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

März 2013

Kompaktkurs Einführung in die Programmierung

Übungsblatt 11: Vermischtes

Lernziele

- Wiederholung ausgewählter Themen des Kurses.
- Selständigere Umsetzung (Implementierung) von Problemstellungen.
- Realisierung von Benutzereingaben.

1) Zeitumwandlung (Variablenparameter) (P)

Es soll ein Programm geschrieben werden, mit dem eine Zeitangabe „n Minuten“ umgerechnet werden kann in Stunden und restliche Minuten (Anzahl der restlichen Minuten < 60).

- Schreiben Sie eine Funktion `zeitumrechnung` mit einem (Werte-) Parameter `int Zeit`, der mit einer nicht-negativen Anzahl von Minuten besetzt ist. Die Funktion soll diese Zeit in Stunden und restliche Minuten umrechnen und die Ergebnisse in zwei Variablenparametern `int *Stunden` und `int *Minuten` abliefern.
- Schreiben Sie ein Hauptprogramm, das die Funktion `zeitumrechnung` aufruft und die Ergebnisse ausdruckt.

2) Schleifen und Funktionen (P)

In dieser Aufgabe betrachten wir die Funktion

```
void g() {
    int i = 1, s = 0;
    while (s < 5) {
        f(i);
        s = s + i;
        i++;
    }
}
```

die die folgende Funktion `f()` verwendet (`iostream` sei vorausgesetzt):

```
void f(int n) {
    for (int i = 0; i < n; i++)
        std::cout << 'X';
    std::cout << std::endl;
}
```

Schreiben Sie zwei Funktionen `h1()` und `h2()`, so dass wir mit folgender Funktion `gg()` dieselbe Ausgabe erhalten wie mit `g()`.

```
void gg() {
    int i = 1, s = 0;
    while (h1(s)) {
        f(i);
        h2(&i, &s);
    }
}
```

3) Ziffern zählen (P)

i) Schreiben Sie eine Funktion `wieoft()` mit einer Zeichenkette und einem Feld von 10 `int` als Parametern. Die Funktion bestimmt für $0 \leq i \leq 9$, wie oft die Ziffer i in der übergebenen Zeichenkette vorkommt und speichert diesen Wert an die Stelle i des Feldes. *Hinweis:* `s[j]-'0'` ist eine interessante Größe, wenn `s` die Zeichenkette darstellt.

ii) Schreiben Sie eine weitere Funktion `berechne_max()`, die den maximalen Wert der ermittelten Zeichenhäufigkeiten aus a) bestimmt und zurückgibt (die Anzahl der Vorkommen des oder der häufigsten Zeichen).

iii) Schreiben Sie ein geeignetes Hauptprogramm welches Teilaufgaben i) und ii) verwendet und für ein geeignetes Beispiel die Ergebnisse ausdruckt.

4) Test auf Collatz-Folge (P)

Schreiben Sie eine Funktion mit einem Feld ganzer Zahlen a und einer ganzen Zahl n als Parameter, wobei a ein Feld mit n Komponenten sei. Ergebnistyp soll `bool` sein und das Ergebnis soll genau dann `true` sein, wenn das Feld der Anfang einer Collatz-Folge ist.
Das Feld `{13,40,20,10,5,16}` wäre ein Parameter, der `true` als Ergebnis liefern soll.

Hinweis: Zum Lösen dieser Aufgabe ist nicht unbedingt nochmal die Collatz-Aufgabe notwendig. Man kommt alleine mit der Definition der Collatz-Folge aus.

5) Schnittpunkt zweier Geraden (P)

Auf den WWW-Seiten zur Vorlesung finden Sie das Programm `schnittpunkt.cpp`. Eine Gerade, die nicht parallel zur Ordinate ist, ist eindeutig definiert über ihre Steigung m und ihren Achsenabschnitt t mittels der Gleichung

$$y = mx + t.$$

i) Erweitern Sie das Programm so, daß der Schnittpunkt zweier durch den Benutzer eingegebenen Geraden berechnet und gedruckt wird. Sie sollten hierzu in der `main()` Funktion keine weiteren Variablen anlegen. Lediglich Funktionsaufrufe mit den gegebenen Variablen sind nötig.

   Zum Einlesen von Benutzerdaten aus der Konsole können Sie sich mit der C-Funktion `scanf()` z.B. unter `http://www.cppreference.com/wiki/c/io/scanf` vertraut machen.

