Prof. Dr. Franz Schweiggert Institut für Angewandte Informationsverarbeitung 19. Oktober 2009
Michael Mattes / Wolfgang Kaifler Blatt 2

WiMa–Praktikum I / Programmierpraktikum im Wintersemester 2009/2010

Abgabetermin: bis 27. Oktober 2009

Präprozessor mal anders

Mit Hilfe des C-Präprozessors wollen wir in diesem Blatt den Liedtext des *Beatles*-Lieds *Let it be* bearbeiten. Eine bereinigte Form davon ist auf der Vorlesungshomepage zugänglich. Das Ziel ist es, diese Version so anzuändern, dass der Originaltext erst wieder durch den Aufruf des Präprozessors (Tipp: `gcc -E`) zu sehen ist. Leerzeilen und Zeilen mit einem `#` am Anfang können dabei ignoriert werden.

4 Makros (3 Punkte)

Suchen Sie drei Textstücke, die mindestens einmal wiederholt werden, und ersetzen Sie diese durch Makros. Außerdem sollen Autor und Titel des Lieds durch Makros ersetzt werden. Tipp: Wenn Sie das Makro `LIB` für "let it be" verwenden, ersparen Sie sich in den folgenden Aufgaben etwas Arbeit.

5 Einbindung weiterer Dateien (4 Punkte)

Da die Makrodefinitionen im Text stören, sollen Sie diese in eine eigene Datei `makros.h` verlegen. Ebenso sollen Sie den Refrain in die Datei `refrain.h` packen. Verwenden Sie aber auch dort die vorhin definierten Makros.

6 Abwandlung von Strophen (4 Punkte)

Da die Strophen einen ähnlichen Aufbau haben, kann man diese ebenfalls abwandeln. Verwenden Sie die Datei `strophe.h` von der Vorlesungshomepage, indem Sie erst die passenden Makros definieren und danach die Datei einbinden. Sie sollen die Datei `strophe.h` nicht verändern und es dürfen auch keine Warnings generiert werden! Eine Makrodefinition können Sie mit `#undef` entfernen.

7 Write in C (4 Punkte)

Als Parodie von *Let it be* gibt es einen alternativen Liedtext namens "Write in C". Bei Interesse wird jede Suchmaschine den passenden Text liefern.

Hier wollen wir lediglich das Satzstück "let it be" durch "write in C" ersetzen - allerdings nur, wenn das Makro `WRITEINC` definiert ist! Makros können nicht nur in Dateien, sondern auch auf der Kommandozeile via `gcc -DWRITEINC=...` definiert werden. Der Wert kann weggelassen werden und ist standardmäßig 1.

Beispiel zu dieser Aufgabe:

```
$ gcc -E letitbe.c
[...] let it be [...]

$ gcc -DWRITEINC -E letitbe.c
[...] write in C [...]
```

8 Einlesen und Formatierung (15 Punkte)

Schauen Sie sich das Programm `tabelle.c` genau an und erweitern Sie es, damit es einen Anfangswert, eine Schrittweite und ein Ende einliest und zu jeder Zahl im entsprechenden Intervall folgendes ausgibt: Die Zahl selbst zuerst linksbündig mit 2 Nachkommastellen, dann in Exponentialschreibweise mit 4 Nachkommastellen. Danach rechtsbündig die Wurzel der Zahl. Die letzten beiden Felder geben an, ob die Zahl negativ ist und ob sie *betragsmäßig* größer als 100 ist (`man fabs`). Das linke Feld soll linksbündig sein, das rechte rechtsbündig.

Verwenden Sie außerdem statt `ja` und `nein` zwei Makros, mit denen man durch eine einzige Änderung die Antworten z.B. durch `yes` und `no` ersetzen kann.

Achten Sie genau auf eine saubere, bündige Ausgabe! Hier eine Beispielausgabe:

```
$ ./tabelle
Anfangswert: -110
Schrittweite: 33.7
Ende: 140

   Wert    | Exponential|    Wurzel  | Negativ? | gross?
-----------+------------+------------+----------+--------
-110.00    |-1.1000E+02 |        NaN | ja       |     ja
-76.30     |-7.6300E+01 |        NaN | ja       |   nein
-42.60     |-4.2600E+01 |        NaN | ja       |   nein
-8.90      |-8.9000E+00 |        NaN | ja       |   nein
24.80      | 2.4800E+01 |   4.979960 | nein     |   nein
58.50      | 5.8500E+01 |   7.648529 | nein     |   nein
92.20      | 9.2200E+01 |   9.602083 | nein     |   nein
125.90     | 1.2590E+02 |  11.220517 | nein     |     ja
```

Viel Erfolg!