Container usw.

C++ Übung am 23. Juni 2016

Pairs `std::pair<type1, type2>`

- `std::pair<int,int> emptyPair;`
- `std::pair<std::string,float> myPair("text",3.14F);`
- `std::pair<std::string,float> copyPair(myPair);`

- `std::pair<std::string,float> myPair;`
  `myPair = std::make_pair("euler",2.71F);`

- `myPair = std::make_pair(std::string("str"),1);`

- `pos->first, pos->second`
  `oder myPair.first, myPair.second`
  `oder (*pos).first, (*pos).second`

typedef

- eigene Abkürzung definieren

- ```
  typedef map<int, wstring> WstringMap;
  WstringMap::iterator iter;
  ```

- ```
  typedef unordered_map<string, int> HashMap;
  ```

- ```
  typedef deque<wstring> MyDeque;
  typedef MyDeque::iterator MyIter;
  ```

- ```
  typedef vector<wstring> MyVec;
  MyVec DerVektor; = vector<wstring> DerVektor;
  ```

Frequenzlisten

```
#include <iostream>
#include <unordered_map>
#include <map> für Multimap

// Hash zum Aufbau der Frequenzliste
typedef std::unordered_map <std::wstring,int> lexikon_t;

int main() {

  std::setlocale(LC_ALL, "");
  lexikon_t lexikon;
  int num_of_words = 0;
  std::wstring word;

  while (std::wcin >> word) {
    lexikon[word]++;
    num_of_words++;
  }

  std::wcout << "Inserted " << num_of_words << " Words " << std::endl;

  print_sorted_freq(lexikon);

  return 0;
}
```

Frequenzlisten

```
#include <iostream>
#include <unordered_map>
#include <map> für Multimap

// Multimap Container zum Sortieren definieren
typedef std::multimap <int,std::wstring> mm_lexikon_t;

void print_sorted_freq(lexikon_t &words_freq) {
    mm_lexikon_t word_mm;
    // Iterator über unordered_map:
    for (auto lex_it = words_freq.begin();
            lex_it != words_freq.end(); ++ lex_it) {
        std::wcout << "Insert" << lex it->first << "="
                                << lex_it->second << std:: endl;
        // Paare umgekehrt in die Multimap speichern:
        word_mm.insert(std::pair<int,std::wstring>(lex_it->second,
                                                   lex_it->first ));
    }
    std::wcout << std::endl << L"Frequenzen sind sortiert." << std::endl;

  // Reverse-Iterator über Multimap:
    for (auto rev_mm_it = word_mm.rbegin();
             rev_mm_it != word_mm.rend(); ++rev_mm_it)
        // Ausgeben des Worts mit der Frequenz:
        std::wcout << rev_mm_it->second << "="
                   << rev_mm_it->first << std::endl;
}
```

Container

Container	**Deklarieren**	**Einfügen**
array	array<wstring, maxsize> words	words[numberOfWords]=word
vector	vector<wstring> words	words.push_back(word)
deque	deque<wstring> words	words.push_back(word)
list	list<wstring> words	words.push_back(word)
set	set<wstring> words	words.insert(word)
map	map<wstring,int> words	words.insert(pair<wstring,int> (word,1))
unordered_set	unordered_set<wstring> words	words.insert(word)
unordered_map	unordered_map<wstring,int> words	words.insert(pair<wstring,int> (word,1))

Container	**Element finden**	**Sortieren**	**Sonstiges**
array	find (algorithm)	sort (algorithm)	C++11
vector	find (algorithm)	sort (algorithm)	
deque	find (algorithm)	sort (algorithm)	
list	find (algorithm)	sort (algorithm)	
set	words.find(word)	ist bereits sortiert	
map	words.find(word)	ist bereits sortiert	Duplikate möglich
unordered_set	words.find(word)	kann nicht sortiert werden (Hash)	C++11
unordered_map	words.find(word)	kann nicht sortiert werden (Hash)	C++11, Duplikate möglich

