МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КПІ»

Кафедра інформаційних систем та технологій

Звіт

з комп'ютерного практикуму 4(5)

«Пошук у масиві. Відкрите хешування»

з дисципліни

«Теорія алгоритмів»

Бригада – 10

Варіант № 1

Перевірила:

ст. вик. Солдатова М.О.

Виконали:

Бойко Катерина,

Гоголь Софія,

Павлова Софія,

Хіврич Володимир

Київ 2022

Комп’ютерний практикум 5

Тема: Пошук у масиві. Відкрите хешування

Мета роботи: порівняння алгоритмів розв’язку задачі, побудованих різними методами.

Завдання

Постановка задачі:

Варіант 1:

Скласти список дат народження дівчат потоку для можливості надалі вручення їм квітів на день народження, а для того, щоб зберегти це в таємниці згенерувати відповідну хеш-таблицю.

Мета задачі – зробити список дат народження, зробити хеш-таблицю для секретності

Використані структури даних: масиви, покажчики, рядки, цілочисельні типи, створені власні класи (Student, HashTable).

Обрані наступні хеш-функції:

Метод ділення

Метод ділення досить простий – використовується залишок від ділення на M: h (K) = K mod M. В якості M краще використовувати просте число.

Метод множення

Для мультиплікативного хешування використовується наступна формула : h (K) = [M * ((C * K) mod 1)]; Тут відбувається множення ключа на константу C, що лежить в інтервалі [0 .. 1]. Після цього береться дробова частина цього виразу і множиться на деяку константу M, обрану таким чином, щоб результат не вийшов за межі хеш-таблиці. Оператор [] повертає найбільше ціле, яке менше аргументу.

Метод вирішення колізій при відкритому хешуванню:

Ситуація, коли для різних ключів отримується одне й те саме хеш-значення, називається колізією. При відкритому хешуванні для вирішення колізій усі елементи, що хешуються в одну комірку, об’єднуються у зв’язний список.

Модель:

Основні величини:

