SAARLAND UNIVERSITY
COMPUTER SCIENCE

Arbeitsgruppe Programmierung
Jun.-Prof. Dr. Sebastian Hack
Dipl.-Inform. Christoph Mallon

Programmierung 2 — Übungsblatt 10

Abgabe: **Dienstag, 14. Juli 2009, 11.45 Uhr, Geb. E1 3, Briefkasten im EG**
Lösung mit **Name**, **Matrikelnummer** und **Name des Tutors** beschriften!

Aufgabe 10.1: C: Zeiger, Adressen, Zugriffe über Zeiger, Zeigerarithmetik

1. Unter `http://cslibrary.stanford.edu/104/` finden Sie eine kurze Erklärung zum Umgang mit Zeigern als Stop-Motion-Animation.

2. Betrachten Sie folgendens C-Progamm:

```
1  #define RLIMIT 11
2
3  int main(int argc, char* argv[])
4  {
5      double d = 3.0, e = 6.0, rr[RLIMIT];
6      double* dp;
7      dp          = &d;
8      e           = *dp;
9      *dp         = 1.0;
10     *dp         = *dp - 6.5;
11     dp          = &rr[1];
12     *(dp + 5)   = 100.50;
13     *dp++       = 99.0;
14     *++dp       = 50.5;
15     return 0;
16 }
```

   Beantworten Sie folgende Fragen und begründen Sie Ihre Antwort: Welche Werte haben die verwendeten `double`-Variablen nach jeder Anweisungsausführung des Programms? Auf welche Variable verweisen die eingesetzten Zeiger nach jeder Anweisungsausführung des Programms? Notieren Sie jeweils nur die Änderungen.

3. Ergänzen Sie im folgenden Programm die „. . .“-Stellen, so dass das Programm danach folgende Ausgabe zeigt:

   ```
   23 42
   42 23
   ```

   Hinweis: Selbstverständlich soll `swap()` die beiden Zahlen tauschen und nicht einfach nur neue Werte zuweisen, d.h. das Programm soll auch mit anderen Zahlenwerten funktionieren.

```
#include <stdio.h>

void swap(... a, ... b)
{
  ...
}

int main(void)
{
  int x = 23;
  int y = 42;
  printf("%d %d\n", x, y);
  swap(..., ...);
  printf("%d %d\n", x, y);
  return 0;
}
```

4. Implementieren Sie folgenden Funktionen:
   a) `int strlen(char const* str)`, die die Länge einer Zeichenkette zurückgibt.
   b) `char* strcat(char* a, char const* b)`, die die Zeichenkette `b` an die Zeichenkette `a` anhängt. Zurückgegeben wird `a`. Beachten Sie, dass keine neue Zeichenkette produziert werden soll. Es sollen lediglich die Zeichen von `b` an die Zeichen von `a` angehängt werden.

Aufgabe 10.2: UPN-Rechner

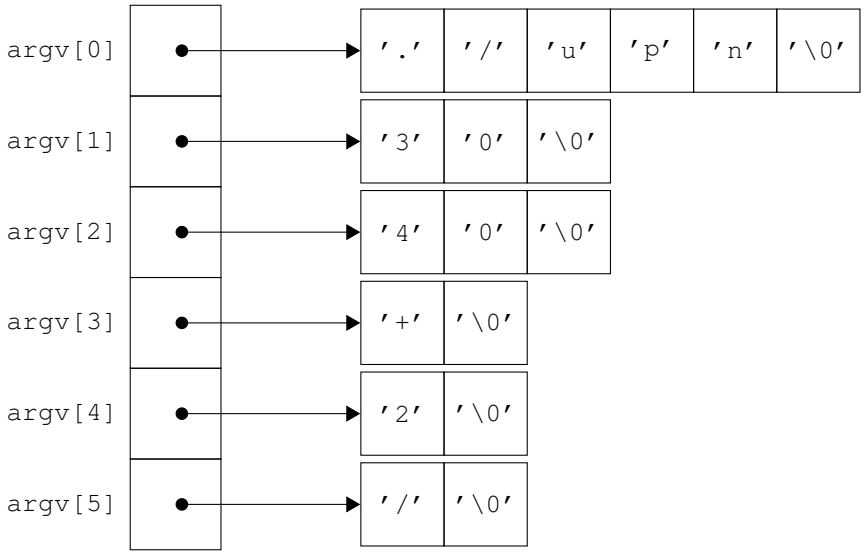
Implementieren Sie einen UPN-Rechner in C. Der Rechner soll wie folgt funktionieren: Der UPN-Ausdruck wird dem UPN-Rechner über die Kommandozeile übergeben. Heißt die vom C-Compiler erzeugte ausführbare Datei `upn`, so soll man den Rechner folgendermaßen aufrufen können:

```
%./upn 30 40 + 2 /
```

