

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Інститут прикладного системного аналізу

Кафедра математичних методів системного аналізу

Звіт

про виконання лабораторної роботи №6 (2-2) з дисципліни «Алгоритми та структури даних»

Виконав:

студент I курсу, групи КА-07

Москаленко Максим Геннадійович

Прийняв:

Київ — 2020

|  |  |
| --- | --- |
| МЕТА ТА ВАРІАНТ | 1 |
| ЗАДАЧА 1 | 2 |
| ВИСНОВОК | 5 |

МЕТА

Ознайомитись і дослідити структуру даних хеш-таблиця. Набути навичок реалізації хеш-таблиці за методом ланцюжків мовою програмування C++, познайомитись з використанням STL контейнерів на прикладі unordered\_map та порівняти власну реалізацію з готовим бібліотечним рішенням.

ВАРІАНТ 18

ЗАДАЧА

1) Реалізувати функцію для генерації випадкового великого числа, що буде унікальним ключем-ідентифікатором даних у хеш-таблиці.

2) Реалізувати хеш-таблицю за методом ланцюжків:

2.1 Створити структуру Data для зберігання різнотипних даних відповідно до варіанту, вона буде використовуватись у якості значень для хеш-таблиці. Реалізувати генерацію випадкових даних для полів структури у пустому конструкторі Data().

2.2 Створити структуру HashNode для зберігання ключа та значення

2.3 Створити структуру LinkedList з підтримкою додавання елементів

2.4 Створити структуру HashTable, яка буде містити масив зв’язних списків bucketsArray статичного розміру M = 10000 та реалізувати в ній основні функції:

* hash(key) знаходження хеш-функції від ключа
* insert(key, value) додавання значення під відповідним ключем (заміна у випадку існування ключа)
* find(key) знаходження значення під відповідним ключем, функція  
  повертає вказівник на знайдений елемент або NULL
* get() зчитування n-го елементу
* erase(key) видалення значення під відповідним ключем
* size() знаходження кількості елементів в хеш-таблиці  
  Всі функції повинні працювати за O(1).

3) Оптимізуватихеш-таблицю,додавшидинамічнузмінурозмірумасиву bucketsArray в залежності від loadFactor (максимально можлива завантаженість таблиці):

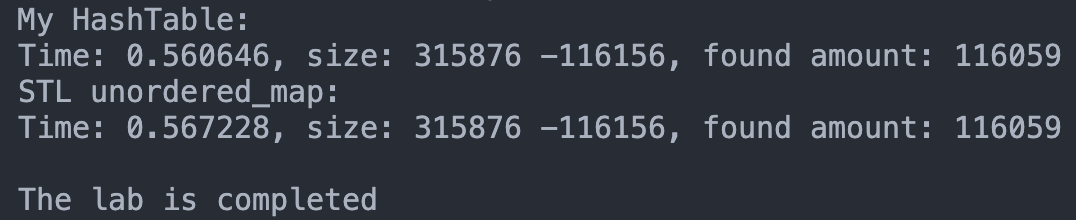
4) Провести тестування, використавши вказану нижче функцію testHashTable(). Обрати оптимальне значення loadFactor. Перевірити правильність та швидкість роботи, порівнявши з готовим бібліотечним рішенням STL unordered\_map.

5) Реалізувати ще одну хеш-таблицю, де в якості ключів використати тип

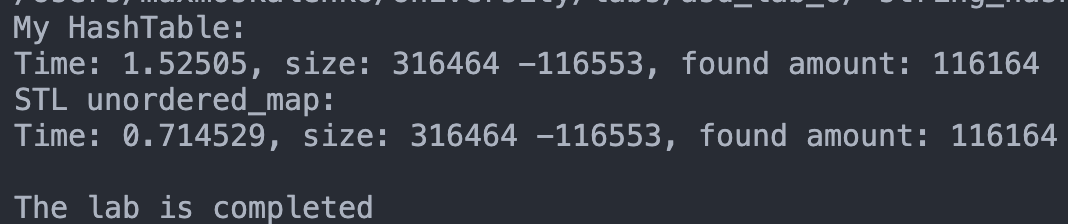
string

6) Реалізувати ще одну хеш-таблицю за методом відкритої адресації

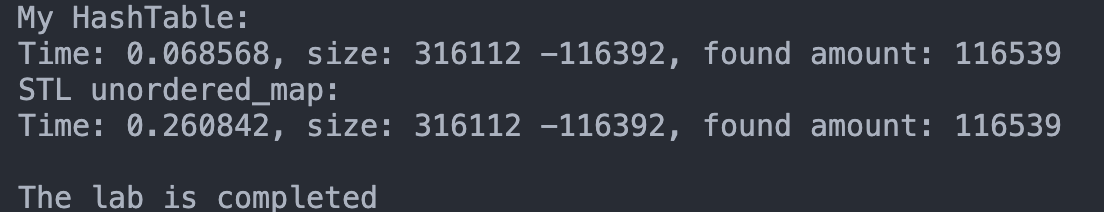
Результат виконання програми (основна частина):



Результат виконання програми (ключі типу string):



Результат виконання програми (відкрита адресація):

  
Код:

|  |
| --- |
| Весь код у доданому до файлу архіві:  main\_hash\_table.cpp - основна задача  string\_hash\_table.cpp - задача для ключів типу string  open\_hash\_table.cpp - задача для таблиці з відкритою адресацією |

ВИСНОВКИ

У результаті лабораторної роботи було створено та реалізовано структуру даних: Хеш-таблиця. Дану структуру було реалізовано методом ланцюжків та методом відкритої адресації. Для методу ланцюжків було додатково створено структуру даних двозв’язний список. Хеш-таблиця з відкритою адресацією була реалізована методом лінійного зондування. Для хешування ключів було використано функцію ділення за модулем.