Höhere Programmierkonzepte
Übung II

Prof. Dr. Nikolaus Wulff

Zum 06. Jan. 2020

```
1
2 void* maximum(int len, void* array, size_t size,
3                int (*cmp)(const void*, const void*));
```

Listing 1: Prototyp einer C Maximum-Funktion.

1 C void* Zeiger Arithmetik

Das Listing (1) zeigt den Prototyp einer generischen Maximum-Funktion, die das größte Element eines Feldes zurückliefert. Das Problem ist ähnlich zum `sort`-Algorithmus: Da diese Funktion generisch für alle möglichen Datentypen funktionieren soll – nicht nur für numerische Typen wie `float` oder `int` –, werden die Elemente ganz allgemein strukturlos als `void*` modelliert und die Vergleichsrelation per Funktionszeiger `cmp` vom aufrufenden Programm zur Verfügung gestellt.

Aufgabe

1. Implementieren Sie obige, allgemeine `maximum`-Funktion in C.
   - Überlegen Sie, wie sich ein Iterator über das Feld realisieren lässt, da es für `void*` Zeiger keine Pointer-Arithmetic gibt. Zu diesem Zweck wird das `size` Argument an die Funktion übergeben, um geeignet die nächste Position im Feld ermitteln zu können.
   - Testen Sie Ihre C Implementierung für die Datentypen `double` und für Elemente einer Struktur `Color`, die drei RGB-Werte enthält, durch Bereitstellung entsprechender Vergleichsoperationen.
2. Wie sieht eine analoge Schnittstelle für Java mit/ohne Generics aus?
3. Wie lässt sich die in 2. definierte Schnittstelle durch eine λ-Funktion in Java implementieren - vorausgesetzt 2. ist ein *@FunctionalInterface*?

2 Java/C++ Filter mit λ-Ausdruck

```
#include <iostream>
#include <algorithm>
#include <vector>

void print(double v) {
        cout << v << ' ';
}

void for_each_Test() {
        vector<double> list;
        // fill the vector ...
        for(double x=1; x<=5; x++) {
                list.push_back(x);
        }

        // generic for-each loop with functor
        for_each( list.begin(), list.end(), print);
}
```

Listing 2: Standard Template Library Beispiel eines C++ Functors.

Das Listing (2) zeigt die Befüllung eines generischen Vectors der C++ STL[1], der in einer `for`-Schleife befüllt wird und anschließend per *for_each* ausgegeben wird. *for_each* ist kein C++ Schlüsselwort, sondern ein Algorithmus der Standard Template Bibliothek. Als drittes Argument wird ein C++ Functor, ein Zeiger auf die Methode `print` übergeben.

Aufgabe

- Studieren Sie die C++ STL und machen Sie sich mit dem *for_each* Konstrukts für die Verarbeitung von generischen Mengentypen vertraut.
- Implementieren Sie vollkommen analog zu C++ ein *for_each* Konstrukt in Java, dem auch eine Art Java Functor übergeben werden kann, als einfaches Beispiel mag auch hier die print Ausgabe genügen.
- Sehen Sie zusätzlich einen Filter vor, so daß z.B. eine Bedingung erfüllte werden muss bevor die Ausgabe erfolgt.
- Entwickeln Sie einen Filter, der bei numerischen Collection, alle Elemente größer einer vorgegebenen Zahl innerhalb der *for_each* Schleife ausgibt.

[1] STL ≡ Standard Template Library, die Entsprechung der Java *java.util* Collection Klassen und den *java.util.function @FunctionalInterfaces*.