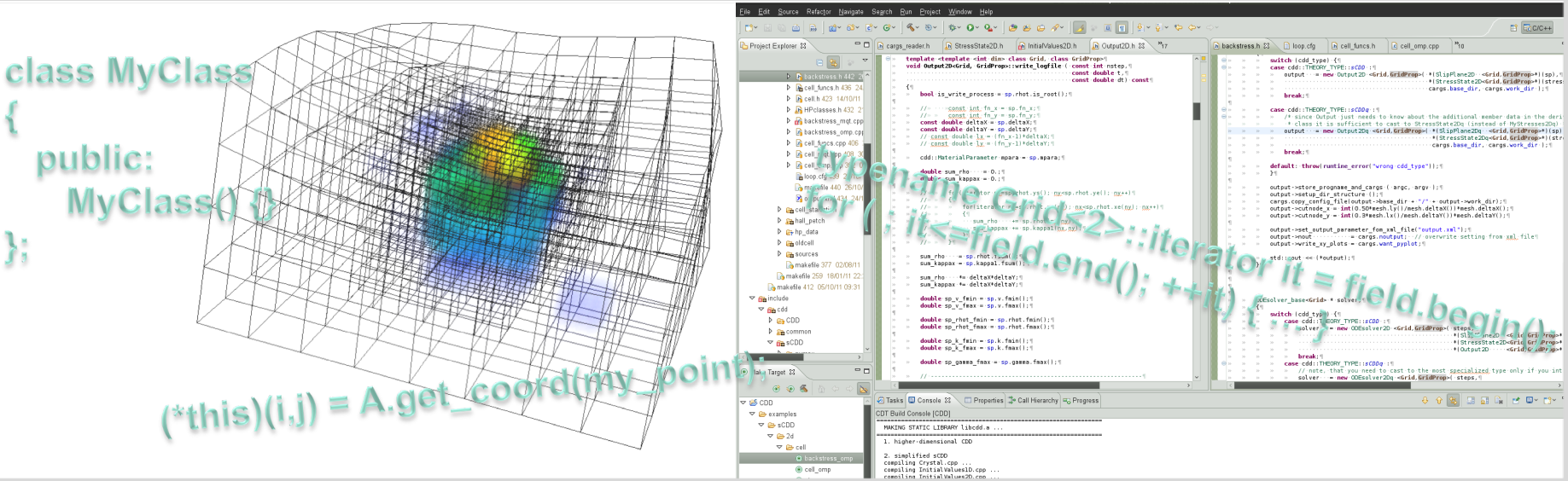
Karlsruher Institut für Technologie

Wissenschaftliches Programmieren für Ingenieure

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

Übung 7:
Lambda Funktion, OpenMP Parallelisierung und Mandelbrot-Menge

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



Lambda Funktionen

Info

- Lambda Funktionen sind namenlose Funktionen, die vor Ort definiert werden, z.B. als Argument in einem STL Algorithmus
(siehe Vorlesungsfolien)

Todo

- Schreiben Sie ein kurzes Programm, das in einem STL Container (z.B. vector< > oder deque<>) Objekte der Struktur data abspeichert.

```
struct data{
    unsigned int val1;
    int val2;
};
```

- Füllen Sie den Container mit 10 Elementen und Initialisieren Sie die Komponente `val1` mit dem Index des Eintrags und `val2` durch Zufallszahlen (`rand()`). Verwenden Sie jeweils eine Initialisierungsliste.

Lambda Funktionen

- Geben sie alle Einträge des Containers auf der Konsole aus, wobei Sie folgende Möglichkeiten des STL nutzen sollen
  - Verwendung von `for_each()` (STL <algorithm> Bibliothek)
  - Lambda Funktion für Ausgabe
- Sortieren Sie die Elemente innerhalb des Containers nach der Größe des Wertes der Komponente `val2`.
  - Verwenden Sie `sort()`
  - Schreiben Sie die entsprechenden Vergleichsbedingung als Lambdafunktion. Diese wird dem Sortieralgorithmus als Argument übergeben.
  - Geben Sie das Ergebnis auf der Konsole aus.
- Veränderung der Komponente `val2`:
  - Schreiben Sie eine Lambdafunktion, die jeweils den Wert von val2 auf val1*val1 setzt.
  - Geben Sie das Ergebnis auf der Konsole aus.

