МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

«Харківський Політехнічний Інститут»

Кафедра «Стратегічне управління»

Звіт з лабораторної роботи №6

«Хеш-таблиці та робота з ними»

з дисципліни

"Алгоритми та структури даних"

Варіант № 6

Перевірив:

ст. викл. каф. стратегічного

управління Мошко Є.О.

Виконав: Колій Дмитро

ст. гр. КН-320б

Харків – 2022

Мета: Вивчити роботу алгоритмів: прямої адресації, хеш-таблиці і відкритої адресації. Реалізувати перелічені вище алгоритми і провести їх порівняльний аналіз.

Індивідуальне завдання:

1. Нехай розмір хеш-таблиці дорівнює m = 20, а хеш-функція має вигляд: h(k) = [m(kA mod 1)], где А = ( 5 - l)/2. В які позиції потраплять ключі (див. таблицю, стовпець 3)?

2. Як буде виглядати хеш-таблиця з ланцюжками після того, як в неї послідовно помістили елементи з ключами (див. таблицю, стовпець 2) (в зазначеному порядку)? Число позицій в таблиці дорівнює 9, хеш-функція має вигляд h(k) = k mod 9.

3. Виконайте додавання ключів (в зазначеному порядку, див. таблицю стовпець 2) в хеш-таблицю з відкритою адресацією розміру m = 11. Для обчислення послідовності проб використовується лінійний метод з h'(k) = k mod m. Виконайте те ж завдання, якщо використовується квадратичний метод з тієї ж h', c1 = 1, c2 = 3, а також для подвійного хешування з h1 = h' и h2(k) = 1 + (k mod (m - 1)) .

Варіант завдання вибирається за номером студента в списку групи:

|  |  |  |
| --- | --- | --- |
| № варіанта | Масив | Ключі |
| 13 | 7, 5, 2, 19, 37, 54, 49, 8, 10 | 46, 47, 48, 49, 50 |

Вставка елемента

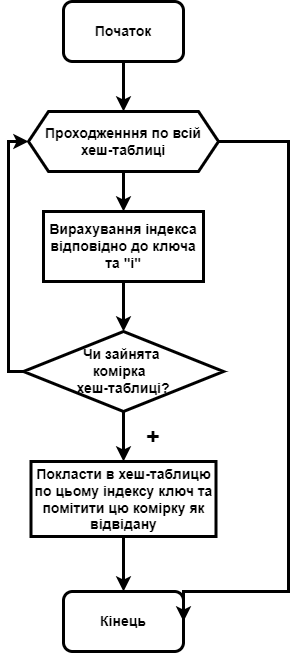


Рисунок 1 – Блок-схема для додавання елементу

Програмна реалізація функції додавання елементу:

void open\_adres::add\_elem(vector<visited\_or\_not>& main\_array, bool from\_funk\_1, int elem, bool check\_or\_not) {

int posit;

for (int i = 0; i < main\_array.size(); i++)

{

posit = hesh\_funk\_open\_adres(elem, from\_funk\_1, i, main\_array.size());

if (!main\_array[posit].was\_or) {

main\_array[posit].key = elem;

main\_array[posit].was\_or = 1;

break;

}

}

}

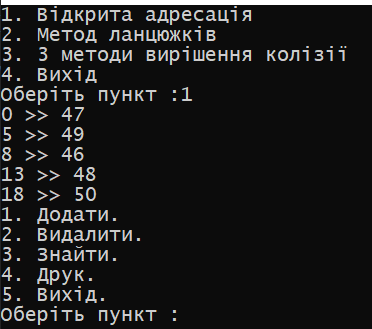


Рисунок 2 – Демонстрація роботи функції додавання елементу

Пошук

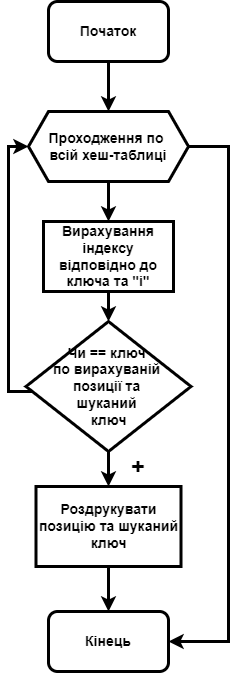


Рисунок 3 – Блок-схема для пошуку елементу

Програмна реалізація функції пошуку елемента:

void open\_adres::searche(vector<visited\_or\_not> main\_array, bool from\_funk\_1, int elem) {

int posit;

for (int i = 0; i < main\_array.size(); i++)