Hausaufgabe 11

Lexikon.h

```
#include <iostream>
#include <algorithm>
#include <vector> // CONTAINER! zB. #include <deque>
using namespace std;

#ifndef LEXIKON_H_
#define LEXIKON_H_

class Lexikon {
    public:
        Lexikon(wstring wo);
        bool is_element(wstring wort);
        bool insert(wstring wort);
        void sort_me();
        void print_number_of_words();
        void head(int n);
    private:
        vector<wstring> words; // CONTAINER! zB. deque<wstring> words;
        void rm_punct(wstring& wort);
        int number_of_words;
};

#endif
```

Hausaufgabe 11

Lexikon.cpp

```
#include <iostream>
#include <algorithm>
#include <vector> // CONTAINER! zB. #include <deque>

using namespace std;
#include "DasLexikon.h"

Lexikon::Lexikon(wstring wo) {
     words.push_back(wo); // CONTAINER!
     number_of_words = 1;
}
void Lexikon::rm_punct(wstring& wort) {
     while(iswpunct(wort.front())) {
          wort.erase(0,1);
     }
     while(iswpunct(wort.back())) {
          wort.erase(wort.length()-1,1);
     }
}
bool Lexikon::is_element(wstring wort) {
     rm_punct(wort);
     vector<wstring>::iterator pos = find(words.begin(), words.end(), wort); // CONTAINER!
     if(pos != words.end()) {
          wcout << *pos;
          return true;
     }
     wcout << L"ist noch kein element -> ";
     return false;
} ...
```

Hausaufgabe 11

```
...

bool Lexikon::insert(wstring wort) {
     rm_punct(wort);
     if(!is_element(wort)) {
          words.push_back(wort); // CONTAINER!
          number_of_words++;
          wcout << L"eingefügt: ";
          wcout << number_of_words << endl;
          return true;
     }
     return false;
}

void Lexikon::print_number_of_words() {
     wcout << L"Das Lexikon enthält: " << number_of_words << L" Wörter." << endl;
}

void Lexikon::sort_me() {
     sort(words.begin(), words.end()); // CONTAINER!
}

void Lexikon::head(int n) {
     sort_me();
     for (int i=0; number_of_words < n && i < n; i++) {
          wcout << words.at(i); // CONTAINER!
     }
}
```

Lexikon.cpp

Hausaufgabe 11

LexikonMain.cpp

```
#include <iostream>
#include <vector> // CONTAINER!
using namespace std;

#include "DasLexikon.h"

int main() {
    setlocale(LC_ALL, "de_DE.UTF-8");
    wstring eins = L"Eins";
    Lexikon myLexikon(eins);
    myLexikon.print_number_of_words();

    wstring word;
    wcout << L"Bitte geben Sie ein Wort ein:" << endl;
    while (getline(wcin, word) and word != L"") {
        myLexikon.insert(word);
        wcout << L"Bitte geben Sie ein Wort ein oder beenden Sie die Eingabe mit
Enter" << endl;
        word = L"";
    }
    myLexikon.print_number_of_words();
    wcout << endl << L"Die ersten fünf Wörter des sortierten Lexikons: " << endl;
    myLexikon.head(5);
    return 0;
}
```

Laufzeitvergleich

Vector:

24000: 12.9 s
12000: 3.13s
6000:   0.81s
3000:   0.22s
1500:   0.06s

List:

24000: 11.1 s
12000: 2.67s
6000:   0.68s
3000:   0.18s
1500:   0.06s

Deque:

24000: 15.4 s
12000: 3.75s
6000:   0.92s
3000:   0.25s
1500:   0.07s

Set:

24000: 22.5 s
12000: 4.68s
6000:   1.13s
3000:   0.29s
1500:   0.07s

Map:

24000: 25.8 s
12000: 5.04s
6000:   1.20s
3000:   0.31s
1500:   0.08s

Klausurübung

Was gibt das Programm aus?

-> Programm auf Übungshomepage

Lösung

```
0 str1= Am Montag wird schon Klausur geschrieben

1 result= Achtung! Am Freitag wird Klausur geschrieben

2 result= Achtung! Am Freitag wird Klausur geschrieben Am Montag
wird schon Klausur geschrieben

3 result= Achtung! Am Freitag wird wird schon Klausur
geschrieben

4 str6=feiern Grill

5 str6=Feiern Grill

6 str6=Feiern und Grill

7 resultAchtung! Am Freitag wird wird schon Klausur geschrieben
zum letzten Mal! Feiern und Grillen
```