Lambda Funktionen: capture

Info

- In der [] Klammer werden die Regeln festgelegt, welche Zugriffsmuster auf Umgebungsvariablen zugelassen werden (siehe Vorlesungsfolien).

Todo

- Definieren Sie im Hauptprogramm eine Variable: `int val {42};`
  Aufgabe (A)
  - Weisen Sie der Komponente `val2` aller Elementen des Kontainers den Werte der Variablen `val` zu
  - Geben Sie das Ergebnis auf der Konsole aus.

  Aufgabe (B)
  - Erhöhen Sie nach jeder Zuweisung nun den Wert von `val` um eins. (geht dies unmittelbar? Welche Möglichkeiten haben Sie?)
  - Geben Sie das Ergebnis auf der Konsole aus.
  - Geben Sie auch den Wert von `val` aus;

Parallelisierung: OpenMP

Info

- Mit OpenMP können Sie Programmabläufe parallelisieren (siehe Vorlesung)
  - `omp.h` ist die Headerdatei / Schnittstelle zu den OpenMP Funktionen
  - Mit `export OMP_NUM_THREADS=2` auf der **Konsole** können Sie die Anzahl der verwendeten Threads festlegen (im Beispiel zwei Threads)
  - `#pragma omp parallel {}` definiert einen parallelen Block
  - Verschiedene Anweisungen wie z.B. `atomic, private, critical` steuern die Parallelisierung
  - Mit `omp_get_thread_num()` kann die Nummer des aktuellen Threads ausgelesen werden
  - Mit `omp_get_wtime()` kann die aktuelle Zeit gemessen werden: Nützlich für Laufzeitmessungen
  - Zum Kompilieren benötigen Sie die Compiler-Option `–fopenmp` (Beispiel: `g++ -fopenmp datei.cxx`)

Grundlagen der Parallelisierung mit OpenMP

Karlsruher Institut für Technologie

Todo

- Im Ordner `vorlesung` finden sich Beispiel-Programme für OpenMP Anweisungen:
  - `omp_parallel.cxx`
  - `omp_atomic.cxx`
  - `omp_critical.cxx`
  - `omp_private.cxx`
  - `omp_firstprivate.cxx`
  - `omp_reduction.cxx`
  - `omp_for.cxx`

- Kompilieren Sie die Programme mit der Compiler-Option `–fopenmp` **direkt in der Konsole**.
- Testen Sie die Programme mit/ohne den verschiedenen Anweisungen:
  - Was bewirken die Anweisungen?
  - Messen Sie die Zeit für unterschiedliche Anzahlen von Threads!

Anwendung OpenMP: Matrix/Vector Klassen

Todo

- Verwenden Sie die Klassen in **`solver_loesung.tgz`** aus Übung_05.

- 1. Parallelisieren Sie:

  - a) die **Matrix/Vector Multiplikation** in der **Matrix-Klasse**
  - b) das **Skalarprodukt** in der **Vector-Klasse**

  Siehe Quellcode

- 2. Vergleichen Sie die Laufzeiten des CG-Solvers für das Wärmeleitproblem seriell/parallel.

Bei genügend hoher Anzahl von Stützpunkten N skaliert der Löser gut:
Beispiel: N=500 1Thread 0,33s ; 4 Threads 0,09s (ca +-10% Streuung)

Anwendung OpenMP: Mandelbrot-Menge

Karlsruher Institut für Technologie

Info

- Die Mandelbrot-Menge ist die Menge aller komplexen Zahlen C, für welche die rekursive Folge
  $Z_0 = 0,$
  $Z_{n+1} = Z_n^2 + C$
  beschränkt (<2) ist.
- Eine visuelle Darstellung erfolgt in der komplexen Zahlenebene:
  `x: Realteil(C), y: Imaginärteil(C), Wert: n`

Todo

- Kompilieren Sie das Beispielprogramm `mandelbrot.cxx` und führen es aus.
- Plotten Sie das Ergebnis mit Gnuplot:
  - `set pm3d map`
  - `set size ratio 1`
  - `spl “mandel.dat“`
- Überlegen Sie, wie das Programm sinnvoll parallelisiert werden kann:
  - Testen Sie das Programm an kleinen Systemen auf Richtigkeit.
  - Vergleichen Sie die serielle/parallele Laufzeit für unterschiedliche Systemgrößen.