Текстовод Уникальность https://textovod.com/unique

Отчет проверки уникальности текста

Дата проверки: 2023-06-13 19:34:52

Уникальность 69%

Хорошо. Подойдет для большинства текстов.

Текст

```
#include < iomanip>
#include < iostream>
#include < stdexcept>
#include < string>
#include < vector>
template < typename T = int> class HashMap
{
public:
typedef unsigned key_t;
private:
struct HashEntry
{
HashMap: :key_t key = (HashMap: :key_t)-1;
T value = (T)0;
};
unsigned capacity;
std: :vector< HashEntry> map;
unsigned hash(key_t key)
{
return 10 + (key % 5);
}
void showMap(HashMap: :key_t hash, std: :string label, unsigned ansi_color)
{
std: :cout < < " map["
< < "\033[36m" < < std: :setw(2) < < hash < < "\033[0m"
< < "] - "
< < "\033[" < < ansi_color < < "m" < < label < < "\033[0m" < < std: :endl;
```

```
}
public:
HashMap(HashMap: :key_t capacity): capacity(capacity), map(std: :vector<
HashEntry> (capacity))
{
}
void show()
{
std: :cout < < "Hash table: " < < std: :endl;

for (HashMap: :key_t i = 0; i < map.size(); i++)
{
if (this-> map[i].key! = (HashMap: :key_t)-1)
{
this-> showMap(
i,
"Key: \033[36m" + std: :to_string(this-> map[i].key) + "\033[0m" + "; Value: " +
"\033[33m"
+ std: :to_string(this-> map[i].value),
0
);
}
else
{
this-> showMap(i, (std: :string) "Empty", 32);
}
}
}
void insert(HashMap: :key_t key, T value)
{
HashMap: :key_t hash = this-> hash(key);
HashMap: :key_t initial = hash;
std: :cout < < "Inserting '"
< < "\033[33m" < < value < < "\033[0m"
< < "' with key '"
< < "\033[36m" < < key < < "\033[0m"
< < "': " < < std: :endl;
// Find empty cell
if (this-> map[hash].key! = (HashMap: :key_t)-1)
{
do
```

```
{
if (this-> map[hash].key! = (HashMap: :key_t)-1)
{
this-> showMap(hash, "FILLED", 31);
}

hash = map.size() - 1 == hash? 0: hash + 1;
} while (initial! = hash & & this-> map[hash].key! = (HashMap: :key_t)-1);
}
if (initial == hash & & this-> map[hash].key! = (HashMap: :key_t)-1)
{
std: :cout < < " "
< < "\033[31m"
< < " Hash map is filled. Skipping."
< < "\033[0m" < < std: :endl;
}
else
{
this-> showMap(hash, "EMPTY \033[0m - filling", 32);
this-> map[hash] = HashMap: :HashEntry{key, value};
}
}
T *find(HashMap: :key_t key)
{
HashMap: :key_t hash = this-> hash(key);
HashMap: :key_t initial = hash;
do
{
if (this-> map[hash].key == key)
{
this-> showMap(hash, "HIT", 32);
return & (this-> map[hash].value);
}
this-> showMap(hash, "MISS", 31);
hash = map.size() - 1 == hash? 0: hash + 1;
} while (initial! = hash);
return nullptr;
}
};
#include < chrono>
#include < iomanip>
```

```
#include < iostream>
#include < random>
#include < stdlib.h>
#include "hash_map.h"
template < typename T = unsigned> void showVector(std: :vector< T> vec)
{
std: :cout < < "Input array ("
< < "\033[33m" < < vec.size() < < "\033[0m"
< < "): " < < std: :endl;
for (size_t i = 0; i < vec.size(); i++)
{
std: :cout < < " array["
< < "\033[36m" < < std: :setw(2) < < i < < "\033[0m"
< < "] - "
< < "\033[33m" < < vec[i] < < "\033[0m" < < std: :endl;
}
std: :cout < < std: :endl;
}
int main()
{
const int M = 20;
const int n = 15;
const int min = 12000;
const int max = 34000;
// Some random C++ bullshit
std: :random_device rd;
// seed value is designed specifically to make initialization
// parameters of std: :mt19937 (instance of std: :mersenne_twister_engine< > )
// different across executions of application
std: :mt19937: :result_type seed
= rd()
^ ((std: :mt19937: :result_type
)std: :chrono: :duration_cast< std: :chrono: :seconds> (std: :chrono: :system_clock:
:now().time_since_epoch())
.count()
+ (std: :mt19937: :result_type)std: :chrono: :duration_cast< std: :chrono:
:microseconds> (
std: :chrono: :high_resolution_clock: :now().time_since_epoch()
)
.count());
std: :mt19937 gen(seed);
```

```
std: :uniform_int_distribution< unsigned> distribution(min, max);
std: :vector< unsigned> vec;
HashMap< unsigned> map(M);
for (int i = 0; i < n; i++)
{
vec.push_back(distribution(gen));
map.insert(i + 1, vec.back());
std: :cout < < std: :endl;
}
showVector(vec);
map.show();
std: :cout < < std: :endl;
while (true)
{
int searchKey;
std: :cout < < "Search for value with key ("
< < "\033[35m"
< < "use ctrl + c to exit"
< < "\033[0m"
< < ") - "
< < "\033[33m";
std: :cin > > searchKey;
std: :cout < < "\033[0m";
unsigned *value = map.find(searchKey);
if (value == nullptr)
{
std: :cout < < std: :endl
< < "\033[31m"
< < " Key is not present in the hash map."
< < "\033[0m" < < std: :endl
< < std: :endl;
}
else
{
std: :cout < < std: :endl
< < " Found: "
< < "\033[33m" < < *value < < "\033[0m" < < std: :endl
< < std: :endl;
}
}
return EXIT_SUCCESS;
```

