ulm university universität
uulm

Institut für Angewandte Informationsverarbeitung
Dr. Andreas F. Borchert
Fabian Berstecher

3. November 2017
Blatt 3

Systemnahe Software I (WS 2017/2018)

Abgabe bis zum 10. November 2017, 14:00 Uhr

Lernziele:

- Präprozessor-Anweisungen
- Implementierung von Bit-Arrays auf Basis eines ganzzahligen Datentyps

Aufgabe 4: Zur Funktionsweise des C-Präprozessors

Um die Funktionsweise und Möglichkeiten des C-Präprozessors genauer kennen zu lernen, wollen wir in dieser Aufgabe das ein traditionelles Arbeitslied der Seefahrer *What shall we do with the drunken sailor* betrachten. Den Text stellen wir auf der Vorlesungsseite zur Verfügung.

Der Text des Liedes wird im Rahmen dieser Aufgabe in mehrere Dateien zerlegt und etwas komprimiert. In der vorgegebenen Datei *strophe.h* finden sich die Texte der einzelnen Strophen, jedoch ohne den sich wiederholenden Refrain.

Die sich wiederholenden Teile sind als Makros in der Header-Datei *macros.h* unterzubringen. In *sailor.c* sollten Sie dann unter Verwendung von *macros.h* und *strophe.h* nach Nennung des Titels der gesamte Text folgen lassen. Am Ende sollte es möglich sein, mit

```
thales$ gcc -E sailor.c | sed '/^#/d'
```

den gewünschten Text zu produzieren. Die Option „-E“ des *gcc* nimmt nur die Funktionalität des Präprozessors in Anspruch, ohne den so erzeugten Text an den eigentlichen Übersetzer weiterzugeben. Die Ausgabe erfolgt dann auf die Standardausgabe, die hier mit den Werkzeugen *sed* von unnötigen Ballast befreit wird.

Etwas trickreich ist die Inklusion der jeweiligen Strophe aus *strophe.h*. Dies gelingt durch eine mehrfache Inklusion dieser Datei, wobei jeweils vorher mit Hilfe von **#define** das die Strophe auswählende Symbol zu definieren ist, dann mit **#include** die Inklusion erfolgt und anschließend mit **#undef** die Definition des Symbols wieder entfernt wird. In *strophe.h* sind

alle Strophen enthalten. Es werden aber mit Hilfe von **#ifdef** nur die Strophen jeweils berücksichtigt, deren Symbol vorher definiert wurde.

Einreichen können Sie Ihre Lösung mit folgendem Kommando auf der Thales:

```
submit ss1 4 sailor.c macros.h
```

Wenn Sie Ihre Lösung einreichen, wird diese einem automatisiertern Test unterzogen. Die Ausgabe des Präprozessors wird (abgesehen von den Leer- und Hash-Zeilen) mit dem Originaltext verglichen.

Aufgabe 5: Grundy's Game

Zu implementieren ist das Spiel Grundy's Game. Bei diesem Spiel treten zwei Spieler gegeneinander an. Zu Beginn liegen n gleiche Gegenstände zusammen auf einem einzigen Haufen. Die Zahl n sollte zu Beginn innerhalb geeigneter Grenzen pseudo-zufällig gewählt werden. Die Spieler wählen abwechselnd einen der Haufen aus und teilen diesen in zwei unterschiedlich große Haufen. Derjenige Spieler, der keinen gültigen Zug mehr machen kann (das ist der Fall, wenn nur noch Haufen der Größe 1 oder 2 übrig sind) verliert das Spiel.