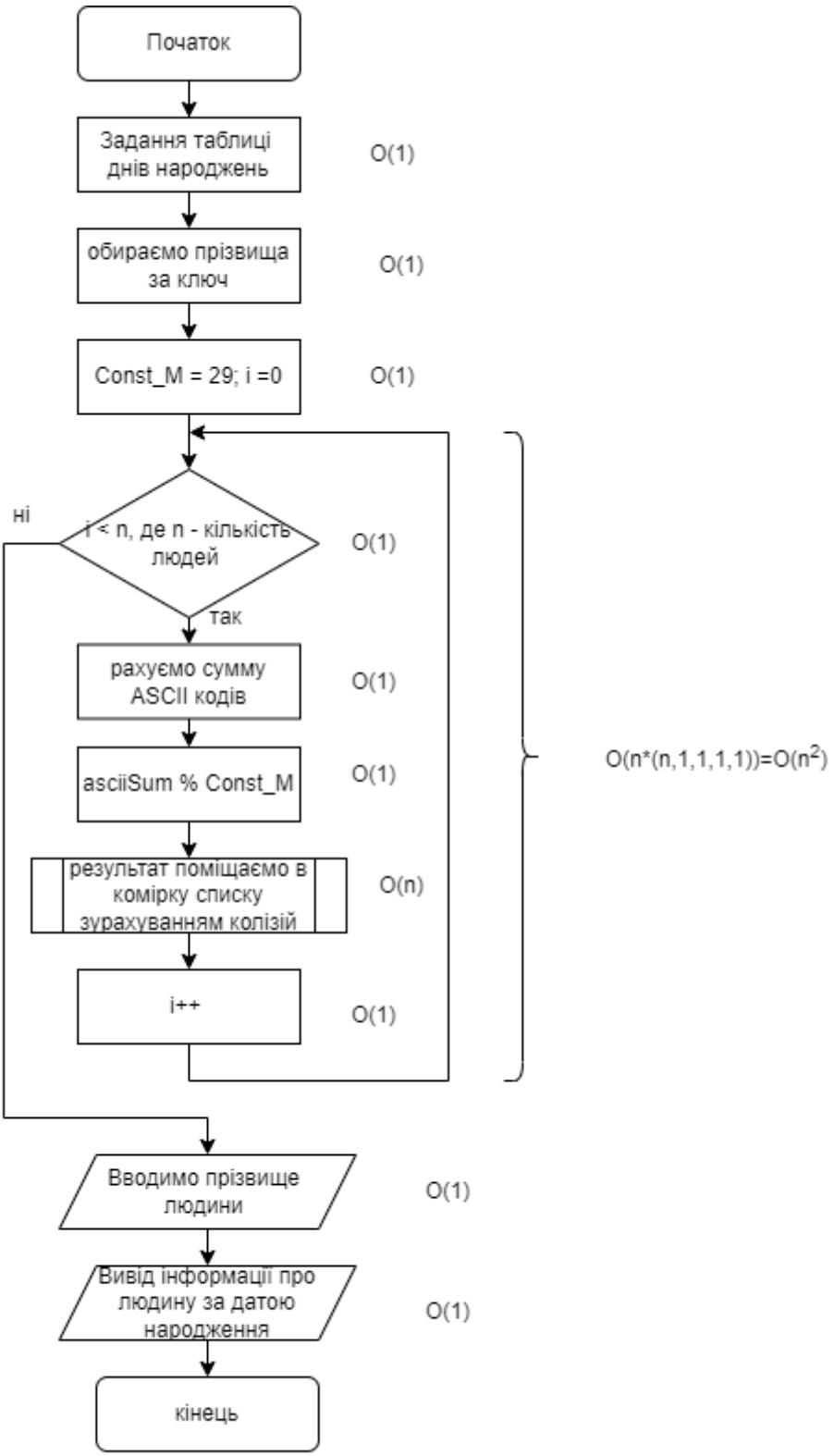
Назва змінної	Тип змінної	Значення змінної
next	Student*	За появи колізії елементи студент будуть записані в однозв'язний список
name	string	ім’я студента
surname	string	прізвище студента
age	string	дата народження студента
hash	int	хеш номер студента
table	Student*	таблиця студентів

Допоміжні величини:

Назва змінної	Тип змінної	Значення змінної
asciSum	int	сума ASCII кодів
newTable	*HashTable*	таблиця студентів
type	int	тип хеш-функції

