Лабораторна робота № 6

Тема: Реалізація хеш-таблиць та розв’язання колізій методами лінійного зондування та побудови ланцюжків.

Мета роботи: набуття практичних навичок побудови хеш-таблиць та розв’язування колізій.

Хід роботи

Завдання для виконання

1. Виконати Завдання 1 та Завдання 2**,** встановивши кількість слотів таблиці рівної кількості літер Вашого повного власного імені. Заповнити таблицю 10тю цифрами Вашого номера телефону, взявши за ключ остачу від ділення на кількість слотів. Результат роботи виводити на екран після кожних 3х цифр та відобразити у звіті.

Завдання 1. Реалізуйте клас HashTable з використанням методу лінійного зондування за наведеним кодом.

Лістинг 6.1 – Код програми

```
//Чавурська Єлизавета, П-31, АіСД, Варіант 20, ЛР6
#include <iostream>
#include <iomanip>
#include <string>
using namespace std;

class HashTable {
private:
	int* table;
	bool* occupied;
	int size;
	int hashFunction(int key) const {
		return key % size;
	}

public:
	HashTable(int s) {
		size = s;
		table = new int[size];
		occupied = new bool[size];
		for (int i = 0; i < size; i++) {
			table[i] = -1;
			occupied[i] = false;
```

```
            }
        }
        ~HashTable() {
            delete[] table;
            delete[] occupied;
        }
        void insert(int value) {
            int key = hashFunction(value);
            int index = key;
            int step = 0;

            while (occupied[index]) {
                index = (key + ++step) % size;
            }

            table[index] = value;
            occupied[index] = true;
        }

        // Вивід таблиці
        void display() const {
            cout << "\n----------------------------";
            cout << "\n Індекс | Значення ";
            cout << "\n----------------------------";
            for (int i = 0; i < size; i++) {
                cout << "\n " << setw(6) << i << " | ";
                if (occupied[i])
                    cout << table[i];
                else
                    cout << "_";
            }
            cout << "\n----------------------------\n";
        }
};

int main() {
    setlocale(LC_ALL, "ukr");
    string fullName = "Yelyzaveta Chavurska";
    int slots = 0;
    for (char c : fullName)
        if (isalpha(c)) slots++;

    cout << "Кількість слотів у таблиці: " << slots << endl;
    HashTable ht(slots);
    int phoneDigits[10] = { 0, 6, 6, 0, 8, 0, 3, 7, 3, 2 };
    int count = 0;
    for (int i = 0; i < 10; i++) {
        ht.insert(phoneDigits[i]);
        count++;

        if (count % 3 == 0) {
```

```
                cout << "\nПісля вставки " << count << " цифр(и):";
                ht.display();
            }
        }

        cout << "\nКінцевий стан таблиці:";
        ht.display();

        cout << "\nЧавурська Єлизавета, П-31, АіСД, Варіант 20, ЛР6\n";
        return 0;
}
```

```
К?льк?сть слот?в у таблиц?: 19

П?сля вставки 3 цифр(и):
--------------------------
 ?ндекс | Значення
--------------------------
      0 | 0
      1 | _
      2 | _
      3 | _
      4 | _
      5 | _
      6 | 6
      7 | 6
      8 | _
      9 | _
     10 | _
     11 | _
     12 | _
     13 | _
     14 | _
     15 | _
     16 | _
     17 | _
     18 | _
--------------------------

П?сля вставки 6 цифр(и):
--------------------------
 ?ндекс | Значення
--------------------------
      0 | 0
      1 | 0
      2 | 0
      3 | _
      4 | _
      5 | _
      6 | 6
      7 | 6
      8 | 8
      9 | _
     10 | _
     11 | _
     12 | _
     13 | _
     14 | _
     15 | _
     16 | _
     17 | _
     18 | _
--------------------------
```

