Vorlesung 6

Allgemeine Relativitätstheorie mit dem Computer

PC-Pool Raum 01.120 Johann Wolfgang Goethe Universität 23. Mai, 2016

Matthias Hanauske

Frankfurt Institute for Advanced Studies
Johann Wolfgang Goethe Universität
Institut für Theoretische Physik
Arbeitsgruppe Relativistische Astrophysik
D-60438 Frankfurt am Main
Germany

Allgemeines

Ort und Zeit:

PC-Pool Raum 01.120, immer Montags von 16.15 bis 17.45 Uhr
Zusätzlicher, freiwilliger Übungstermin 15.00 bis 16.15 Uhr

Vorlesungs-Materialien und *Lon Capa* Online-Lernplatform:

http://th.physik.uni-frankfurt.de/~hanauske/VARTC/

http://lon-capa.server.uni-frankfurt.de/

Plan für die heutige Vorlesung:

Wiederholung: Herleitung und numerische Lösung der Tolman-Oppenheimer-Volkoff Gleichung (Innenraum Lösung der Schwarzschildmetrik) mit Maple.

Metrik-Komponenten innerhalb und außerhalb eines Neutronensterns. Masse-Radius Diagramm einer Sequenz von Neutronensternen. Maximale Masse eines Neutronensterns.

Berechnung der Eigenschaften von weißen Zwergen.

Kurze Einführung in C++. Numerische Lösung der TOV-Gleichung in C++.

Wir gehen hier zunächst zu Teil II über, werden dann jedoch zu den noch nicht behandelten Themen des Teil I zurückkommen.

C++ Grundgerüst und Variablen

Einlesen von Header-Files (Definition nötiger C++ Funktionen)

Beginn des Hauptprogramms

Ausgabe eines Strings

```
#include <iostream>                    //Ein-/Ausgabe (Include-Dateien)
using namespace std;                   //Fuer cout

main(void)  //Hauptprogramm
  {

  cout<<"Hello \n";

  }
```

Prog1.cpp

Deklaration einer Integer (natürliche Zahl) und einer double (reelle Zahl) Variable

Variablen bekommen einen festen Zahlenwert (Initialisierung)

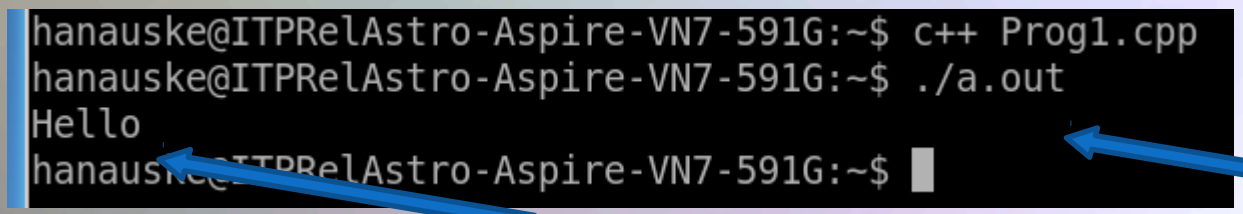
Ausgabe des Wertes der Variablen

```
#include <iostream>                    //Ein-/Ausgabe (Include-Dateien)
using namespace std;                   //Fuer cout

main(void)  //Hauptprogramm
  {
    //Variablendeklarationen
    int i;
    double a;

    //Variableninitialisierung
    i=3;
    a=1.435553;

    cout<<"i="<<i<<"\n";
    cout<<"a="<<a<<"\n";

  }
```

Vom Quellcode zum ausführbaren Programm

Der Quellcode (z.B. Prog1.cpp) muss compiliert werden um ein ausführbares Programm (a.out) zu erzeugen. Man öffnet hierzu in dem Verzeichnis in dem sich der Quellcode befindet, ein Terminal und führt das folgende Kommando aus:

c++ Prog1.cpp



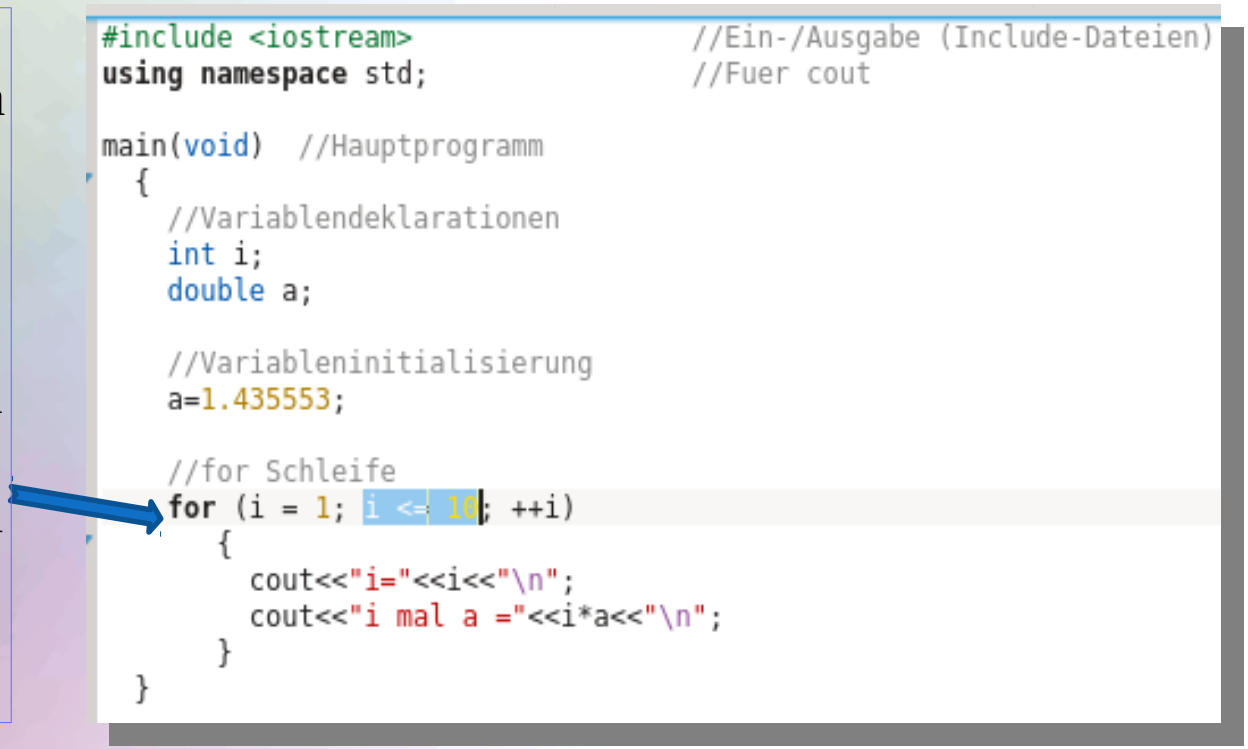
Das Programm wird gestartet und erzeugt im Terminal die Ausgabe "**Hello**"

