Abteilung für Angewandte Mathematik
Prof. Dr. S. Bartels
M.Sc. C. Palus

28.05.2018

Übung zur Vorlesung

Einführung in die Programmierung

SoSe 2018 – Blatt 6

Abgabe: Briefkästen RZ/E-Mail bis Montag, den 4.6.2018 um 16:00 Uhr

Aufgabe 1 (10 Punkte)**.** Ein NOR-Gatter ist ein Logikgatter mit zwei Eingängen A, B und einem Ausgang Y, zwischen denen die logische Verknüpfung $Y = \neg(A \vee B)$ besteht.
(i) Zeichnen Sie einen Schaltplan für die Realisierung eines NOR-Gatters mittels Transistoren und zeichnen Sie für jede Kombination von Eingangssignalen (also $(0,0)$, $(0,1)$, $(1,0)$ und $(1,1)$) den tatsächlichen Stromfluss in den Schaltplan ein.
(ii) Anwesenheitsaufgabe: Realisieren Sie diese Schaltung im Tutorat auf einer Steckplatine. Vorbereitend dazu finden Sie auf der Vorlesungshomepage Erklärungen zur Funktionsweise der verwendeten Steckplatinen und Bauteile.

Aufgabe 2 (10 Punkte)**.** Ein gebräuchliches Schaltsymbol für ein NOR-Gatter ist das eines OR-Gatters mit angehängtem Kreissymbol. In der Abbildung sehen Sie das Schema eines sogenannten Flipflops aus zwei NOR-Gattern. Es verfügt über zwei Eingänge S (set) und R (reset) und zwei Ausgänge Q und $\neg Q$.

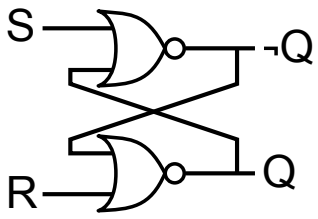

Abbildung 1. Flipflop aus zwei NOR-Gattern.

Welchen Wert hat der Ausgang Q nach einem Signal am S-Eingang, das heißt für $S = 1$, $R = 0$? Was passiert mit dem Ausgangssignal Q nach dem Abstellen des Eingangssignals S, das heißt, wenn jetzt $S = R = 0$ ist? Was passiert danach bei einem Signal am R-Eingang, also wenn jetzt $S = 0$ und $R = 1$ gilt? Wie ändert sich das Ausgangssignal, wenn das Signal am R-Eingang wieder abgestellt wird, also wieder $S = R = 0$ ist? Erläutern Sie die Funktion dieser Schaltung.
Freiwillig: Überlegen Sie sich, wie man ein Flipflop auf einer Steckplatine mit Hilfe von Transistoren und Widerständen realisieren könnte. Zeichnen Sie dazu einen entsprechenden Schaltplan. Bei Interesse können Sie diesen Schaltungsentwurf dann im Tutorat umsetzen.

Aufgaben 3 und 4 auf der Rückseite

Aufgabe 3 (10 Punkte)**.** Schreiben Sie ein Programm, welches die ganzzahligen Einträge einer Matrix mit 4 Zeilen und 3 Spalten per Benutzerabfrage einliest, diese in einem Array `int matrix[4][3]` speichert und anschließend in übersichtlicher Weise in der Konsole ausgibt. Definieren Sie für die Ausgabe eine Unterfunktion `void print_matrix( int matrix[4][3] )`. Die Ausgabe sollte die Zahlen in jeder Spalte rechtsbündig darstellen und in etwa folgendermaßen aussehen:

```
/             \
|   1     2   123 |
|  13   -14    14 |
|  23    23    12 |
|   2    -2     7 |
\             /
```

Hinweis: Um die Ausgabe unabhängig von der Größe der Zahlen übersichtlich zu gestalten, müssen Sie die größte benötigte Spaltenbreite ermitteln. Anschließend können Sie für jeden Eintrag entscheiden, wieviele Leerzeichen ihm voranzustellen sind. Die genaue Umsetzung ist dabei Ihnen überlassen.

Aufgabe 4 (10 Punkte)**.** (Rekursion). Die Programmiersprache C++ erlaubt die Definition von rekursiven Funktionen, das heißt von Funktionen, die sich innerhalb ihres Funktionsrumpfes selbst aufrufen. Dabei ist es wichtig, auf geeignete Rekursionsverankerungen zu achten, um unendliche Rekursionen zu vermeiden. Die folgende Funktion berechnet etwa rekursiv die Summe s_n der ersten n natürlichen Zahlen, wobei die Rekursionsverankerung durch $s_0 = 0$ und der rekursive Aufruf durch $s_n = n + s_{n-1}$ gegeben ist:

```
int rek_summe( int n ) {
  // Rekursionsverankerung
  if ( n==0 ) {
    return ( 0 );
  }
  else {
    // rekursiver Funktionsaufruf
    return ( n + rek_summe( n-1 ) );
  }
}
```

Die Fibonacci-Folge ist eine Folge natürlicher Zahlen, welche nach dem Mathematiker Leonardo Fibonacci benannt ist, der damit im Jahr 1202 das Wachstum einer isolierten Kaninchenpopulation beschrieb. Er ging davon aus, dass jedes Kaninchenpaar pro Monat ein neues Paar Kaninchen wirft. Außerdem nahm er an, dass neugeborene Paare erst in ihrem zweiten Lebensmonat selbst Nachwuchs bekommen. Beginnend mit einem einzigen neugeborenen Paar, bilden die Anzahlen der Kaninchenpaare in jedem Monat die Fibonacci-Zahlen:

$$1, 1, 2, 3, 5, 8, 13, 21, 34 \ldots$$

Die zwei ersten Glieder f_1, f_2 der Fibonacci-Folge sind jeweils 1. Alle weiteren Folgenglieder f_n ergeben sich als Summe ihrer beiden Vorgänger, das heißt es gilt die Regel $f_n = f_{n-1} + f_{n-2}$. Schreiben Sie ein Programm, welches eine natürliche Zahl $n \geq 1$ einliest und das n-te Folgenglied f_n der Fibonacci-Folge berechnet. Implementieren Sie die Berechnung dabei auf zwei Arten: Als Funktion `int fibonacci_vorwaerts( int n )`, welche das n-te Folgenglied mit Hilfe einer Schleife berechnet und als rekursive Funktion `int fibonacci_rekursiv( int n )`.
Testen Sie die Laufzeit Ihres Programms, indem Sie es mit vorangestelltem `time`-Befehl wie folgt aufrufen:

```
$ time ./programmname
```

Vergleichen Sie so die benötigte Zeit jeweils für die iterative und rekursive Berechnung der Folgenglieder $f_{41}, f_{42}, \ldots, f_{46}$.

Abgabe der Programme per E-Mail, (handschriftlich) kommentierte Ausdrucke der Programme und Rechnungen auf gehefteten, mit Namen versehenen Zetteln in die Briefkästen

Homepage zur Vorlesung: `https://aam.uni-freiburg.de/agba/lehre/ss18/einfprog`