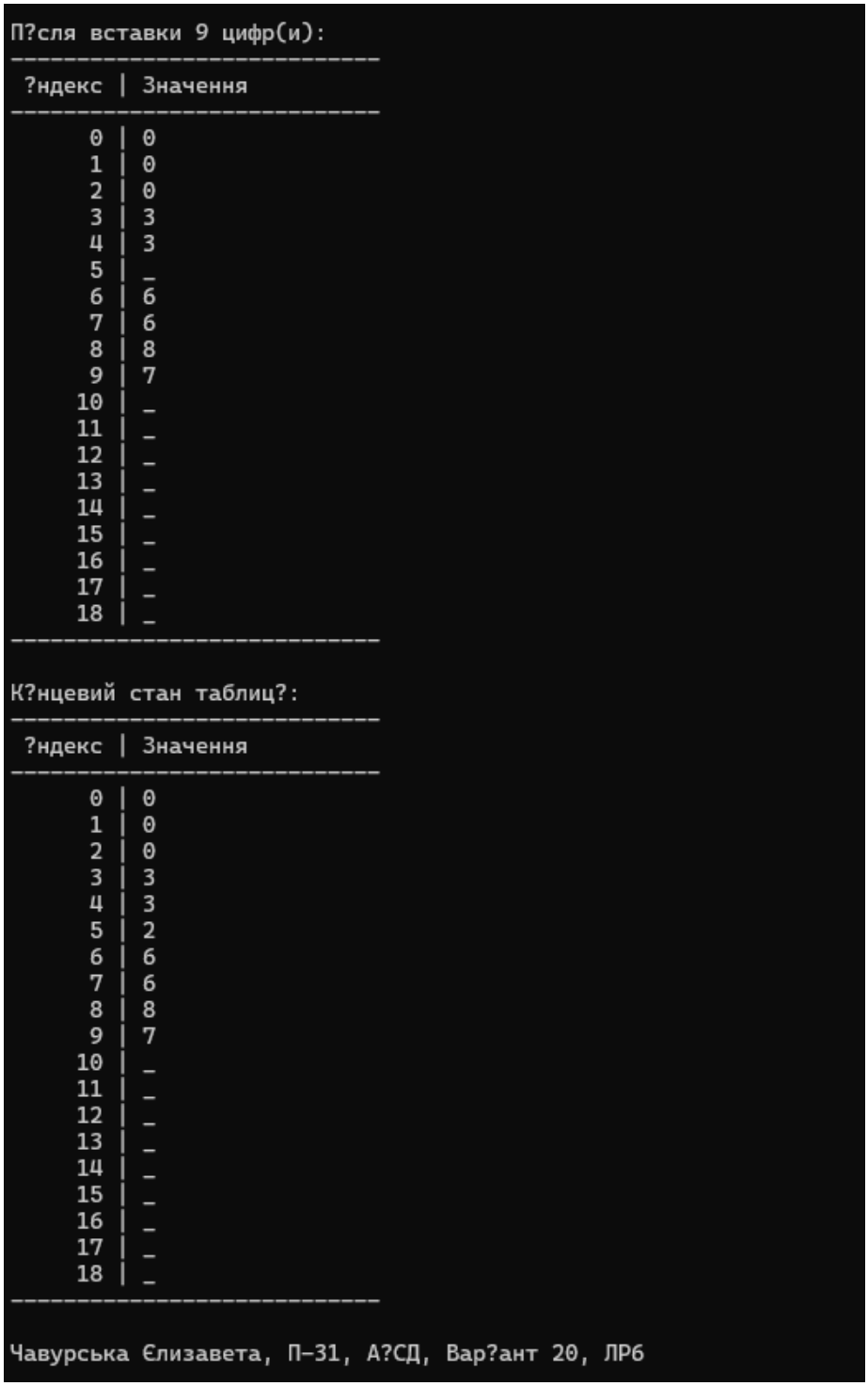

Рисунок 6.1 – Вікно виконання програми

Завдання 2. Реалізуйте перевантаження спеціальних методів для класу HashTable за наведеним кодом.