{

posit = hesh\_funk\_open\_adres(elem, from\_funk\_1, i, main\_array.size());

if (!main\_array[posit].was\_or) {

cout << "Такого елемента в масиві немає !!!" << endl;

break;

}

else if (main\_array[posit].key == elem) {

cout << "Элемент " << elem << " знаходиться під індексом " << posit << endl;

break;

}

}

}

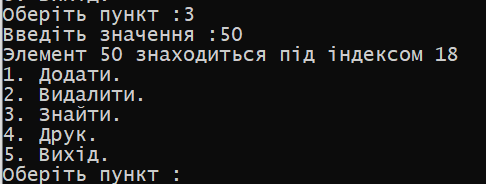


Рисунок 4 – Демонстрація роботи функції пошуку елементу

Видалення

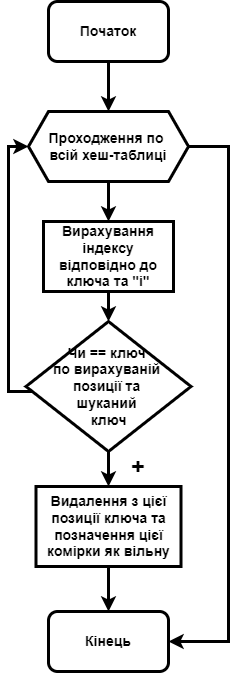


Рисунок 5 – Блок-схема для видалення елементу

Програмна реалізація функції видалення елементу з хеш-таблиці:

void open\_adres::delet(vector<visited\_or\_not>& main\_array, bool from\_funk\_1, int elem) {

int posit;

for (int i = 0; i < main\_array.size(); i++)

{

posit = hesh\_funk\_open\_adres(elem, from\_funk\_1, i, main\_array.size());

if (!main\_array[posit].was\_or) {

cout << "Такого елемента в масиві немає !!!" << endl;

break;

}

else if (main\_array[posit].key == elem) {

main\_array.erase(main\_array.begin() + posit);

break;

}

if (i == main\_array.size() - 1) {

cout << "Такого елемента в масиві немає !!!" << endl;

}

}

}

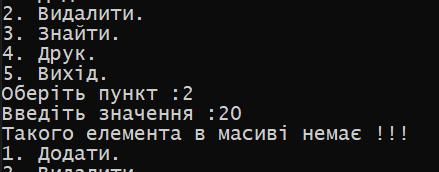


Рисунок 6 – Демонстрація роботи функції видалення елементу

Асимптотичний аналіз

Таблиця 1 – Асимптотична трудомісткість алгоритмів хеш-таблиці



Висновок

Під час виконання даної лабораторної роботи були отриманні навички роботи з основними алгоритмами роботи з хеш-таблицею: пряма адресація, відкрита адресація.

Хеш-таблиця – це структура даних, яка реалізує інтерфейс асоціативного масиву. Під час побудови хеш-таблиці може виникнути колізія, коли хеш-функції дала індекс вже зайнятої клітинки хеш-таблиці. Основними методами вирішення колізії є метод ланцюжків та прямої адресації.

Одним за методів уникнення колізії є ланцюжковий метод. Цей метод замість збільшення комірок хеш-таблиці збільшує кількість клітинок по кожному індексу хеш-таблиці, утворюючи масив списків.

Ще одним методом вирішення колізії є метод прямої адресації. Він при виникненні колізії перевіряє чи зайнята наступна комірка, якщо вона вільна, то поміщає значення ключа, в разі якщо вона вже зайнята, переходить до наступної комірки і так далі.