   Implementieren Sie insgesamt eine Funktion `eingabe()` zum Einlesen der Benutzerdaten, eine Funktion `schnittpunkt_zweier_geraden()` zur Berechnung des Schnittpunktes und eine Funktion `ausgabe()` die die Ausgabe steuern soll.

   Realisieren sie innerhalb `schnittpunkt_zweier_geraden()` verschiedene Rückgabewerte falls

z.B. die Geraden identisch sind. Der Rückgabewert soll in `ausgabe()` über ein `switch()`-Konstrukt spezielle Meldungen hervorrufen.

Hinweise:

- Zum Einlesen können 4 Parameter benutzt werden m_1, t_1, m_2, t_2.
- Da der Schnittpunkt über Funktionsgrenzen hinweg verändert werden soll und sie keine neue Variablen innerhalb `main()` anlegen sollen, werden Sie mit Zeigern bzw. Referenzen arbeiten müssen.

ii) Welche Problemstellung ergibt sich z.B. beim Testen ob die Geraden identisch sind und wie könnte man dem entgegenwirken?

6) Konjugierte Gradienten (★)

Das Verfahren der konjugierten Gradienten (Conjugate Gradients (CG)) ist eine numerische Methode zur Lösung von großen, symmetrisch, positiv definiten, linearen Gleichungssystemen der Form

$$Ax = b. \tag{1}$$

Es gehört zur Klasse der Krylov-Unterraum-Verfahren. Die Lösung x wird dabei iterativ als eine Folge $\{x^{(k)}\}$ über folgenden Ansatz berechnet:

$$\begin{aligned}
p^{(0)} &= r^{(0)} = b - Ax^{(0)} \\
\alpha^{(k)} &= -\frac{\langle r^{(k)}, r^{(k)} \rangle}{\langle p^{(k)}, Ap^{(k)} \rangle} \\
x^{(k+1)} &= x^{(k)} - \alpha^{(k)} p^{(k)} \\
r^{(k+1)} &= r^{(k)} + \alpha^{(k)} Ap^{(k)} \\
\textbf{if } \|r^{(k+1)}\|_2^2 &\leq \epsilon \textbf{ then break} \\
\beta^{(k)} &= \frac{\langle r^{(k+1)}, r^{(k+1)} \rangle}{\langle r^{(k)}, r^{(k)} \rangle} \\
p^{(k+1)} &= r^{(k+1)} + \beta^{(k)} p^{(k)}
\end{aligned}$$

Typische Parameter zur Steuerung der Iteration sind:

- $x^{(0)}$: Das Verfahren benötigt eine Startlösung $x^{(0)}$. Meistens wird $x^{(0)} = (0, 0, 0, \ldots, 0)^T$ oder $x^{(0)} = (1, 1, 1, \ldots, 1)^T$ gewählt.
- ϵ: Ist die 2-norm des Residuums $r = b - Ax$ kleiner als eine Toleranzgrenze ϵ wird die Iteration beendet. Ausreichend ist oft ein Wert von z.B. $\epsilon = 10^{-6}$ oder $\epsilon = 10^{-9}$.
- `k_stop`: Zusätzlich wird meistens ein maximale Anzahl an Iterationen vorgeschrieben. Typisch sind z.B. eine Ausführung von maximal 1000 Iterationen.

i) Machen Sie sich durch weitere Quellen mit dem CG Verfahren vertraut.

ii) Implementieren Sie das CG Verfahren zur Lösung von (1). Ihre Matrix kann z.B. mit Zufallswerten besetzt sein. Alternativ könnten Sie eine Einleseprozedur für Matrizen aus Matrix Market `http://math.nist.gov/MatrixMarket/` benutzen, die aus speziellen Problemstellungen resultieren. Hier finden Sie auch schlecht konditionierte Systeme für die CG unter Umständen nicht

in `k_stop` iterationen konvergiert. Die Matrizen aus Matrix Market sind dünnbesetzt und liegen im row-column-value Format vor.

iii) Schreiben Sie zusätzlich das Residuum jeder Iteration $r^{(k)}$ in eine Datei `residual.dat`. Was können Sie für das Residuum (z.B. für schlecht konditionierte Systeme) beobachten?

iv) Zum Ende Ihrer Berechnung sollten Sie überprüfen ob Ihre finale Iterierte (1) erfüllt.