Лістинг 6.2 – Код програми

```
// Чавурська Єлизавета, П-31, АіСД, Варіант 20, Лр 6
#include <iostream>
#include <string>
using namespace std;
```

```
class HashTable {
private:
	struct Entry {
		int key;
		int value;
		bool isEmpty;
		bool isDeleted;
	};
	Entry* table;
	int capacity;
	int current_size;

	int hashFunction(int key) const {
		return key % capacity;
	}
public:
	HashTable(int cap) {
		capacity = cap;
		current_size = 0;
		table = new Entry[capacity];
		for (int i = 0; i < capacity; i++) {
			table[i].isEmpty = true;
			table[i].isDeleted = false;
		}
	}

	~HashTable() {
		delete[] table;
	}

	void put(int key, int value) {
		int index = hashFunction(key);
		int start = index;

		while (!table[index].isEmpty && !table[index].isDeleted &&
table[index].key != key) {
			index = (index + 1) % capacity;
			if (index == start) {
				cout << "Таблиця заповнена!\n";
				return;
			}
		}

		if (table[index].isEmpty || table[index].isDeleted)
			current_size++;

		table[index].key = key;
		table[index].value = value;
		table[index].isEmpty = false;
		table[index].isDeleted = false;
	}
```

```
    void print() const {
        cout << "Ключі:     ";
        for (int i = 0; i < capacity; i++) {
            if (!table[i].isEmpty && !table[i].isDeleted)
                cout << table[i].key << "\t";
            else
                cout << "_\t";
        }
        cout << "\nЗначення:  ";
        for (int i = 0; i < capacity; i++) {
            if (!table[i].isEmpty && !table[i].isDeleted)
                cout << table[i].value << "\t";
            else
                cout << "_\t";
        }
        cout << "\n-----------------------------------------\n";
    }
};

int main() {
    setlocale(LC_CTYPE, "Ukr");
    string fullName = "Chavurska Yelyzaveta"; // 18 літер
    int slots = fullName.length();
    cout << "Кількість слотів у таблиці = " << slots << endl;
    int phoneDigits[10] = { 0, 9, 6, 7, 5, 4, 3, 2, 1, 0 };
    HashTable ht(slots);

    for (int i = 0; i < 10; i++) {
        int value = phoneDigits[i];
        int key = value % slots; // ключ = остача від ділення
        ht.put(key, value);

        if ((i + 1) % 3 == 0 || i == 9) {
            cout << "\nПісля вставки " << i + 1 << " цифр:\n";
            ht.print();
        }
    }
    cout << "Чавурська Єлизавета, П-31, АіСД, Варіант 20, Лр 6\n";
    return 0;
}
```

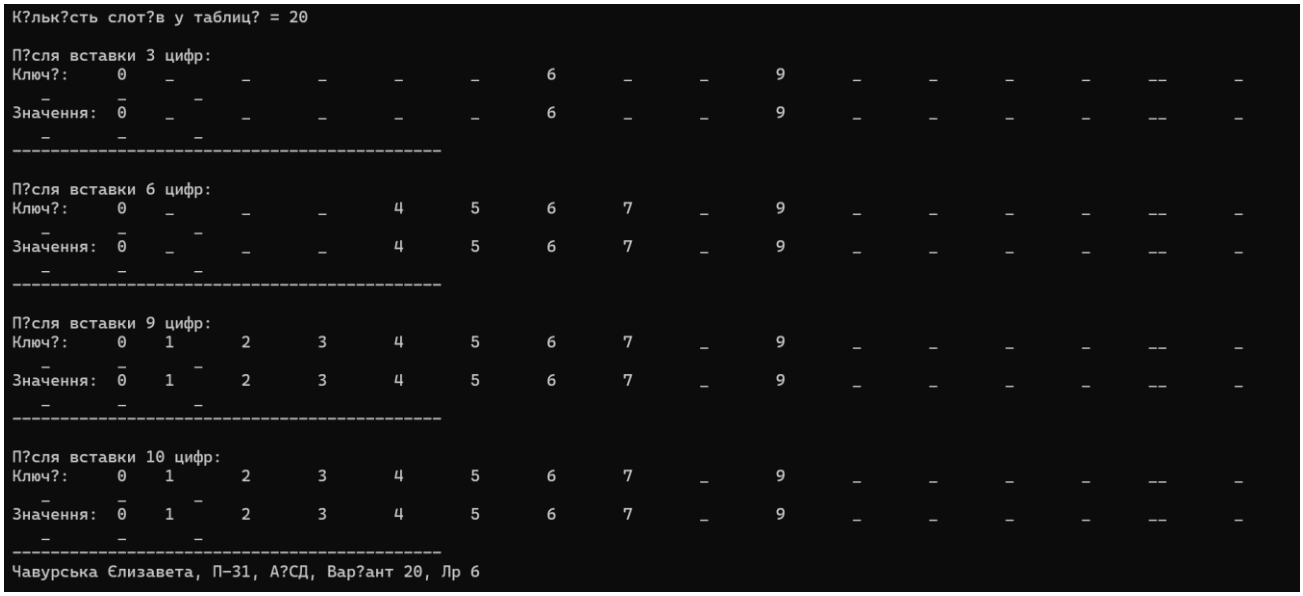

Рисунок 6.2 – Вікно результатів програми

2. Виконати Завдання 3* та перевірити роботу програми, видаливши кілька ключів. Результат роботи вивести на екран та занести у звіт **(додатково).**

Завдання 3*. Запропонуйте та реалізуйте метод видалення ключів у хеш-таблиці з відкритою адресацією