Табл 1. Таблиця величин

Блок-схема



Складність хеш функції ділення $O(n^2)$

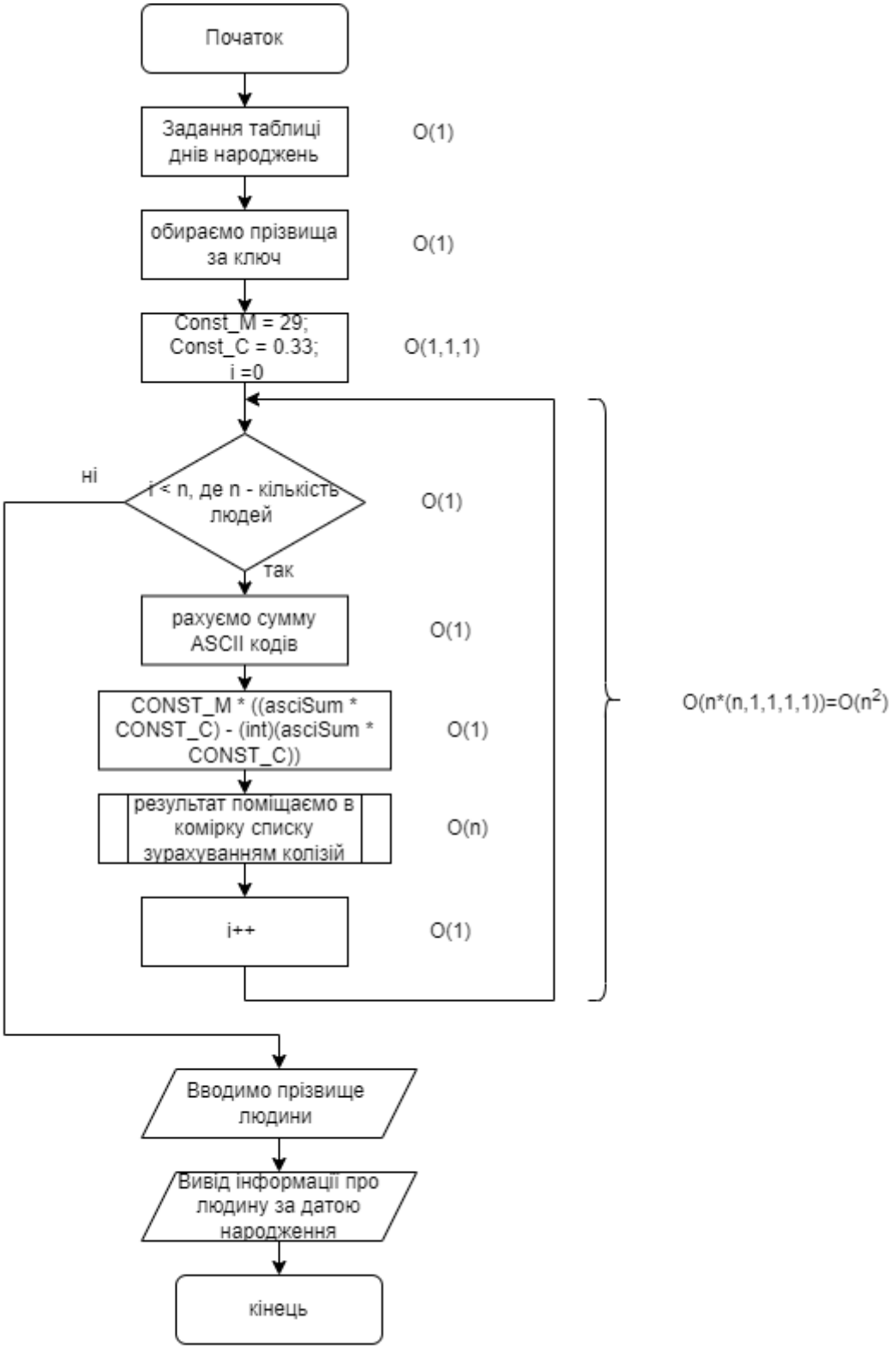


Рис.1-2 Блок-схеми реалізації алгоритму

Лістінг програми:

```
#include <iostream>
#include <string>
#include <windows.h>
#include <random>

#define CONST_M 29      // Просте число - константа M використане при хешуванні
#define CONST_C 0.33    // Константа C для хешування множенням

using namespace std;

// Клас студента
class Student {
public:
    Student* next;      // За появи колізії елементи студент будуть записані в однозв'язний список
    string name;
    string surname;
    string age;
    int hash;

    // Конструктор студента
    Student(string name, string surname, string age, int hash)
    {
        this->name = name;
        this->surname = surname;
        this->age = age;
        this->hash = hash;
        this->next = NULL;
```

```
    }

    // Видалення студента (викликається після return 0)
    ~Student()
    {
        cout << "* Delete " << this->surname << " *" << endl;
        if (this->next != NULL)
            delete this->next;
    }
};

// Хеш-таблиця, представлена у вигляді масиву (кожен елемент масиву може бути списком)
class HashTable
{
    Student* table[CONST_M];

    // Хеш-функція ділення
    // Рахує суму ASCII кодів, ділить її на константу M і отримує остачу від ділення
    static int hashDivision(string str)
    {
        int asciSum = GetASCII(str);
        return asciSum % CONST_M;
    }

    // Хеш-функція множення
    // Рахує суму ASCII кодів, множить її на константу C, відкидає цілу частину, множить результат на константу M і відкидає дробову частину
    static int hashMultiplication(string str)
    {
        int asciSum = GetASCII(str);
```

```
        return (int)(CONST_M * ((asciSum * CONST_C) - (int)(asciSum * CONST_C)));
    }

    static int GetASCII(string str)
    {
        int asciSum = 0;
        for (int i = 0; i < str.length(); i++)
            asciSum += str[i];
        return asciSum;
    }
public:
    // Конструктор таблиці
HashTable()
    {
        for (int i = 0; i < CONST_M; i++)
            table[i] = NULL;
    }

    // Видалення таблиці (викликається після return 0)
    ~HashTable()
    {
        cout << "\n* Delete table *" << endl << endl;
        for (int i = 0; i < CONST_M; i++)
            delete table[i];
    }

    // Вставляє елемент студент у таблицю і виводить його фамілію та хеш номер
    void push(string name, string surname, string age, int type)
    {
        int hashNumber = FindHashNumber(surname, type);
```

```