Beim Compilierungsprozess wird eine Datei (a.out) erzeugt, die man dann mittels des folgenden Kommandos ausführen kann:

./a.out

C++ Die for-Schleife

Mittels einer for-Schleife können iterative Aufgaben im Programm implementiert werden. Die for-Schleife benötigt einen Anfangswert (i=0), die Angabe wie lange sie die Interation dürchführen soll (i<=10) und die Angabe um wieviel sie die Variable in jedem Schritt verändern soll (i++). i++ ist nur eine Kurzschreibweise von i=i+1.



```
#include <iostream>                    //Ein-/Ausgabe (Include-Dateien)
using namespace std;                   //Fuer cout

main(void)  //Hauptprogramm
  {
    //Variablendeklarationen
    int i;
    double a;

    //Variableninitialisierung
    a=1.435553;

    //for Schleife
    for (i = 1; i <= 10; ++i)
       {
         cout<<"i="<<i<<"\n";
         cout<<"i mal a ="<<i*a<<"\n";
       }
  }
```

C++ Die do-Schleife

Mittels einer do-Schleife können iterative Aufgaben im Programm implementiert werden. Die do-Schleife benötigt lediglich eine Abbruchbedingung (while(i<=10);) wobei im Inneren der Schleife die Variable i in jedem Schritt verändert werden muss (i++;). Die Variable I muss jedoch zunächst außerhalb der Schleife initialisiert werden (i=1;).

```
#include <iostream>                    //Ein-/Ausgabe (Include-Dateien)
using namespace std;                   //Fuer cout

main(void)  //Hauptprogramm
  {
    //Variablendeklarationen
    int i;
    double a;

    //Variableninitialisierung
    i=1;
    a=1.435553;

    //do Schleife
    do
      {
        cout<<"i="<<i<<"\n";
        cout<<"i mal a ="<<i*a<<"\n";
        i++;
      }
       while(i <= 10);
  }
```

C++ Lösen der TOV-Gleichung

```
#include <iostream>                    //Ein-/Ausgabe (Include-Dateien)
#include <math.h>                      //Mathematisches
using namespace std;                   //Fuer cout

//Definition der Zustandsgleichung
double eos(double p)
  {
    double e;
    e=pow(p/10,3.0/5);
    return e;
  }

main(void)  //Hauptprogramm
  {
    //Variablendeklarationen
    double M,p,e,r,dM,dp,de,dr;
    double eos(double);

    //Variableninitialisierung
    M=0;
    r=pow(10,-14);
    p=10*pow(0.0005,5.0/3);
    dr=0.000001;

    //do-while Schleife (Numerische Lösung der TOV-Gleichung)
    do
      {
        e=eos(p);                                   //Wert der Energiedichte bei momentanen Druck
        dM=4*M_PI*e*r*r*dr;                         //Massenzunahme bei momentanem r und Schrittweite dr
        dp=-(p+e)*(M+4*M_PI*r*r*r*p)/(r*(r-2*M))*dr;  //Druckzunahme bei momentanem r und Schrittweite dr (TOV-Gleichung)
        r=r+dr;                                     //momentaner Radius des Neutronensterns
        M=M+dM;                                     //momentane Masse des Neutronensterns innerhalb des Radius r
        p=p+dp;                                     //momentaner Druck des Neutronensterns innerhalb des Radius r
      }
      while(p>0);

    //Ausgabe der Masse und des Radius auf dem Bildschirm
    cout<<"Neutronensternradius [km]           = "<<r<<"\n";
    cout<<"Neutronensternmasse [Sonnenmassen] = "<<M/1.4766<<"\n";

    return 0;                                       //main beenden (Programmende)
  }
```

Die polytrope **Zustandsgleichung** ist als eine Funktion außerhalb des Hauptprogramms definiert

Deklaration der nötigen **Variablen** und der Zustandsgleichungsfunktion

Festlegung der **Anfangswerte** im Sternzentrum (M,r,p) und der Radiusschrittweite dr

TOV-Gleichungen

Ausgabe auf dem Bildschirm