Technische Universität München
Institut für Informatik
Dr. rer. nat. Tobias Weinzierl
Matous Sedlacek

März 2012

Kompaktkurs Einführung in die Programmierung

Übungsblatt 3: Ausdrücke

Lernziele

- Operatoren und Operatorpräzedenz.
- Nachvollziehen gegebenen Quellcodes.
- Verzweigungen.

1) Ausdrücke zum Üben (T)

Was ist nach Ausführung der folgenden Anweisungen jeweils der Inhalt der Variablen?

a) Zuweisungen:

```
int x = 3, y = 4, z = 100;
x = y;
y = z;
z = x;
```

b) Zuweisungen und Additionen:

```
int x = 3, y = 4, z = 100;
x = x+y;
y = x+y;
z = x+y;
```

c) Division von `int`:

```
int x = 5, y = 2, z;
z  = x / y;
```

d) Division von `double`:

```
double x = 5.0, y = 2.0, z;
z  = x / y;
```

e) Division gemischt:

```
int x = 5;
double y = 2.0, z;
z  = x / y;
```

f) Division gemischt (2):

```
int x = 5, y = 2;
double  z;
z  = x / y;
```

g) Inkrement und Dekrement:

```
int x = 3, y = 4, z = 100;
y = ++x;
z = x--;
```

h) Rechnen mit `char`:

```
char x = 'A', y, z;
y = ++x;
z = x--;
```

i) Und-Verknüpfung (einfach):

```
int x = 3, y = 4;
if (x > 2 && y > 4) { x = 0; }
```

j) Und-Verknüpfung (böse):

```
int x = 3, y = 4;
if (--x > 2 && ++y > 4) { x = 0; }
```

k) Bedingungs-Operator:

```
int x = 3, y = 4, z = 100;
z = x > y ? x++ : y--;
```

l) Inkrement/Dekrement (2):

```
int x = 2, y = (--x)++;
y = (x++ % 2 == 0) + (--y % 3 != 0);
```

2) Finde Foo (T)

Finden Sie heraus was folgende Funktion `foo(int n)` berechnet. Versuchen Sie den Algorithmus nur mit Hilfe Ihres Kopfes und gegebenenfalls mit Papier und Stift nachzuvollziehen. Abschließend könnten Sie Ihre Vermutung durch Verwendung eines Compilers überprüfen.

```
int foo(int n) {
    int fn, f2 = 0, f1 = 1;
    if (n == 0) return f2;
    if (n == 1) return f1;
    for (int i = 2; i <= n; i++) {
        fn = f2 + f1;
        f2 = f1;
        f1 = fn;
    }
    return fn;
}
```

3) Collatz-Folge (P)

Eine *Collatz-Folge* ist eine Folge (a_i) positiver ganzer Zahlen, bei der man mit einem beliebigen a_1 beginnt und bei der für alle $i \in \mathbb{N}$ gilt

$$a_{i+1} = \begin{cases} \frac{a_i}{2}, & \text{falls } a_i \text{ gerade,} \\ 3a_i + 1 & \text{sonst.} \end{cases}$$

Ergänzen Sie in dem Programm `collatz.cpp` von der WWW-Seite der Vorlesung die Funktion `void collatz1(int a1, int n)` und analog `collatz2`, so dass beide die Folgenglieder $a_1, \ldots, a_n$ ausdrucken;für z.B. `collatz1(7,5)` also 7, 22, 11, 34 und 17. Dabei soll `collatz1` eine Fallinterscheidung mit `if` enthalten, während `collatz2` zum Vergleich den Bedingungs-Operator `?:` verwendet. Der Bedingungsoperator ist somit eine weitere Möglichkeit bedingter Programmausführung. Bezüglich der Vorrangsregeln: versuchen Sie, in dem Ausdruck mit dem Bedingungsoperator so wenig Klammern wie möglich zu setzen!

4) Binärausgabe (★)

Machen Sie sich mit der Darstellung von natürlichen Zahlen $\mathbb{N}$ in Rechensystemen vertraut. (Auch) natürliche Zahlen werden im Dualsystem (zur Basis 2) verarbeitet, d.h., dass eine natürliche Zahl n als Binärfolge

$$a_m 2^m + a_{m-1} 2^{m-1} + \ldots + a_1 2^1 + a_0 = \sum_{k=m}^{0} a_i 2^i, \quad \text{mit} \quad a_i \in \{0, 1\},\ m \in \mathbb{N}$$

interpretiert wird. Schreiben Sie ein Programm welches eine natürliche Zahl einliest und diese zur Basis 2, also in ihrer korrespondierenden Binärdarstellung, auf die Konsole ausgibt. Beispiel:

$$1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 8 + 4 + 0 + 1 = 13_{10} = 1101_2$$