    Student* pers = new Student(name, surname, age, hashNumber);
    Student* place = table[hashNumber];

    cout << pers->surname << " (#" << pers->hash << ")" << endl;

    if (place == NULL)
    {
        table[hashNumber] = pers;
        return;
    }

    while (place->next != NULL)
        place = place->next;
    place->next = pers;
}

// Отримує студента з таблиці по його фамілії
Student* find(string surname, int type)
{
    int hashNumber = FindHashNumber(surname, type);
    Student* result = table[hashNumber];
    if (!result)
    {
        cout << "Student not found!" << endl;
        return NULL;
    }
    while (result->surname != surname)
    {
        if (!result->next)
        {
            cout << "Student not found!" << endl;
```

```
            return NULL;
        }
        result = result->next;
    }
    return result;
}

static int FindHashNumber(string surname, int type)
{
    int hashNumber;
    switch (type)
    {
    case 0:
        hashNumber = hashDivision(surname);
        break;
    case 1:
        hashNumber = hashMultiplication(surname);
        break;
    }
    return hashNumber;
}
};

int main()
{
    HashTable newTable;
    string type0;
    int type;

    cout << "\t * -------- * Computer Workshop 4 * -------- * " << endl;
    cout << "\t\t     Brigade 10. Variant 1 " << endl;
```

```
cout << "\t\t\t Open hashing " << endl << endl;

cout << "------------Menu------------" << endl;
cout << " 0 - realised on division" << endl;
cout << " 1 - realised on multiplication" << endl;
cout << "----------------------------" << endl;
cout << "Choose hash-function (0/1): ";
cin >> type0;
if (type0 != "0" && type0 != "1")
{
   cout << "\n-----------Error------------" << endl;
   cout << "\nNo such type" << endl;
   cout << "\n----------------------------" << endl;
   return 0;
}
type = atoi(type0.c_str());     // конвертація string у int

cout << "\n----------Students----------" << endl << endl;

newTable.push("Sofiia", "Pavlova", "04.03.2004", type);
newTable.push("Valeria", "Ryabenko", "03.09.2003", type);
newTable.push("Sofiia", "Hohol", "13.09.2004", type);
newTable.push("Anna", "Gonchar", "04.11.2004", type);
newTable.push("Kate", "Kolomoychenko", "03.12.2003", type);
newTable.push("Kate", "Boyko", "07.12.2003", type);
newTable.push("Amaliya", "Pavlenko", "07.01.2004", type);
newTable.push("Tanya", "Guseva", "23.01.2004", type);
newTable.push("Mariia", "Krushinska", "27.01.2004", type);
newTable.push("Lisa", "Djhioeva", "17.03.2004", type);
newTable.push("Kate", "Melnicova", "28.03.2004", type);
```

