МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»
Кафедра програмної інженерії та інформаційних технологій управління

Звіт з лабораторної роботи № 2
з дисципліни «Основи теорії алгоритмів»

Виконав:
ст. гр. КН-221в
Бородько Я.А.
Перевірила:
доцент каф. ПІТУ
Солонська С.В.

Харків
2022

ТЕМА: БАЗОВІ СТРУКТУРИ ДАНИХ. ХЕШ-ТАБЛИЦІ

ЗАВДАННЯ НА ЛАБОРАТОРНУ РОБОТУ

Розробити програму, яка читає з клавіатури цілі числа N, M (1 < N, M < 256), N пар <ключ, значення> (ключ — ціле, дійсне число або рядок в залежності від варіанту завдання; значення — рядок; усі рядки до 255 символів), жодний з яких не повторюється та ще M ключів. Всі рядки розділяються пробілом або новим рядком. Програма зберігає пар рядків до хеш-таблиці та видає на екран значення, що відповідають переліченим ключам.

Використати ключем ціле число, та застосувати тривіальне хешування.

МЕТА РОБОТИ

Ознайомлення з хеш-функціями та хеш-таблицями та отримати навички програмування алгоритмів, що їх обробляють.

1 ОСНОВНІ ТЕОРЕТИЧНІ ПОЛОЖЕННЯ

Часто бувають потрібні динамічні множини, що підтримують тільки «словникові операції» додавання, пошуку і видалення елемента. У цьому випадку часто застосовують так зване хешування; відповідна структура даних називається «хеш-таблиця». У гіршому випадку пошук в хеш-таблиці може займати стільки ж часу, скільки пошук у списку ($O(n)$), але на практиці хешування вельми ефективно.

У той час як при прямій адресації елементу з ключем kвідводиться позиція номер k, при хешуванні цей елемент записується в позицію номер $h(k)$в хеш-таблиці (hashtable) $T[0..m-1]$, де $h:U \to 0,1,...,m-1$- деяка функція, звана хеш-функцією (hash function).

Хороша хеш-функція повинна (наближено) задовольняти припущеннями рівномірного хешування: для чергового ключа всі mхеш-значень повинні бути рівноймовірні.

Тривіальне хешування полягає у використанні хеш-функції, що повертає власний аргумент.

Побудова хеш-функції методом ділення із залишком (division method) полягає в тому, що ключу kставиться у відповідність залишок від ділення kна m, де m- число можливих хеш-значень: $h(k)=kmodm$.

Наприклад, якщо розмір хеш-таблиці m=12і ключ дорівнює 100, то хеш-значення дорівнює 4.

Хороші результати зазвичай виходять, якщо вибрати в якості mпросте число, далеко відстоїть від ступенів двійки.

Побудова хеш-функції методом множення (multiplication method) полягає в наступному. Нехай кількість хеш-значень дорівнює m. Зафіксуємо константу A в інтервалі (0,1), і покладемо $h(k)=[m(kAmod1)]$, де $kAmod1$- дрібна частина kA.

Перевага методу множення в тому, що якість хеш функції мало залежить від вибору m. Звичайно як mвибирають ступінь двійки, оскільки в більшості комп'ютерів множення на таке mреалізується як здвигання слова.

Хешування Пірсона (Pearson hashing) - алгоритм, запропонований Пітером Пірсоном для процесорів з 8-бітними регістрами, завданням якого є швидке обчислення хеш-коду для рядка довільної довжини. На вхід функція виходить слово W, що складається з nсимволів, кожен розміром 1 байт, і повертає значення в діапазоні від 0 до 255. Значення хеш-коду залежить від кожного символу вхідного слова.