Лістинг 6.3 – Код програми

```
// Чавурська Єлизавета, П-31, АіСД, Варіант 20, Лр 6
#include <iostream>
#include <string>
using namespace std;

class HashTable {
private:
	struct Entry {
		int key;
		int value;
		bool isEmpty;
		bool isDeleted;
	};

	Entry* table;
	int capacity;

	int hashFunction(int key) const {
		return key % capacity;
	}

public:
	// Конструктор
	HashTable(int cap) {
```

```
            capacity = cap;
            table = new Entry[capacity];
            for (int i = 0; i < capacity; i++) {
                  table[i].isEmpty = true;
                  table[i].isDeleted = false;
            }
      }

      // Деструктор
      ~HashTable() {
            delete[] table;
      }

      // Додавання елементів (лінійне зондування)
      void put(int key, int value) {
            int index = hashFunction(key);
            int start = index;

            while (!table[index].isEmpty && !table[index].isDeleted &&
table[index].key != key) {
                  index = (index + 1) % capacity;
                  if (index == start) {
                        cout << "Таблиця заповнена!\n";
                        return;
                  }
            }

            table[index].key = key;
            table[index].value = value;
            table[index].isEmpty = false;
            table[index].isDeleted = false;
      }

      // Пошук значення за ключем
      int get(int key) {
            int index = hashFunction(key);
            int start = index;

            while (!table[index].isEmpty) {
                  if (!table[index].isDeleted && table[index].key == key)
                        return table[index].value;

                  index = (index + 1) % capacity;
                  if (index == start)
                        break;
            }
            cout << "Ключ " << key << " не знайдено!\n";
            return -1;
      }

      // Метод видалення елемента за ключем
```

```
    void remove(int key) {
        int index = hashFunction(key);
        int start = index;

        while (!table[index].isEmpty) {
            if (!table[index].isDeleted && table[index].key == key) {
                table[index].isDeleted = true;
                cout << "Ключ " << key << " видалено.\n";
                return;
            }
            index = (index + 1) % capacity;
            if (index == start)
                break;
        }
        cout << "Ключ " << key << " не знайдено для видалення.\n";
    }

    // Виведення таблиці
    void print() const {
        cout << "Ключі:     ";
        for (int i = 0; i < capacity; i++) {
            if (!table[i].isEmpty && !table[i].isDeleted)
                cout << table[i].key << "\t";
            else
                cout << "_\t";
        }
        cout << "\nЗначення:  ";
        for (int i = 0; i < capacity; i++) {
            if (!table[i].isEmpty && !table[i].isDeleted)
                cout << table[i].value << "\t";
            else
                cout << "_\t";
        }
        cout << "\n";
    }
};

int main() {
    setlocale(LC_ALL, "ukr");

    string fullName = "Chavurska Yelyzaveta";
    int slots = fullName.length();
    cout << "Кількість слотів у таблиці = " << slots << endl;

    int number[10] = { 0, 6, 6, 0, 8, 0, 3, 7, 3, 2 };

    HashTable ht(slots);

    // Додавання чисел у таблицю
    for (int i = 0; i < 10; i++) {
        int value = number[i];
```

```
            int key = value % slots; // ключ = остача від ділення
            ht.put(key, value);
            if ((i + 1) % 3 == 0 || i == 9) {
                  cout << "\nПісля вставки " << i + 1 << " цифр:\n";
                  ht.print();
            }
      }

      // Перевірка видалення кількох ключів
      cout << "\n--- Видалення кількох ключів ---\n";
      ht.remove(3);  // видаляємо ключ 3
      ht.remove(8);  // видаляємо ключ 8
      ht.remove(0);  // видаляємо ключ 0

      cout << "\nПісля видалення:\n";
      ht.print();

      cout << "\nЧавурська Єлизавета, П-31, АіСД, Варіант 20, ЛР6\n";
      return 0;
}
```

```
К?льк?сть слот?в у таблиц? = 20

П?сля вставки 3 цифр:
Ключ?:     0    _    _    _    _    _    6    _    _    _    _    _    _    _
Значення:  0    _    _    _    _    _    6    _    _    _    _    _    _    _

П?сля вставки 6 цифр:
Ключ?:     0    _    _    _    _    _    6    _    8    _    _    _    _    _
Значення:  0    _    _    _    _    _    6    _    8    _    _    _    _    _

П?сля вставки 9 цифр:
Ключ?:     0    _    _    3    _    _    6    7    8    _    _    _    _    _
Значення:  0    _    _    3    _    _    6    7    8    _    _    _    _    _

П?сля вставки 10 цифр:
Ключ?:     0    _    2    3    _    _    6    7    8    _    _    _    _    _
Значення:  0    _    2    3    _    _    6    7    8    _    _    _    _    _

--- Видалення к?лькох ключ?в ---
Ключ 3 видалено.
Ключ 8 видалено.
Ключ 0 видалено.

П?сля видалення:
Ключ?:     _    _    2    _    _    _    6    7    _    _    _    _    _    _
Значення:  _    _    2    _    _    _    6    7    _    _    _    _    _    _

Чавурська Єлизавета, П-31, А?СД, Вар?ант 20, ЛР6
```