}

Источники

- https://caiorss.github.io/C-Cpp-Notes/math_and_numerical_computing.html (15%)
- https://sodocumentation.net/cplusplus/topic/681/std--map (14%)
- https://github.com/ketorg0z/ADMMMetro (10%)
- https://ru.stackoverflow.com/questions/440179/c-%D0%9A%D0%B0%D0%BA-
%D0%BE%D0%B1%D0%BC%D0%B5%D0%BD%D1%8F%D1%82%D1%8C-
%D0%B7%D0%BD%D0%B0%D1%87%D0%B5%D0%BD%D0%B8%D1%8F%D0%BC%D0%0
%D0%B4%D0%B2%D0%B5-
%D0%BF%D0%B5%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%BD%D1%8B%D1
(9%)
- https://programmerall.com/article/49561892968/ (9%)
- https://marketsplash.com/tutorials/visual-studio/visual-studio-cpp/ (8%)
- https://gist.github.com/santa4nt/5ba0c75836294a8bf315 (8%)
- https://metanit.com/cpp/tutorial/2.3.php (8%)
- https://metanit.com/cpp/tutorial/6.1.php (8%)
- https://www.CyberForum.ru/cpp-beginners/thread3108960.html (8%)
- https://www.CyberForum.ru/cpp-beginners/thread1918231.html (8%)
- https://tproger.ru/articles/iskljuchenija-v-cpp-tipy-sintaksis-i-obrabotka/ (8%)
- https://www.gormanalysis.com/blog/making-a-binary-search-tree-in-cpp/ (8%)
- https://thenewstack.io/getting-started-with-c-and-influxdb/ (5%)
- https://www.CyberForum.ru/cpp-beginners/thread981414.html (4%)
- https://ru.stackoverflow.com/questions/1149871/%D0%9F%D1%80%D0%BE%D0%B3%D
%D0%B7%D0%B0%D0%B2%D0%B5%D1%80%D1%88%D0%B0%D0%B5%D1%82%D1%
%D0%B4%D0%BE-%D0%B2%D0%B2%D0%BE%D0%B4%D0%B0-
%D0%B7%D0%BD%D0%B0%D1%87%D0%B5%D0%BD%D0%B8%D1%8F (3%)
- https://www.udacity.com/blog/2021/08/creating-a-new-line-in-cpp.html (2%)
- https://habr.com/ru/articles/527044/ (2%)