Jedes einzelne Element des Ausdrucks (Zahl oder Operator) ist dann ein Kommandozeilenargument. Diese Kommandozeilenargumente werden dem C-Programm vom Betriebssystem übergeben. Sie können über die formalen Parameter von `main()` angesprochen werden.

```
int main(int argc, char* argv[]);
```

Die Reihung `argv` enthält Zeichenketten der einzelnen Argumente. Genauer: `argv` zeigt auf das erste Element einer Reihung von `char`-Zeigern. Jeder dieser `char`-Zeiger zeigt auf das erste Element einer Reihnung von `char`s. Folgende Skizze zeigt die Gestalt der argv-Reihung zu obigem Beispiel:



Hierbei ist zu beachten, dass das nullte Argument immer der Name des Programms ist. Das erste Kommandozeilenargument steht also in `argv[1]`. Die Variable `argc` gibt die Länge der Reihung `argv` an (im obigen Beispiel also 6).
Hinweise:

- Behandeln Sie nur die Operatoren $+$, $-$, $*$, $/$.
- Als Operanden treten nur positive Ganzzahlen auf.
- Verwenden Sie für den Auswertekeller eine Reihung. Fordern Sie den Speicher für diese mittels `malloc()` dynamisch an, so dass der Keller niemals zu klein ist. Geben Sie den reservierten Speicher in jedem Fall wieder frei.
- Schreiben Sie eine Funktion, um die Zahlen von Textform in `int`-Werte umzuwandeln:

  ```
  int str_to_int(char const* text);
  ```

- Fangen Sie Kellerunterläufe (jemand könnte z.B. `20 *` übergeben) ab und verhindern Sie so, dass Sie außerhalb der Reihungsgrenzen auf den Keller zugreifen. Beenden Sie dann Ihr Programm vorzeitig, indem Sie eine Fehlermeldung ausgeben und dann `exit(1)` aufrufen.
- Geben Sie den Wert des UPN-Ausdrucks am Ende mittels `printf()` aus.
- Verwenden Sie die entsprechenden Include-Dateien, d.h. `<stdio.h>` für `printf()` und `<stdlib.h>` für `exit()`.
- Bei manchen Kommandozeileninterpretern (z.B. bei bash oder tcsh) hat `*` eine Sonderbedeutung. Verwenden Sie hier `\*` um `*` an das Programm zu übergeben:

  ```
  %./upn 20 30 + 10 \*
  ```

- Auf den meisten unixoiden Systemen ist ein C-Übersetzer standardmäßig vorhanden. Unter Windows können Sie Cygwin (`www.cygwin.com`) verwenden oder sich mittels SSH auf einen der Pool-Rechner verbinden.

Aufgabe 10.3: Implementierungsabhängig, unspezifiert, undefiniert

Der C-Standard unterscheidet implementierungsabhängiges Verhalten (engl. implementation-defined behavior), unspezifiziertes Verhalten (unspecified behavior) und undefiniertes Verhalten (undefined behavior). Die Bedeutung der Begriffe wird im C99-Standard in Abschnitt 3 definiert[1]. Da C eine sehr systemnahe Sprache ist und dem Programmierer viele Freiheiten lässt, im Gegenzug aber nur wenig Sicherheit vor Fehlern bietet, ist die Kenntnis über diese Begriffe enorm wichtig.

1. Nennen Sie drei Beispiele für implementierungsabhängiges Verhalten und geben Sie an, welches genaue Verhalten für ihren Rechner zutrifft. In der Vorlesung wurden bereits einige Beispiele genannt.
2. Möglicherweise enthält ihr UPN-Rechner eine Zeile folgender Form zur Implemetierung der Subtraktion:

   ```
   push(pop() - pop());
   ```

   Erklären Sie anhand von Abschnitt 6.5, warum bei der Eingabe von

   ```
   %./upn 5 2 -
   ```

   sowohl `3` als auch `-3` standardkonforme Resultate sind. Wie können Sie die Zeile abändern, so dass nur `3` als gültiges Ausgabe auftreten kann?

[1] Eine Entwurfsversion des C99-Standards finden Sie unter `http://www.open-std.org/JTC1/SC22/wg14/www/docs/n1124.pdf`

3. Eine große Falle in C ist undefiniertes Verhalten.
   a) Der Ausdruck `i = i++;` für eine Variable `i` vom Typ `int` besitzt in C undefiniertes Verhalten. Erklären Sie dies anhand von Abschnitt 6.5. Welche definierte Wirkung hat dieser Ausdruck in Java (ausprobieren!).
   b) Gegeben sei eine Reihung `int arr[10];`. Was geschieht bei folgender Verwendung (Vorlesungsfolien zu C Seite 25 und Abschnitt 6.5.3.2):

      ```
      printf("%d\n", arr[10]);
      ```

      Wie verhält sich Java?