Database Systems II – Exercise #7

Sheet #7: Lambda Expressions, Physical Algebra Implementation

Daniel Flachs

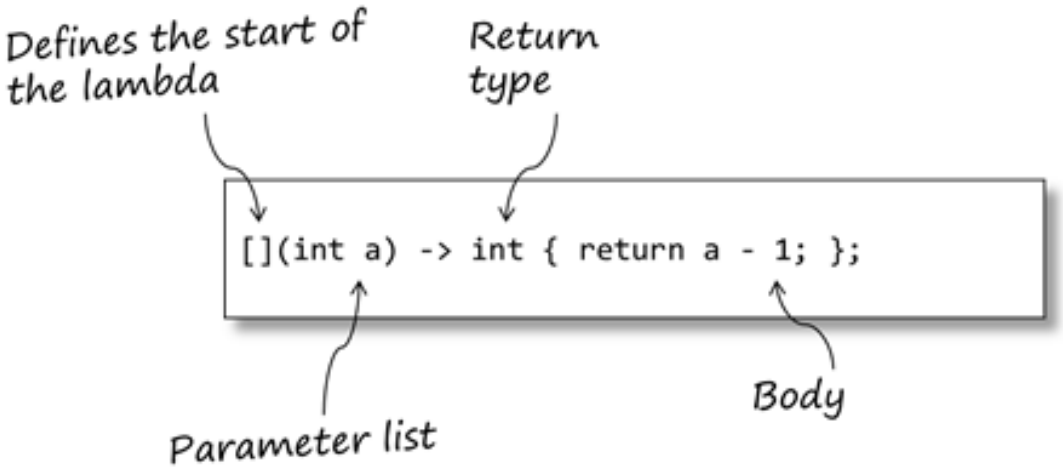
Chair of Practical Computer Science III:
Database Management Systems

10/04/2019

Created April 10, 2019

Contents

Lambda Expressions



Source: https://blog.feabhas.com/2014/03/demystifying-c-lambdas/

Task 1

You are given a `std::vector<int>` called `vec`, i. e., a vector of integers.

b) Using `std::for_each`, write a lambda expression to print the vector element-wise.

c) Use `std::for_each` and a lambda expression to modify each value of `vec` such that it is replaced by its absolute value, e. g., $-7 \rightarrow 7$, and $11 \rightarrow 11$.

d) Implement a lambda function that calculates the sum of all elements in the vector.

Task 2

This exercise builds upon the physical algebra implementation from exercise sheet 6 and uses the same simple main-memory database implementation. Sheet 6 was concerned with table scans, selections and projections, whereas this sheet deals with different implementations for joins.

Task 2a

Recap how the *Nested Loop Join (NLJ)* and the *Hash Join (HJ)* algorithm work. Write down their pseudocode.

Remarks

- The following join algorithms compute the join $R \bowtie_p S$.
- For the Hash Join, the join predicate p is restricted to equality.
- By convention, for $R \bowtie^{\text{hj}} S$, the right relation S denotes the build relation (usually the smaller relation), the left relation R the probe relation.
- For two tuples x, y from some relation(s), $x \circ y$ denotes tuple concatenation.

Task 2a

Nested Loop Join

Input: two relations R and S; a join predicate p
Output: the tuples in $R \times S$ that satisfy p

```
1  for each tuple r ∈ R
2      for each tuple s ∈ S
3          if p(r, s)
4              output the tuple r ∘ s
```

Task 2a

Hash Join

Input: a build relation S, a probe relation R;
a join predicate p; a hash function h
Output: the tuples in $R \times S$ that satisfy p

```
1  Initialize H to be an empty hash table with h as a hash function.
2  for each tuple s ∈ S                    // Build
3      Insert s into H[h(s)].
4  for each tuple r ∈ R                    // Probe
5      for each t ∈ H[h(r)]
6          if p(r, t)
7              Output the tuple r ∘ t.
```

Task 2b

List the pros and cons of both the Nested Loop Join and the Hash Join.

Nested Loop Join

+ Can handle arbitrary join predicates.
+ No materialization of intermediate results (tuples) necessary.
– Slow runtime: $O(|R| \cdot |S|)$

Hash Join

+ Fast runtime: $O(|R| + |S|)$
– In general, only equi-joins are possible.
– Pipeline breaker: Tuples must be materialized in hash table.