Рисунок 6.3 – Вікно результатів програми

3. Виконати Завдання 4** та перевірити роботу алгоритму при досягненні 50%, 75% та 90% заповнення. Результати занести у звіт **(дуже додатково).**

Лістинг 6.4 – Код програми

```
// Чавурська Єлизавета, П-31, АіСД, Варіант 20, Лр 6
#include <iostream>
#include <string>
using namespace std;

class HashTable {
```

```
private:
	struct Entry {
		int key;
		int value;
		bool isEmpty;
		bool isDeleted;
	};

	Entry* table;
	int capacity;
	int current_size;
	double loadThreshold;

	int hashFunction(int key) const {
		return key % capacity;
	}
	void rehash() {
		int oldCapacity = capacity;
		capacity *= 2; // подвоюємо кількість слотів
		cout << "\n Перевищено поріг. Збільшуємо таблицю до " <<
capacity << " слотів.\n";

		Entry* oldTable = table;
		table = new Entry[capacity];

		for (int i = 0; i < capacity; i++) {
			table[i].isEmpty = true;
			table[i].isDeleted = false;
		}

		current_size = 0;

		// Повторно вставляємо старі значення
		for (int i = 0; i < oldCapacity; i++) {
			if (!oldTable[i].isEmpty && !oldTable[i].isDeleted) {
				put(oldTable[i].key, oldTable[i].value);
			}
		}

		delete[] oldTable;
	}

public:
	// Конструктор
	HashTable(int cap, double threshold = 0.75) {
		capacity = cap;
		current_size = 0;
		loadThreshold = threshold;
		table = new Entry[capacity];
		for (int i = 0; i < capacity; i++) {
			table[i].isEmpty = true;
```

```
                table[i].isDeleted = false;
            }
        }

        ~HashTable() {
            delete[] table;
        }

        double loadFactor() const {
            return (double)current_size / capacity;
        }

        void put(int key, int value) {
            // Якщо перевищено поріг — збільшуємо таблицю
            if (loadFactor() > loadThreshold) {
                rehash();
            }

            int index = hashFunction(key);
            int start = index;

            while (!table[index].isEmpty && !table[index].isDeleted &&
table[index].key != key) {
                index = (index + 1) % capacity;
                if (index == start) {
                    cout << "Таблиця повна!\n";
                    return;
                }
            }

            if (table[index].isEmpty || table[index].isDeleted)
                current_size++;

            table[index].key = key;
            table[index].value = value;
            table[index].isEmpty = false;
            table[index].isDeleted = false;
        }

        void print() const {
            cout << "\nКлючі:     ";
            for (int i = 0; i < capacity; i++) {
                if (!table[i].isEmpty && !table[i].isDeleted)
                    cout << table[i].key << "\t";
                else
                    cout << "_\t";
            }
            cout << "\nЗначення:  ";
            for (int i = 0; i < capacity; i++) {
                if (!table[i].isEmpty && !table[i].isDeleted)
                    cout << table[i].value << "\t";
```

```
                else
                        cout << "_\t";
            }
            cout << "\nLoad factor = " << loadFactor() * 100 << "%\n";
      }
};

int main() {
      setlocale(LC_ALL, "ukr");

      string fullName = "Chavurska Yelyzaveta";
      int slots = fullName.length(); // початковий розмір
      cout << "Початкова кількість слотів: " << slots << endl;

      // Створюємо таблицю з початковим порогом 0.5 (50%)
      HashTable ht(slots, 0.5);

      int number[10] = { 0, 6, 6, 0, 8, 0, 3, 7, 3, 2 };

      for (int i = 0; i < 10; i++) {
            int value = number[i];
            int key = value % slots;
            ht.put(key, value);
            cout << "\nПісля вставки " << i + 1 << " елементів:\n";
            ht.print();
      }

      cout << "\nТест із порогами 50%, 75% та 90% заповнення
завершено.\n";
      cout << "Чавурська Єлизавета, П-31, АіСД, Варіант 20, ЛР6\n";
      return 0;
}
```