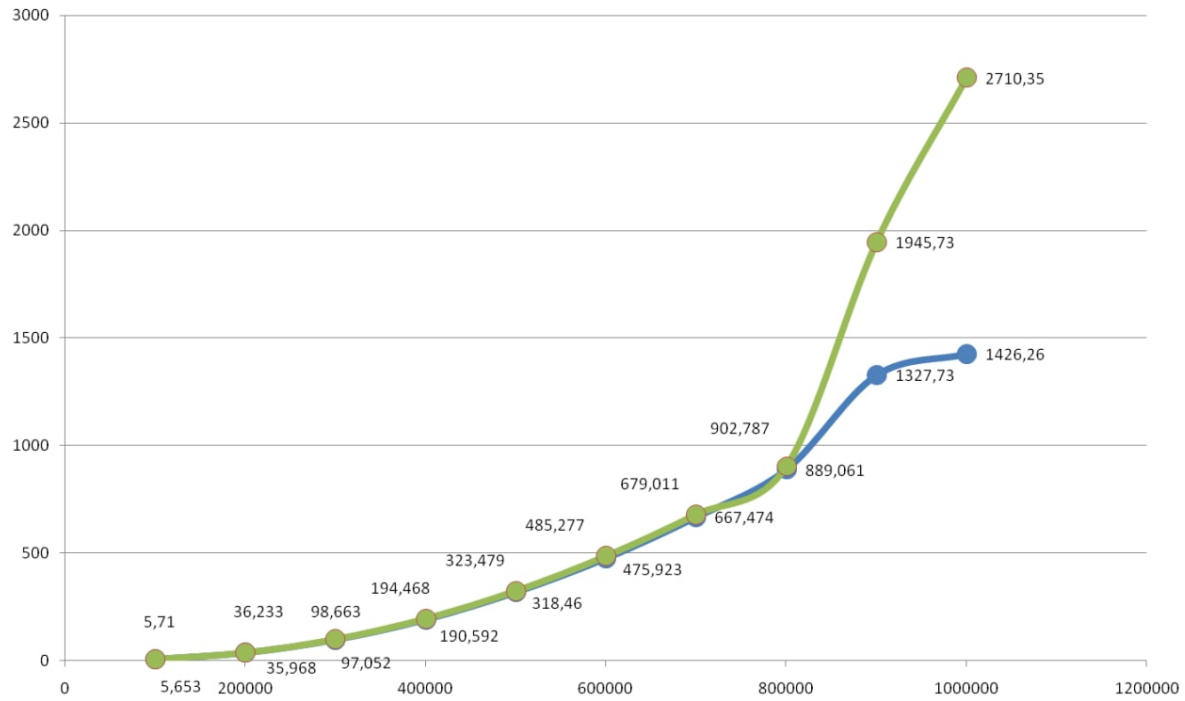
```
    string studentName;
    cout << "\nEnter student's surname to find her date of birth: ";
    cin >> studentName;
    cout << "\n-----------Result-----------" << endl << endl;
    Student* search = newTable.find(studentName, type);
    if (search)
        cout << search->name << " " << search->surname << " - " << search->age
<< endl;
    cout << "\n----------------------------" << endl;
    return 0;
}
```

Перевірка обчислювальної складності програми:

