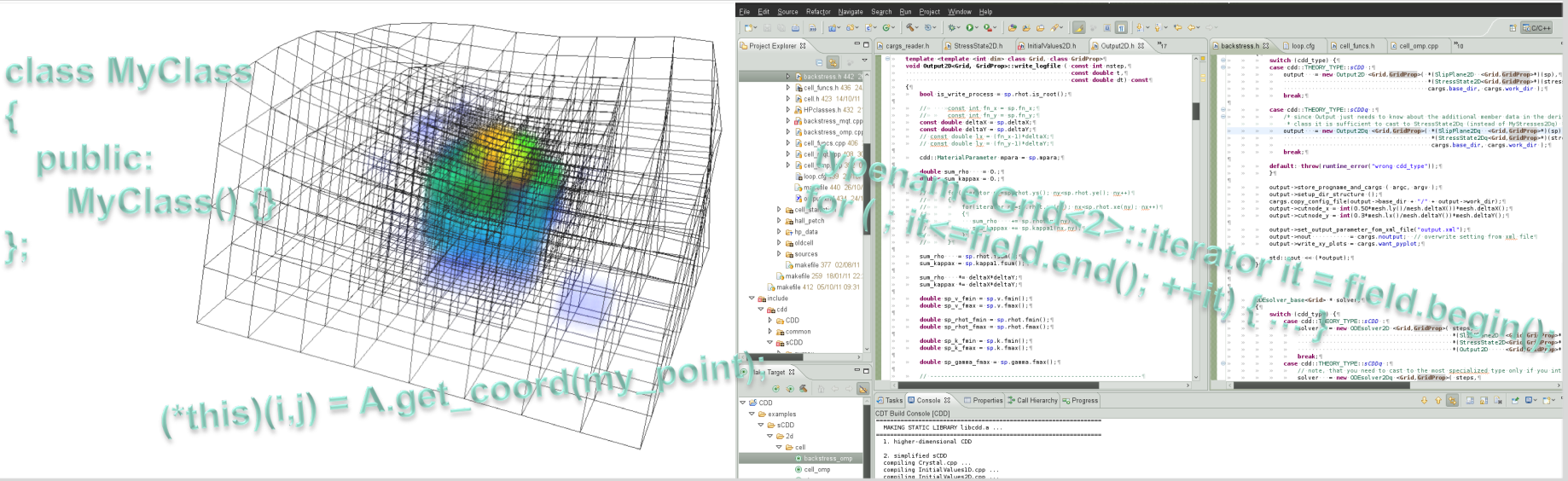
Karlsruher Institut für Technologie

Wissenschaftliches Programmieren für Ingenieure

A. Trenkle, Dr. D. Weygand, T. Achkar, Dr. Ing. M. Stricker

Übung 2:
Gültigkeitsbereiche von Variablen; Funktionen; Referenzen; Fibonacci-Folge, Monte-Carlo Integration, Typumwandlung

Institute for Applied Materials – Computational Materials Science (IAM-CMS)
KIT – The Karlsruhe Institute of Technology – University of the State of Baden-Wuerttemberg, GERMANY



Gültigkeitsbereich von Variablen

Karlsruher Institut für Technologie

Todo | Beantworten von Fragen zum Gültigkeitsbereich von Variablen

- Laden Sie das File `gueltigkeitsbereich.cxx` von ILIAS herunter und beantworten sie folgende Fragen anhand des Beispielprogramms:
  - Welche Gültigkeitsbereiche für Variablen gibt es?
    - Wie kann der Gültigkeitsbereich einer Variablen eingeschränkt werden?
    - Welchen Gültigkeitsbereich haben jeweils die Variablen `a,b,c` im Beispielprogramm?
    - (Was passiert mit Variablen nach Ende des Gültigkeitsbereichs?)
  - Was passiert bei Namensgleichheit von globalen und lokalen Variablen?
    - Wie kann in einem solchen Fall gezielt auf globale Variablen zugegriffen werden?
    - Kommentieren Sie Definitionen von Variablen in den einzelnen Blöcken des Programms aus: Wie ändert sich dadurch das Verhalten des Programms?

