Algorithmische Mathematik I

Wintersemester 2009/2010
Prof. Dr. Mario Bebendorf
Dr. Jan Hamaekers

universitätbonn

Vorbereitung zu Programmieraufgaben

Zur Arbeit mit `C/C++` oder `java` sind lediglich ein normaler Text-Editor und ein entsprechender Compiler notwendig.

Text-Editor

Im Prinzip können Sie hier von Windows `notepad` über machtvolle Texteditoren wir `vim`, `emacs` oder `ultraedit` bis hin zu voll funktionsfähigen Entwicklungswerkzeugen wie `eclipse` alles benutzen.
Der Vorteil von fortgeschrittenen Editoren ist, dass dort manche wichtige Teile farbig markiert werden, Klammern und Tab-Stops automatisch gesetzt werden und auch manch andere Arbeit abgenommen werden kann.

Compiler

In der Vorlesung ist nicht die Zeit, vertiefend auf moderne Programmiersprachen einzugehen. Daher werden sämtliche Beispiele in `C` ausgeführt. Ein sehr guter — und dazu noch freier — `C`-Compiler ist der `gcc/g++`. Er ist für alle gängigen Plattformen verfügbar (Windows, Linux, Max). Der Compiler wird mit dem `Cygwin`-Paket ausgeliefert, dass zudem auch noch eine Reihe von Werkzeugen zur Arbeit mit Kommandozeilen mitliefert. Informationen zur Installation und weitere Links sind unter

`http://wissrech.ins.uni-bonn.de/teaching/algmath/ws09_i/compiler.html`

aufgeführt. Nach erfolgreicher Installation funktioniert der `gcc` wie folgt.

- Man erstelle die Quellcode-Datei(en) mit seinem Texteditor. Wir nehmen an, die Datei heisst `beispiel.c`.
- Dann gibt man an einer Kommandozeile

  ```
  gcc beispiel.c
  ```

- Dadurch wird eine ausführbare Datei namens `a.out` erstellt, die man durch Eingabe von

  ```
  a.out
  ```

  starten kann. Man kann natürlich auch einen Doppelklick nutzen, dann verschwindet aber u.U. die Textausgabe sofort wieder. Wenn man das Programm anders als `a.out` nennen möchte, kann man das mit

  ```
  gcc -o beispiel beispiel.c
  ```

  tun — in diesem Fall heißt die resultierende Datei `beispiel`.

Beispielprogramme

In der Vorlesung wurden Auszüge aus C-Programmen gezeigt. Um diese konkret in ein ausführbares Machinenprogramm zu übersetzen, benötigt es noch etwas mehr. Dazu betracht wir das folgende Beispiel. Einige Erläuterungen finden sich unter dem Codefragment.

Algorithmus 1.3: Umwandlung Dezimal zu b-adisch. Die Datei ist auf der Webseite.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <assert.h>
4
5  /* hier startet jedes C/C++ - programm: */
6  int main()
7  {
8      /* Vorbereiten der Argumente: */
9      const unsigned int n = 4;
10     unsigned int x = 1364;
11     unsigned int b = 8;
12
13     unsigned int y[n];
14     unsigned int copy_of_x = x;
15
16     unsigned int i = 0;
17     for( i = 0; i<n; i++ ) y[i] = 0;
18
19     /* Start des Algorithmus: */
20     i = 0;
21     while( x > 0 ) {
22         assert( i < n );     /* Sicherheitsabfrage */
23         y[i]    = x % b;
24         x       = x / b;
25         i++;
26     }
27
28     /* Ausgabe des Resultats: */
29     printf(
30         "Umwandlung␣von␣x=(%d)_10␣ins␣%d-adische␣System␣der␣Breite␣n=%d:\n
31         "y=(",
32         copy_of_x,b,n );
33     /* gebe (y_{n-1} y_{n-2} ... y_1 y_0) aus. */
34     for( i = 0; i<n; i++ ) printf( "%d,␣", y[n-1-i] );
35     printf( ")_%d\n", b );
36     return 0; /* beenden des programms */
37 }
```

Dazu einige Erläuterungen:

- In Zeile 6 wird der Eintrittspunkt definiert. Sobald man die ausführbare Datei startet, wird hier angefangen mit dem Programm — egal, was noch alles irgendwo steht. Man kann auch „`int main(int argc, char** argv)`“ schreiben, wenn man dem Programm Argumente übergeben will. Dann ist „`int argc`“ die Anzahl der Argumente und `char** argv` sind die Argumente.[1] Beispielsweise könnte man

[1] Der Zugriff ist dann so: `argv[0]` ist der Name des Programms (`a.out`), `argv[1]` das erste Argument, `argv[2]` das zweite und so weiter.

dann mit dem Programmaufruf

```
a.out 1368 8
```

Die Werte x und b einlesen.[2] Näheres dazu finden Sie bei Bedarf in der Literatur.

- In Zeile 22 wird eine Sicherheitsabfrage durchgeführt. Das `assert()` prüft die Bedingung — in unserem Fall `(i<n)` — und bricht das Programm mit einer Fehlermeldung ab, wenn die Bedingung verletzt ist.

  Diese Art der Abbruchbedingung ist reichlich brutal und wird in der Praxis selten benutzt. Besser sind Sicherheitsabfragen, die ähnlich wenig Zeilen zu schreiben sind, aber trotzdem eine vielsagende Fehlermeldung produzieren und nicht direkt das Programm beenden. Für den Anfang ist aber ein `assert()` besser als nichts; andere Methoden findet man in der Literatur.

Algorithmus 1.5: Umwandlung b-adisch zu Dezimal. Die Datei ist auf der Webseite.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <assert.h>
4
5  /* hier startet jedes C/C++ - programm: */
6  int main()
7  {
8      /* Vorbereiten der Argumente: */
9      const unsigned int n = 4;
10     /* ( y_0, y_1, y_2, y_3 ... y_n ) */
11     /* hier ist die Eingabe andersrum als man liest, Vorsicht */
12     unsigned int    y[] = {4, 2, 5, 2};
13     unsigned int    b = 8;
14     unsigned int    x = 0;
15     signed int      i = 0;
16
17     /* Start des algorithmus: */
18     for( i = n-1; i>=0; i-- ) x = x*b + y[i];
19
20     /* Ausgabe des Resultats: */
21     printf( "Umwandlung␣von␣y=(" );
22     for( i = 0; i<n; i++ ) printf( "%d,␣", y[n-1-i] );
23     printf( ")_%d␣ins␣Dezimalsystem:␣x=%d\n", b, x );
24     return 0; /* beenden des Programms */
25 }
```

Literatur

Bitte folgen Sie den Literaturhinweisen auf unserer Webseite,
`http://wissrech.ins.uni-bonn.de/teaching/algmath/ws09_i`.

Hinweis: Die Quellcodes für die oben angegebenen Programme finden sich auf der Vorlesungswebseite. Falls die Einrückung in Ihrem Editor anders aussieht, können Sie Ihren Editor so konfigurieren, daß er für jeden `TAB` (Tabulatortaste) 4 Leerzeichen einfügt.

[2] In `C` können die Werte auch innerhalb der Laufzeit z.B. mit Hilfe der Funktion `scanf` eingelesen werden.