2 ОПИСАННЯ РОЗРОБЛЕНОГО ЗАСТОСУНКУ

Програма реалізована в одному файлі main.cpp

```
#include<iostream>
#include<string>
#include<cstring>
using namespace std;

const int M = 256;
string arr[M];
class IncorrectInput { };

void HashFoo(int k, string value)
{
```

```
	if (k < 0 || k > 256)
		throw IncorrectInput();
	arr[k] = value;
}

string returnFoo(int k) { return arr[k]; }
void HashInput(int n)
{
	for (int i = 0; i < n; i++)
	{
		int key;
		string word;
		cout << "Введіть ключ, що є цілим числом" << endl;
		cin >> key;
		cout << "Введіть рядок, що є значенням" << endl;
		cin >> word;
		HashFoo(key, word);
	}
}

void HashOutput(int m)
{
	for (int i = 0; i < m; i++)
	{
		int key;
		cout << "Введіть номер ключа, з тих що ви вводили при записі до
хеш-таблиці, для виводу значення" << endl;// При введенні неправиьного ключа,
ви отримаєте порожній рядок
		cin >> key;
		cout << returnFoo(key) << endl;
	}
}

void main()
{
	try
	{
		setlocale(LC_ALL, "Ukr");
		int n, m;
		cout << "Введіть кількість пар ключів та значень" << endl;
		cin >> n;
		if (n > 256 || n <= 0)
			throw IncorrectInput();
		cout << "Введіть кількість ключів, що будуть виведені" << endl;
		cin >> m;
		if (m > 256 || m <= 0)
			throw IncorrectInput();
		HashInput(n);
		HashOutput(m);
	}
	catch (IncorrectInput&)
```

```
	{
		cout << "Ви ввели неправильне число, що виходить за межі вводу" << endl;
	}
	}
```

Результати роботи програми наведені на рис. 3.1:

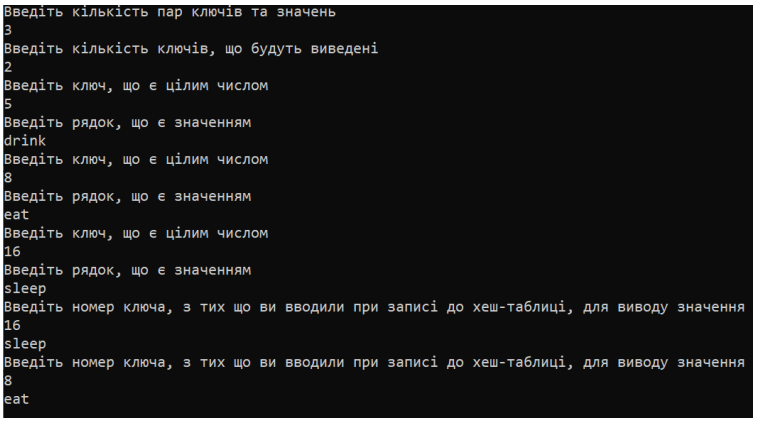

Рисунок 3.1 – Результат

ВИСНОВКИ

Під час цієї лабораторної роботи була розроблена програма, яка створює хеш-таблицю з введених з клавіатури символів, які заповнюють структуру даних, що визначена відповідно мого персонального варіанту, а саме хеш-таблицю з ключем, яким є ціле число, використавши тривіальне хешування. Під час цієї лабораторної роботи була викладена інформація про різні типи динамічних структур даних та їх використання при виконанні різних програм. Ця програма може виконувати різні дії, які визначені в роботі з хеш-таблицями.

СПИСОК ВИКОРИСТАНИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1 Методичні вказівки до виконання лабораторних робіт з курсу "Алгоритми і структури даних": для студентів, які навчаються за спец. 121 "Інженерія програмного забезпечення" [Електронний ресурс] / уклад. Н. К. Стратієнко, І. О. Бородіна ; Харківський політехнічний інститут, національний технічний університет

університет – Електрон. текстові дані. – Харків, 2017. – 36 с. – Режим доступу: http://repository.kpi.kharkov.ua/handle/KhPI-Press/26426. (дата звернення: 18.02.2020)

2 Алгоритми і структури даних: практикум: навчальний посібник/ Н.К. Стратієнко, М.Д. Годлевський, І.О. Бородіна.- Харків: НТУ"ХПІ", 2017. - 224 с. (дата звернення: 18.02.2020)

3 Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. Data Structures and Algorithms. Addison-Wesley, 1983.

(А.В. Ахо, Д. Е.Хопкрофт, Д. Д.Ульман: Структури даних і алгоритми. - М. : Вільямс, 2003 - 384 с.) (дата звернення: 18.02.2020)

4 N. Wirth. Algorithms + Data Structures = Programs. Prentice-Hall, 1976.

(Н. Вірт. Алгоритми і структури даних. Нова версія для Оберона. - М.: ДМК Пресс, 2010. -272 с.) (дата звернення: 18.02.2020)