Referenz und Pointer

Info | Zusammenfassung der Vorlesung

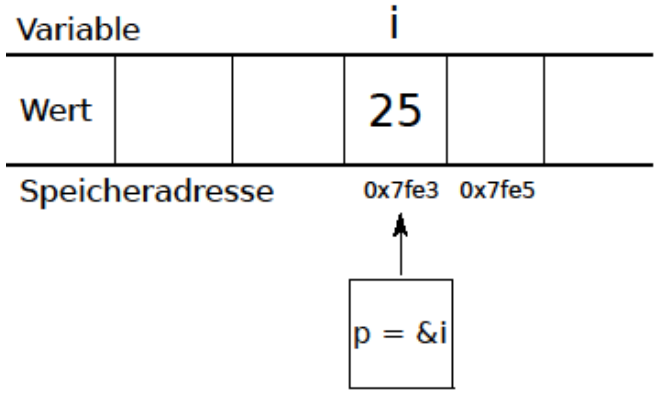
- **Referenzen**: Alias, also ein anderer Name für eine bereits existierendes Objekt (Variable)

```
int i = 4;
int& r = i; // r ist Referenz auf i
r = 5; // i ändert sich mit
cout<<"i: "<<i<<" r :"<<r<<endl;
```

- Mit Referenzen können übergebene Variablen verändert werden, Vermeidung von Kopien

- **Pointer** (Zeiger) zeigen auf die Speicheradresse einer anderen Variablen

```
int i = 25;
int* p; // p ist Pointer
p = &i; // Wert von p ist die Adresse von i
cout<< <<"Adresse: "<<p
<<"Speicherwert: "<<*p<<endl;
```



Vorbereitung Funktionen

Karlsruher Institut für Technologie

Todo

Ziel: Übergabearten von Variablen üben

- File: `main_function_vorlage.cpp` auf ILIAS Server
- Schreiben Sie Funktionen (fall_i) mit Rückgabetyp `void` und einem Argument: Die Funktionen (Argumente) sollen folgenden Anforderungen genügen:
  - **Fall_1:** Argument soll nur übergeben werden; lokale Variable in Funktion soll aber veränderlich sein
    - Wie nennt man diese Art der Übergabe?
  - **Fall_2:** Argument soll übergeben und ein Veränderungen der Variablen in der Funktion soll für das Hauptprogramm sichtbar sein
    - Wie nennt man diese Art der Übergabe?
  - **Fall_3:** Adresse soll übergeben werden:
    - **1:** Inhalt des Objektes soll veränderbar sein; Zeiger soll nicht veränderbar sein
    - **2:** Inhalt des Objektes soll nicht veränderbar sein; Zeiger soll nicht veränderbar sein.
    - **3:** Beides soll veränderbar sein

Fibonacci-Folge

Info

- Die Fibonacci-Folge kann in vielen Vorgängen in der Natur beobachtet werden (Vermehrung von Kaninchen, eindimensionaler Quasikristall, Anordnung von Blüten,...)
  - F(n) ist die n-te Fibonacci Zahl
  - F(0)=0 und F(1)=1 sind die Startwerte
  - Die n+1-te Fibonacci Zahl: F(n+1)=F(n)+F(n-1)

Todo

Programm zur Berechnung der Fibonacci-Folge

- Schreiben Sie ein Programm, unter Verwendung einer rekursiven Funktion, das die n-te Fibonacci Zahl berechnet, wobei `n` eine Eingabe ist. 8ung!: Rekursion für große n sehr langsam. Es ist auch eine Lösung unter Verwendung von Schleifen (for) möglich.
  - Berechnen Sie den Quotienten zweier aufeinanderfolgender Fibonacci Zahlen: `F(n+1)/F(n)`
  - Schreiben Sie das Ergebnis in eine Datei mit folgendem Format
    `n F(n) F(n)/F(n-1)`
  - Tragen Sie mit gnuplot n gegen F(n)/F(n-1) auf: konvergiert der Quotient? Kennen Sie den Wert?
  - Berechnen Sie die Summe der ersten 10 Fibonacci Zahlen.

