6. Übung am 29.11.2010
==========

Anleitung:

Während jeder Übung ist von jeder Gruppe ein (kurzes) Protokoll zu erstellen. Das Protokoll ist eine einfache ASCII-Text-Datei, die mit einem Text-Editor (z.B. gedit, kate) mit dem Sie auch Ihre Programme schreiben, erzeugt wird. Nennen Sie diese Datei unbedingt "PROTOKOLL.txt".

Das Protokoll muss Folgendes enthalten:
1. Datum, Übungsnummer, Gruppennummer, Name(n) der mitwirkenden Studierenden,
2. Benötigter Zeitaufwand für die gestellten Aufgaben (circa),
3. Namen der erstellten Programme (KEINE Listings),
4. Kurze Antwort auf eventuell weiter unten gestellte Fragen,
5. Eventuelle Probleme oder Besonderheiten, falls diese aufgetreten sind.

Sämtliche während der Übung erstellten Dateien (Protokoll, Source Codes, ausführbare Programme, etc.) verbleiben im Verzeichnis für den jeweiligen Übungstag, also z.B. "06Ue2010-11-29/" Ihrer Gruppe.

Das Protokoll und die Übungsprogramme sollten am jeweiligen Übungsnachmittag erstellt werden, spätestens jedoch bis zum nächstfolgenden Übungstag (Montag), 14:00! (Spätere Ausarbeitungen können nur in begründeten Fällen und nach Rücksprache mit Ihrem Betreuer bzw. Tutor berücksichtigt werden!)

==========

Aufgaben zu Kapitel 6 (3 Punkte):

1. Schreiben Sie ein Programm, das den Wert eines Polynoms an der Stelle x0 berechnet. Verwenden Sie dazu das HORNER-Schema (1 Punkt):

   Das Polynom p(x)=x^4 + 3x^3 - 7x^2 + 4x + 2 kann auch wie folgt dargestellt werden:

   p(x)= x(x(x(x+3)-7)+4)+2

   Man kann dies zur effizienten Berechnung von p(x0) benutzen.

   Vorgangsweise:

   1. Speichern Sie die Koeffizienten des Polynoms in einem Array.
   2. Implementieren Sie den Horner-Algorithmus

      ```
      y=k[N];
      for (i=N-1; i>=0; i--) y=x*y+k[i];
      ```

   3. Vergleichen Sie das Ergebnis mit dem "einfachen" Algorithmus, der einfach die Potenzen aufsummiert.

      ```
      for (i=0,y=0;i<=N; i++) y+=k[i] * pow(x,i);
      ```

   Strukturieren Sie Ihr Programm in Unterprogramm (Funktion) + Hauptprogramm:

   ```
   double Horner(double *Koeff, int Length, double x);
   ```
   Berechnung des Polynoms an der Stelle x

   Das Hauptprogramm wird dann wesentlich einfacher und könnte im Prinzip etwa so aussehen:

```
double k[]={2,4,-7,3,1};
double x=5.,y;

y=Horner(k,sizeof(k)/sizeof(double),x);
printf("y(%f)=%f\n",x,y);
```

Frage: Warum ist das Horner-Schema effizienter als der einfache Algorithmus? (1 Punkt)

2. Verallgemeinern Sie Ihr bereits vorhandenes Programm zur Nullstellensuche (4. Übung), sodass es auf beliebige Funktionen anwendbar ist (1 Punkt):

   Formulieren Sie den Algorithmus zur Nullstellensuche in einer eigenen Funktion,

   z.B. NStelle(...),

   und übergeben Sie die konkrete Funktion, deren Nullstelle bestimmt werden soll, mittels Zeiger an die Funktion NStelle.

   Testen Sie Ihr Programm mit verschiedenen Funktionen (z.B. Polynomen).