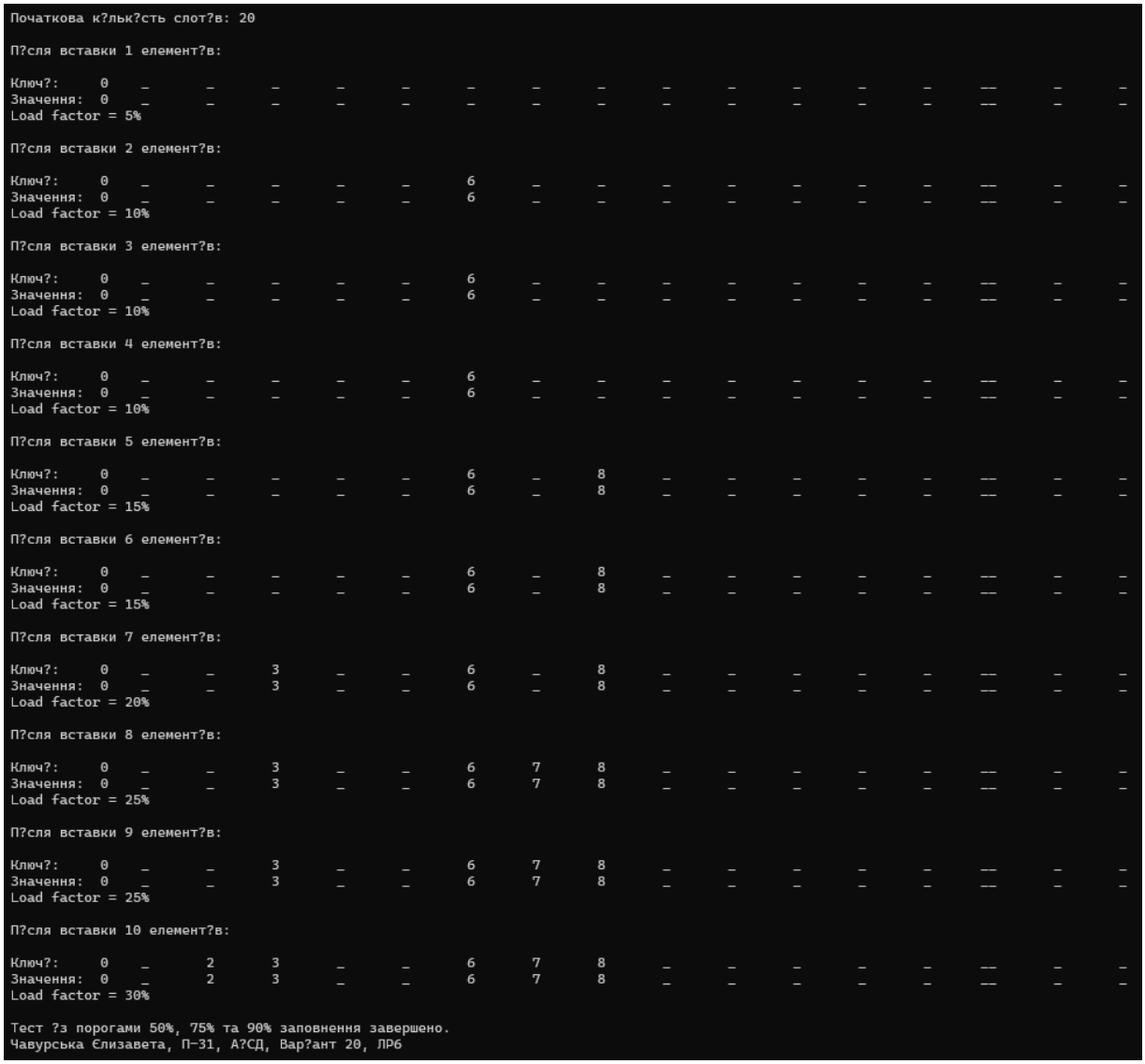

Рисунок 6.4 – Вікно результатів програми

4. Виконати Завдання 5 та Завдання 6, встановивши кількість слотів таблиці рівної кількості літер Вашого прізвища. Заповнити таблицю літерами літерами Вашого прізвища, імені та по батькові, взявши за ключ остачу від ділення Unicode літери на кількість слотів. Результат роботи виводити на екран після кожних 5ти символів та відобразити у звіті.

Завдання 5. Реалізуйте клас HashTableChain з використанням методу ланцюжків за наведеним кодом.