Діагр. 1-2. Діаграми часу виконання програми в залежності від методу

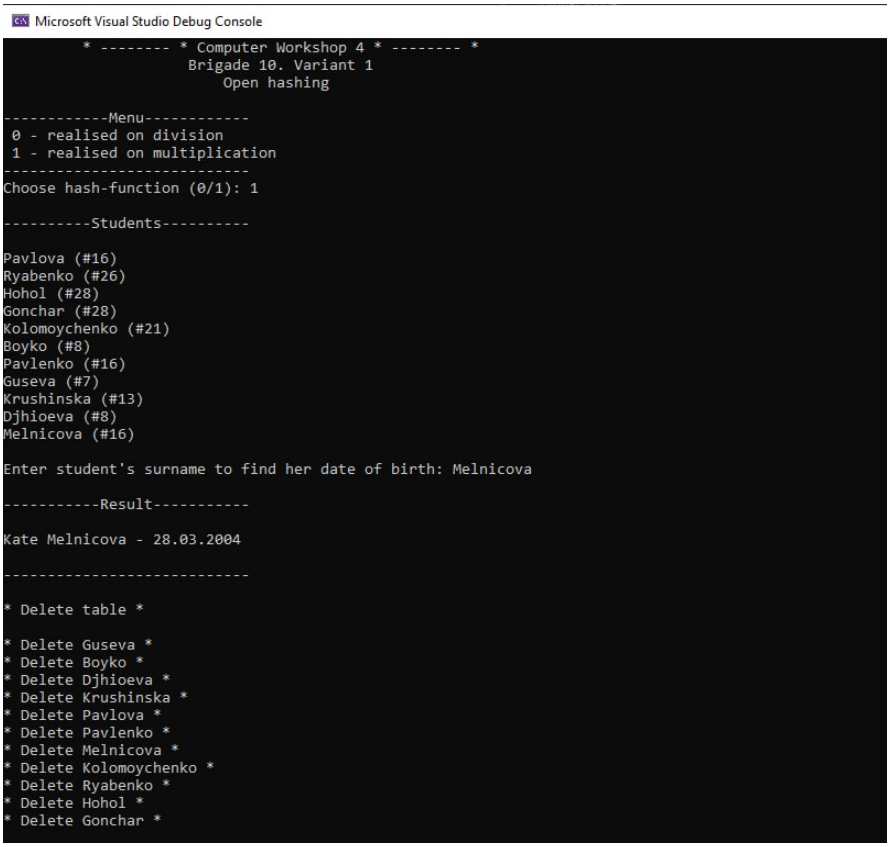
Червоний пунктирний графік – теоретична складність.
Синій/Зелений графік – реальна складність.

Перевірка обчислювальної складності програм підтвердила аналітичну оцінку складності алгоритмів O(n^2).



Діагр. 3. Порівняльна діаграма часу виконання програми в залежності від методу

Скріншоти роботи програми:



