Technische Universität München
Institut für Informatik
Dr. rer. nat. Tobias Weinzierl
Kaveh Rahnema

März 2013

Kompaktkurs Einführung in die Programmierung

Übungsblatt 6: Zeiger

Lernziele

- Initialisieren und Auswerten von Zeigern und Referenzen.
- Referenzieren und Dereferenzieren von (Mehrfach-) Zeigern.
- Rekursion mit Zeigern/Referenzen.

1) Zeiger und Referenzen (1) (T)

Was ist nach Ausführung der folgenden Anweisungen jeweils der Inhalt der Variablen `x` und `y`?

a) Referenzieren und Dereferenzieren:

```
int x = 3, y = 4;
int *p;
p = &x;
*p = *p + y;
```

b) Zeiger zuweisen:

```
int x = 3, y = 4;
int *p1, *p2;
p1 = &x;
p2 = &y;
*p1 = *p1 + y;
p1 = p2;
*p1 = *p1 + y;
```

c) Zeiger auf Zeiger:

```
int x = 3, y = 4;
int *p1 = &x, *p2 = &y;
int **pp = &p1;
(**pp)--;
*pp = p2;
(**pp)++;
```

d) Variablenparameter:

```
int x = 3, y = 4;
zz(&x, &y);
```

mit der Funktion `zz`:

```
void zz(int *a, int *b) {
    int c = *a;
    *a = *b;
    *b = c;
}
```

e) `char` Zeiger:

```
char x = 'a', y = 'k';
char *ppc = &x;
*ppc += 3 ;
y += 2;
```

f) Zeiger und Zeiger:

```
float x = 1.5, y = -0.5;
float *z = &x;
z = &y;
float **ppf = &z;
foo(ppf);
```

mit der Funktion `foo`:

```
void foo(float **pf) {
  **pf *= 4.0;
}
```

2) Zeiger und Referenzen (2) (T)

Gegeben sei folgendes Programm

```
int main()
{
    int a = 2, b = 3, *pi = &a, **ppi, ***pppi = &ppi;
    char c='a', *pc;
}
```

Welche folgender Zuweisungen sind zulässig und welche sind wegen Typ-Konflikten bedenklich bzw. nicht möglich/sinnvoll? Begründen Sie ihre Antworten!

a) `pi = &b;`

b) `pc = pi;`

c) `**pi = &pi;`

d) `pi = &c;`

e) `*pi = *pc;`

f) `***pi = pppi;`

3) Variablenparameter (Satz von Bezout) (P)

Zu zwei ganzen Zahlen a und b gibt es zwei ganze Zahlen n und m, so dass $ggT(a, b) = n \cdot a + m \cdot b$. Bei der Durchführung des euklidischen Algorithmus bekommt man solche Zahlen n und m beinahe geschenkt, daher soll die ggT-Funktion von Blatt 5 um die Berechnung dieser Werte erweitert werden. Da eine Funktion immer nur eine Größe als Ergebnis haben kann, arbeiten wir mit Variablenparametern, so dass die Funktion die berechneten Werte für den ggT, n und m in Variablen der aufrufenden Funktion speichern kann. Gesucht ist also eine Funktion folgender Signatur:

```
void ggT(int a, int b, int *ggt, int *n, int *m)
```

Nach Aufruf von `ggT(a,b,&ggt,&n,&m)` sollen in der Variablen `ggt` der ggT von a und b, und in den Variablen `n` und `m` die oben spezifizierten Zahlen n und m enthalten sein.

Hinweis: Für die (rekursive!) Berechnungsvorschrift für die Zahlen n und m kann die Identität

`c%d == c - (c/d)*d` (für `d != 0`).

benutzt werden. Hierbei stellt `(c/d)` eine ganzzahlige Division dar.

Die Arbeitsweise kann wieder über ein Programmprotokoll erkannt werden. Hier wird wie bei Blatt 5 über einen entsprechenden Zusatzparameter nach Rekursionstiefe eingerückt:

```
|  ggT(72,30) aufgerufen
|  |  ggT(30,72) aufgerufen
|  |  |  ggT(12,30) aufgerufen
|  |  |  |  ggT(6,12) aufgerufen
|  |  |  |  Der ggT ist 6
|  |  |  |  6 = 1*6 + 0*12
|  |  |  6 = -2*12 + 1*30
|  |  6 = 5*30 + -2*72
|  6 = -2*72 + 5*30
```

4) Skalarprodukt (★)

Ziel dieser Aufgabe ist es das Skalarprodukt zweier Vektoren

$$\langle a, b\rangle = a^T b \quad \text{mit} \quad a \in \mathbb{R}^n,\ b \in \mathbb{R}^n \tag{1}$$

beliebig großer Dimension mit Hilfe von OpenMP zu parallelisieren.

i) Implementieren Sie eine mit Hilfe von OpenMP parallelisierte Variante von (1).

ii) Realisieren Sie eine Zeitmessung um die benötigte Zeit zur Berechnung von (1) zu messen. Vergleichen Sie nun Berechnungszeiten für verschiedene Dimensionen von a, b sowie für unterschiedliche Anzahl an Threads. Wieviel Zeit wird benötigt um ein Skalarprodukt von Vektoren der Länge $n = 10^8$ (oder größer?) zu berechnen? Was passiert für noch höhere Dimensionen?

iii) Machen Sie sich mit C++ Exceptions vertraut und fangen Sie eine mögliche Exception ab, die beispielsweise auftritt wenn zu große Vektoren alloziiert werden oder zu wenig Speicher zur Verfügung steht. Mögliche Quelle für Exceptions: `http://www.cplusplus.com/doc/tutorial/exceptions/`