Лістинг 6.5 – Код програми

```
// Чавурська Єлизавета, П-31, АіСД, Варіант 20, Лр 6
#include <iostream>
#include <string>
using namespace std;
```

```
class HashTable {
private:
	// Вузол списку
	struct Node {
		char value;
		Node* next;
	};

	Node** table;  // масив вказівників на ланцюжки
	int capacity;

	// Хеш-функція
	int Func(int key) const {
		return key % capacity;
	}

public:
	// Конструктор
	HashTable(int cap) {
		capacity = cap;
		table = new Node*[capacity];
		for (int i = 0; i < capacity; i++) {
			table[i] = nullptr;
		}
	}

	// Деструктор
	~HashTable() {
		for (int i = 0; i < capacity; i++) {
			Node* current = table[i];
			while (current) {
				Node* temp = current;
				current = current->next;
				delete temp;
			}
		}
		delete[] table;
	}

	// Додавання символу до таблиці
	void put(char value) {
		int key = (int)value;
		int index = Func(key);

		Node* newNode = new Node;
		newNode->value = value;
		newNode->next = nullptr;

		if (table[index] == nullptr) {
			table[index] = newNode;
		}
```

```
            else {
                  Node* current = table[index];
                  while (current->next != nullptr)
                        current = current->next;
                  current->next = newNode;
            }
      }

      // Виведення таблиці
      void print() const {
            cout << "\n-------------------------";
            cout << "\nІндекс | Ланцюжок символів";
            cout << "\n-------------------------\n";
            for (int i = 0; i < capacity; i++) {
                  cout << i << ":\t";
                  Node* current = table[i];
                  if (!current) cout << "_";
                  while (current) {
                        cout << current->value;
                        if (current->next) cout << " -> ";
                        current = current->next;
                  }
                  cout << "\n";
            }
            cout << "-------------------------\n";
      }
};

int main() {
      setlocale(LC_ALL, "ukr");

      string surname = "Chavurska";
      int slots = surname.length(); // кількість слотів дорівнює кількості
літер у прізвищі
      cout << "Кількість слотів у таблиці: " << slots << endl;

      string fullName = "Chavurska Yelyzaveta";
      HashTable ht(slots);

      // Заповнюємо таблицю літерами
      for (int i = 0; i < fullName.length(); i++) {
            char ch = fullName[i];
            if (ch == ' ') continue; // пропускаємо пробіли

            ht.put(ch);

            // Вивід після кожних 5 символів
            if ((i + 1) % 5 == 0 || i == fullName.length() - 1) {
                  cout << "\nПісля вставки " << i + 1 << " символів:";
                  ht.print();
            }
```

```
    }

    cout << "\nЧавурська Єлизавета, П-31, АіСД, Варіант 20, ЛР6\n";
    return 0;
}
```

```
К?льк?сть слот?в у таблиц?: 9

П?сля вставки 5 символ?в:
------------------------
?ндекс | Ланцюжок символ?в
------------------------
0:      u
1:      v
2:      _
3:      _
4:      C
5:      h
6:      _
7:      a
8:      _
------------------------

П?сля вставки 15 символ?в:
------------------------
?ндекс | Ланцюжок символ?в
------------------------
0:      u -> l
1:      v
2:      e
3:      _
4:      C -> y
5:      h -> z
6:      r
7:      a -> s -> a
8:      k -> Y
------------------------

П?сля вставки 20 символ?в:
------------------------
?ндекс | Ланцюжок символ?в
------------------------
0:      u -> l
1:      v -> v
2:      e -> e
3:      _
4:      C -> y
5:      h -> z
6:      r
7:      a -> s -> a -> a -> a
8:      k -> Y -> t
------------------------

Чавурська Єлизавета, П-31, А?СД, Вар?ант 20, ЛР6
```

Рисунок 6.5 – Вікно результатів програми

Завдання 6. Реалізуйте перевантаження спеціальних методів для класу HashTableChain

