typedef int TKey;

typedef int TValue;

typedef struct{

TKey c;

TValue v;

}TPair;

typedef struct{

TPair e;

SLNode\* next;

} SLNode;

typedef struct {

int m;

SLNode\* T[]; //array of pointers to NodeT

} HashTable;

typedef struct{

int currentPos;

SLNode\* currentNode;

HashTable\* table;

} IteratorHashTable;

Subalgorithm create(ith, ht):

ith.table ← ht

ith.currentPos ← 0

while ith.currentPos < ht.m and ht.T[ith.currentPos]= NIL do

ith.currentPos ← ith.currentPos + 1

end\_while

if ith.currentPos < m then

ith.currentNode ← ht.T[ith.currentPos]

else

ith.currentNode ← NIL

end\_if

end\_subalgorithm

Subalgorithm element(ith, elem):

elem ← ith.currentNode->e

end\_subalgorithm

Subalgorithm next(ith):

if ith.currentNode->next ≠ NIL then

ith.currentNode ← iht.currentNode->next

else

ith.currentPos ← ith.currentPos + 1

while ith.currentPos < ith.table.m and ith.table.T[ith.currentPos]=NIL do

ith.currentPos ← ith.currentPos + 1

end\_while

if ith.currentPos ≠ NIL then

ith.currentNode ← ith.table.T[ith.currentPos]

else

ith.currentNode ← NIL

end\_if

end\_subalgorithm

Function hasNext(ith):

// we can check whether the currentNode is NIL or currentPos is m

if ith.currentNode = NIL then

hasNext ← false

else

hasNext ← true

end\_if

end\_function