgendwo noch ein kleiner Fehler versteckt. **Viel Erfolg!**