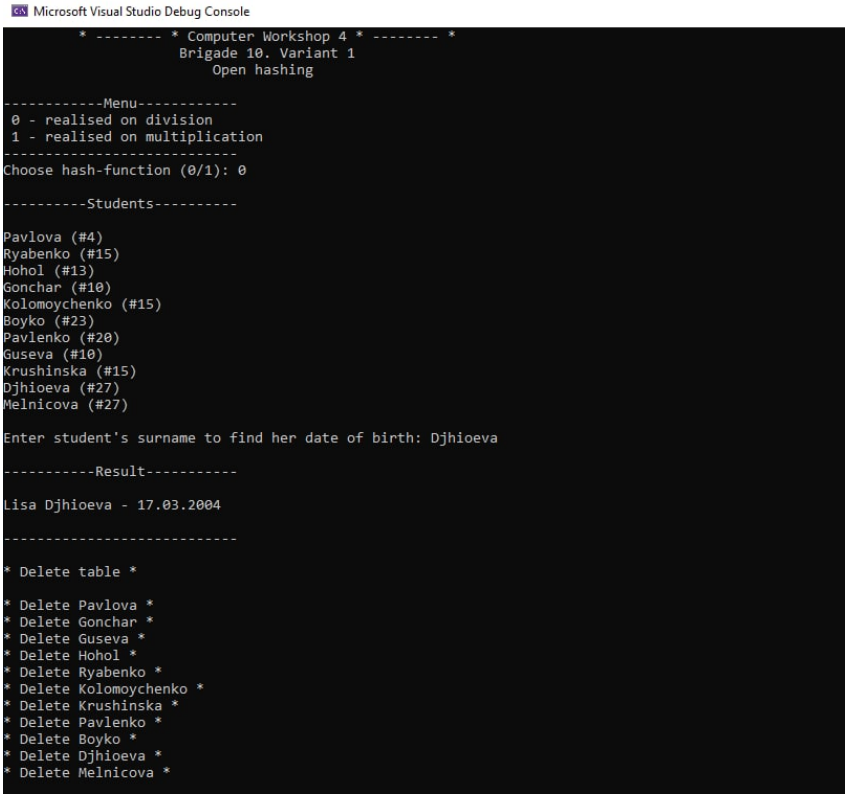


Рис. 3-4. Результат виконання програми

Перевірка правильності програми:

Вхідні дані type0	Результат	Призначення тесту
-1	Відомий	Перевірка, завершення некоректні вхідні дані (додавання повідомлень про помилки, завершення роботи програми)
2	Відомий	Перевірка реакції на некоректні вхідні дані (додавання повідомлень про помилки, завершення роботи програми)
b	Відомий	Перевірка реакції на некоректні вхідні дані (додавання повідомлень про помилки, завершення роботи програми)
*	Відомий	Перевірка реакції на некоректні вхідні дані (додавання повідомлень про помилки, завершення роботи програми)

Табл. 2. Таблиця тестування програми

Висновок

Під час виконання лабораторної роботи ми ознайомились та дослідили різні схеми відкритого хешування. Оцінили переваги та недоліки реалізації різних методів.

Відповіді на контрольні запитання:

1. *Сформулюйте та поясніть термін «задача пошуку».*

Пошук – основна операція, характерна для багатьох обчислювальних завдань. Дані, що обробляються, поділені на записи, або елементи, кожен з яких має ключ, що використовується при пошуку.

Мета пошуку: відшукання елементів з ключами, які відповідають заданому ключу пошуку. Найчастіше призначенням пошуку є отримання доступу до інформації усередині елементу (а не просто до ключа) з метою її обробки.

2. *Розкрийте суть абстрактного типу даних "Словник"*

Словник – абстрактний тип даних, який включає множину з операторами вставки, видалення та пошуку елементів.

3. *Поясніть, виходячи з чого обирається конкретна реалізація абстрактного типу даних «Словник» для розв'язку задачі та перерахуйте можливі варіанти реалізації.*

Найкраща конкретна реалізація вибирається виходячи з набору використаних операторів і розміру множини.

Реалізації словників:

- за допомогою сортованих або несортованих лінійних зв'язних списків;
- за допомогою масиву фіксованої довжини з покажчиком на останній заповнений елемент цього масиву;
- за допомогою двійкових векторів, припускаючи, що елементи множини є цілими числами 1…N або елементи множини можна зіставити з такою множиною цілих чисел (масив, індексований по ключам);
- за допомогою масиву з використанням хешування;
- за допомогою масиву з використанням бінарного пошуку;
- за допомогою нелінійних зв'язних списків з використанням бінарного пошуку.

4. *Поясніть термін «хешування». Перерахуйте та прокоментуйте переваги та недоліки форм хешування.*

Існує дві різні форми хешування:

- **відкрите**, або зовнішнє, хешування - дозволяє зберігати множини в потенційно нескінченному просторі, знімаючи тим самим обмеження на розмір множин;
- **закрите**, або внутрішнє, хешування - використовує обмежений простір для зберігання даних, обмежуючи таким чином розмір множин.

Відкрита форма хешування дозволяє зберігати скільки завгодно багато елементів, а при закритому хешуванні їх кількість обмежена розміром хеш-таблиці. На відміну від відкритого хешування, закрите не вимагає будь-яких додаткових структур даних. У осередках таблиці зберігаються не покажчики, а елементи вихідного масиву, доступ кожного з яких здійснюється за хеш-значення ключа, у своїй одна осередок може містити лише одне елемент.