Hier ist ein beispielhafter Spielverlauf:

```
thales$ grundy
*** Grundy's Game ***
One large heap of 22 beans is given. In each move one of the heaps
may be split into two smaller heaps, provided the resulting heaps
are of different size. The player who is unable to split any remaining
heap loses.

Beans are represented by dots, possible splitting points with digits.
If you are asked for a move, specify one of the breaking points.
Do you want to start? Yes=1 No=2 1
.01.02.03.04.05.06.07.08.09.10..12.13.14.15.16.17.18.19.20.21.
Your move: 14
.01.02.03.04.05.06..08.09.10.11.12.13. .15.16.17..19.20.21.
Computer splits a heap at position 1.
. .02.03.04.05.06.07.08.09.10.11.12.13. .15.16.17..19.20.21.
Your move: 3
. .. .04.05.06.07.08.09.10.11.12.13. .15.16.17..19.20.21.
Computer splits a heap at position 4.
. .. . .05.06.07.08..10.11.12.13. .15.16.17..19.20.21.
Your move: 6
. .. . .. .07.08.09..11.12.13. .15.16.17..19.20.21.
Computer splits a heap at position 7.
. .. . .. . .08.09.10.11.12.13. .15.16.17..19.20.21.
Your move: 21
```

```
. .. . .. . .08.09.10.11.12.13. .15.16.17.18.19.20. .
Computer splits a heap at position 8.
. .. . . .. . . .09.10..12.13. .15.16.17.18.19.20. .
Your move: 10
. .. . . .. . . .. .11..13. .15.16.17.18.19.20. .
Computer splits a heap at position 11.
. .. . . .. . . .. . .12.13. .15.16.17.18.19.20. .
Your move: 17
. .. . . .. . . .. . .12.13. .15.16. .18..20. .
Computer splits a heap at position 12.
. .. . . .. . . .. . . .. .15.16. .18..20. .
Your move: 15
. .. . . .. . . .. . . .. . .. .18..20. .
Computer splits a heap at position 18.
. .. . . .. . . .. . . .. . .. . .19.20. .
Your move: 19
. .. . . .. . . .. . . .. . .. . . .. .
Congratulations, you've won!
thales$
```

Im Rahmen der Aufgabe ist der Spielstand mit Hilfe einer Variable des Typs *unsigned long int* zu realisieren, die uns mindestens 32 Bits zur Verfügung stellt. Die Verwendung von regulären Arrays für die Verwaltung des Spielstands ist ausdrücklich nicht zulässig.[1]

Zu implementieren ist dann ein Spielverlauf, bei dem dann abwechselnd ein Zug vom menschlichen Spieler und dann von Ihrem Programm bestimmt wird. Ihre Lösung muss keine optimalen Züge finden; es genügt vollkommen, wenn die Züge zulässig sind. Sie sind aber herzlich eingeladen, mit Hilfe der Literatur[2] Ihre Lösung zu verbessern.

Sicherzustellen ist aber, dass bei Einlesefehlern (wie etwa einem nicht erfolgreichen *scanf*) der Ablauf terminiert wird. Die eingegebenen Spielzüge sind zu überprüfen und bei unzulässigen Zügen ist eine passende Fehlermeldung auszugeben und die Gelegenheit für eine erneute Zugeingabe zu geben.

Einreichen können Sie Ihre Lösung mit folgendem Kommando auf der Thales:

```
submit ss1 5 grundy.c
```

Wenn Sie Ihre Lösung einreichen, wird diese einer Reihe automatisierter Tests unterzogen. Unter anderem wird überprüft, ob Sie entgegen der Aufgabenstellung doch Arrays in Ihrem Programm verwenden. Außerdem sollten Sie darauf achten, Variablen so lokal wie möglich zu deklarieren.

Viel Erfolg!

[1] Es ist aber zulässig, ein Array für bereits berechnete Nim-Werte zu verwenden, um schneller zum Sieg führende Züge zu bestimmen. Diese sollten dann aber in der entsprechenden Funktion mit **static** deklariert werden.

[2] Elwyin R. Berlekam, John H. Conway, Richard K. Guy: *Winning Ways for Your Mathematical Plays*. Volume 1, Second Edition, A K Peters, Natick 2001, ISBN 1-56881-130-6, S. 96–97.