Hinweis Fibonacci Folge:

Info | Arrays aus STL verwenden

- Falls Sie ein Array/Vektor benötigen, verwenden Sie bitte die STL Bibliothek
  Es geht auch ohne Vektoren!
- STL Bibliothek (C++11: `array<Datentyp,Zahl der Einträge>`)
  http://www.cplusplus.com/reference/array/

```
#include <iostream>
#include <array>
using namespace std;

int  main(){

  array<int,10> feld;  // reserviert ein fixed Feld mit 10 Einträgen

   for(int i=0;i<feld.size();i++){  // .size() Methode, die die Anzahl der Elemente zurückgibt
       feld[i]=9*i;
   }

  for(auto i:feld) {  // alternative Schreibweise, um über
                      // die Elemente (Werte) des Feldes zu laufen
       cout<<"feld["<<i<<"]"<<feld[i]<<endl;
   }
}
```

Fliesszahlen: Grenzen / Grenzwert

- Gegeben sei die Funktion: $f(x) = \frac{\sqrt{16+x}-4}{x}$.
- Berechnen Sie mit Hilfe der Regel von de l‘ Hôpital den Grenzwert der Funktion $\lim_{x \to 0} f(x)$
- Schreiben Sie ein C++ Programm, das den Grenzwert numerisch berechnet: Bestimmen Sie hierzu den Verlauf der Funktion $f(x)$ im Intervall $[a, b]$ wobei $a > 0$ und $b > a$ gilt.
  - Verwenden Sie dabei verschiedene Fließzahlgenauigkeiten (z.B. mit Hilfe von `typedef` => siehe Vorlesungsfolien; verwenden sie zunächst doppelte Genauigkeit: `double`)
  - Skizzieren Sie den Programmaufbau:
    - Eingabeparameter: Intervallgrenzen, Stützstellen
  - Übersetzen Sie das Programm
  - Lassen Sie das Programm für sinnvolle Intervallgrenzen laufen (z.B. $[a, b] = [1e-20, 1e-10]$ und mind. 1000 Stützstellen
  - Tragen Sie mit Hilfe von `gnuplot` das Ergebnis aus. Den Wertebereich der Ausgabe können Sie mit Hilfe des folgenden Kommandos einstellen:
    ```
    gnuplot>  pl [0:1e-11][0.12:0.13] "erg.dat"  w l
    ```
    der Wertebereich ist nun $x \in [0{,}10^{-11}]$ und $x \in [0.12{,}0.13]$
    - Was beobachten Sie? Welchen Wert erhalten Sie?
    - Wie verändert sich das Konvergenzverhalten, wenn Sie für alle Fließzahlen im Programm einfache Genauigkeit verwenden?
    - Welche Schlussfolgerungen ziehen Sie aus den Beobachtungen?
    - Können Sie das Problem umgehen?

Integration: Monte Carlo Verfahren

Zusatzaufgabe

Numerische Integration mit Monte Carlo Verfahren

Berechnen Sie den Flächeninhalt des Einheitskreises numerisch mit Hilfe des Monte Carlo Verfahrens

- **Monte Carlo Integration:**
  - Variablen N, M benötigt, wobei N die Anzahl der gezogenen Zufallszahlen ist und M die Zahl der Zufallszahlen innerhalb des Kreises ist
  1. Gebiet ist durch (x_0, x_1) und (y_0, y_1) geben: Gebietsgröße $(x_1 - x_0) * (y_1 - y_0)$
  2. Ziehen Sie eine Zufallszahlenpaar in diesem Gebiet
  3. wenn Zufallszahl innerhalb des Einheitskreises liegt: erhöhen Sie M um eins.
  4. Gehe zu (2.), wenn noch nicht N Zufallszahlenpaare gezogen wurden

  $\Rightarrow$ Ergebnis: Flächeninhalt = M/N * Gebietsgröße

Info

Zufallsgenerator `rand()` aus `cstdlib`

- `#include <cstdlib>`
  - `int value = rand()` gibt Zufallszahlen zwischen `0` und `RAND_MAX`
  - `RAND_MAX` in Bibliothek definiert (8tung integer)