0x0a – Übungen

’Roarport’

Lambda-Ausdruck

Schreiben Sie jeweils einen Lambda-Ausdruck, der

- drei übergebene double-Werte addiert und zurückgibt;
- testet, ob ein übergebener int-Wert in einem Intervall [a,b] liegt (dabei sind a und b lokale Variablen);
- keine Argumente hat, aber eine lokale int-Variable z auf -z setzt (akademisches Bsp.).

0x0a – Übungen

’Kreley’

Funktionszeiger, function

Definieren Sie einen Funktionszeigerdatentyp für eine Funktion, die zwei ints x,y bekommt und einen double zurückgibt.

- Implementieren Sie eine (klassische) Funktion Q, die aus übergebenen Werten x und y den Wert x/y berechnet.
- Initialisieren Sie einen Funktionszeiger mit Q und rufen Q über diesen auf.
- Verwenden Sie statt des Funktionszeigertyps das Template function.

Erweiterung:

- Definieren Sie einen Funktionszeiger direkt, d.h. analog zu oben aber ohne Verwendung Ihrer Typdefinition und ohne das Template function.

0x0a – Übungen

'Yrouwood'

Funktionszeiger, lambda-Ausdruck

Schreiben Sie eine Funktion approx, die als Parameter einen Startwert x0, eine zu iterierende Funktion f, und ein eps erhält:

- approx berechnet x=f(x), mit x=x0 zu Anfang, solange, bis der neue x-Wert sich vom vorhergehenden um weniger als eps unterscheidet.
- Geben Sie den letzten berechneten Wert zurück.
- Testen Sie approx für das Heron-Verfahren (Wurzel- Berechnung, siehe Wikipedia; zunächst für feste Werte a).
- Nutzen Sie einmal Funktionszeiger und einmal das Template function in der Definition von approx.
- Rufen Sie approx sowohl mit Funktionszeigern als auch mit Lambda-ausdrücken auf.

0x0a – Übungen

’Pleim’

Algorithmen

Generieren Sie einen Vektor mit den Werten (n über k) für festes n, also z.B. für n=4: [1,4,6,4,1].

- Nutzen Sie, abgesehen von der Ausgabe, nur Funktionen der Algorithmen-Bibliothek, also z.B. for_each oder generate und keine expliziten for(i=0;…) Schleifen.
- Verwenden Sie den Datentyp vector.

Erweiterung:

- Generieren Sie so das Pascalsche Dreieck bis zu einer Dimension n in einem vector von vector.