Лістинг 6.6 – Код програми

```
//Чавурська Єлизавета, П-31, АіСД, Варіант 20, ЛР6
#include <iostream>
#include <string>
using namespace std;

class HashTableChain {
private:
	struct Node {
		char value;
		Node* next;
	};

	Node** table;  // масив списків
	int capacity;

	int hashFunction(int key) const {
		return key % capacity;
	}

	// Допоміжна функція для копіювання ланцюжків
	Node* copyList(Node* src) {
		if (!src) return nullptr;
		Node* newHead = new Node{ src->value, nullptr };
		Node* currentSrc = src->next;
		Node* currentDest = newHead;

		while (currentSrc) {
			currentDest->next = new Node{ currentSrc->value, nullptr
};
			currentSrc = currentSrc->next;
			currentDest = currentDest->next;
		}

		return newHead;
	}

public:
	// Звичайний конструктор
	HashTableChain(int cap) {
		capacity = cap;
		table = new Node*[capacity];
		for (int i = 0; i < capacity; i++) {
			table[i] = nullptr;
		}
	}
```

```
//  Конструктор копіювання
HashTableChain(const HashTableChain& other) {
	capacity = other.capacity;
	table = new Node*[capacity];
	for (int i = 0; i < capacity; i++) {
		table[i] = copyList(other.table[i]);
	}
	cout << "\n[Конструктор копіювання викликано]\n";
}

//  Оператор присвоєння
HashTableChain& operator=(const HashTableChain& other) {
	if (this == &other) return *this;

	// очищення поточної таблиці
	for (int i = 0; i < capacity; i++) {
		Node* current = table[i];
		while (current) {
			Node* temp = current;
			current = current->next;
			delete temp;
		}
	}
	delete[] table;

	capacity = other.capacity;
	table = new Node*[capacity];
	for (int i = 0; i < capacity; i++) {
		table[i] = copyList(other.table[i]);
	}

	cout << "\n[Оператор присвоєння викликано]\n";
	return *this;
}

// Деструктор
~HashTableChain() {
	for (int i = 0; i < capacity; i++) {
		Node* current = table[i];
		while (current) {
			Node* temp = current;
			current = current->next;
			delete temp;
		}
	}
	delete[] table;
}

// Додавання символа
void put(char value) {
	int key = (int)value;
```

```
            int index = hashFunction(key);

            Node* newNode = new Node{ value, nullptr };
            if (table[index] == nullptr)
                  table[index] = newNode;
            else {
                  Node* current = table[index];
                  while (current->next != nullptr)
                        current = current->next;
                  current->next = newNode;
            }
      }

      // Вивід таблиці
      void print() const {
            cout << "\n------------------------------";
            cout << "\nІндекс | Ланцюжок символів";
            cout << "\n------------------------------\n";
            for (int i = 0; i < capacity; i++) {
                  cout << i << ":\t";
                  Node* current = table[i];
                  if (!current) cout << "_";
                  while (current) {
                        cout << current->value;
                        if (current->next) cout << " -> ";
                        current = current->next;
                  }
                  cout << "\n";
            }
            cout << "------------------------------\n";
      }
};

int main() {
      setlocale(LC_ALL, "ukr");

      string surname = "Chavurska";
      int slots = surname.length();  // кількість слотів = кількість літер
у прізвищі
      cout << "Кількість слотів у таблиці: " << slots << endl;

      string fullName = "Chavurska Yelyzaveta";
      HashTableChain ht(slots);

      // Заповнення таблиці літерами
      int count = 0;
      for (char ch : fullName) {
            if (ch == ' ') continue;  // пропуск пробілів
            ht.put(ch);
            count++;
```

```
            if (count % 5 == 0) {
                cout << "\nПісля вставки " << count << " символів:";
                ht.print();
            }
        }

        cout << "\n[Копіюємо таблицю в інший об’єкт...]\n";
        HashTableChain copyHt = ht; // виклик конструктора копіювання

        cout << "\n[Присвоюємо таблицю іншій змінній...]\n";
        HashTableChain assignedHt(slots);
        assignedHt = ht; // виклик оператора присвоєння

        cout << "\nОригінал:";
        ht.print();

        cout << "\nКопія:";
        copyHt.print();

        cout << "\nПрисвоєна таблиця:";
        assignedHt.print();

        cout << "\nЧавурська Єлизавета, П-31, АіСД, Варіант 20, ЛР6\n";
        return 0;
}
```

```
К?льк?сть слот?в у таблиц?: 9

П?сля вставки 5 символ?в:
------------------------------
?ндекс | Ланцюжок символ?в
------------------------------
0:      u
1:      v
2:      _
3:      _
4:      C
5:      h
6:      _
7:      a
8:      _
------------------------------

П?сля вставки 10 символ?в:
------------------------------
?ндекс | Ланцюжок символ?в
------------------------------
0:      u
1:      v
2:      _
3:      _
4:      C
5:      h
6:      r
7:      a -> s -> a
8:      k -> Y
------------------------------

П?сля вставки 15 символ?в:
------------------------------
?ндекс | Ланцюжок символ?в
------------------------------
0:      u -> l
1:      v
2:      e
3:      _
4:      C -> y
5:      h -> z
6:      r
7:      a -> s -> a -> a
8:      k -> Y
------------------------------
```

Рисунок 6.6 – Вікно результатів програми

5. Зробити висновок про роботу та записати у звіт.

6. Бути готовим до захисту.

Висновок: на цій лабораторній роботі я набула практичних навичок побудови хеш-таблиць та розв’язування колізій.