Міністерство освіти і науки України
КПІ ім. Ігоря Сікорського
Кафедра ІПІ

ЗВІТ

з виконання лабораторної роботи № 6
з кредитного модуля
“Основи програмування-2. Методології програмування”

Варіант № 21

Виконав:
студент 1-го курсу
гр. ІП-22 ФІОТ
Патріюк Юрій Олексійович

Київ 2023

ПОСТАНОВКА ЗАДАЧІ

21. Спроектувати АТД "Хеш-таблиця з ланцюжками колізій" для контейнера, що містить дані довільного типу. Інтерфейс АТД включає такі операції: визначення розміру таблиці, визначення пустоти таблиці, вставка елемента за ключем, видалення елемента за ключем, ітератор для доступу до елементів таблиці з операціями:

1) встановлення початку таблиці
2) перевірка кінця таблиці
3) доступ до даних поточного елемента таблиці
4) перехід до наступного елементу таблиці

ТЕКСТ ПРОГРАМИ

Вміст файлу main.cpp

```
#include "MainFunctions.h"
int main() {
    int size = 0;
    cout<<"Enter size of hash table:";
    cin>>size;
    string type;
    cout<<"Enter variable type(int,string,float):";
    cin>>type;

    if(type == "string")
        stringMenuCode(size);
    else if(type == "int")
        intMenuCode(size);
    else if(type == "float")
        floatMenuCode(size);
    else
    {
        cout<<"Incorrect entered type"<<endl;
        return 0;
    }

    return 0;
}
```

Вміст файлу HashTable.h

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

template <typename T>
struct Node {
    string key;
    T value;
```

```
    Node* next;
    Node(const string& k, const T& v) : key(k), value(v), next(nullptr) {}
};

template <typename T>
class HashTable {
private:
    Node<T>** table;
    int tableSize;
    int amountOfElements = 0;

public:
    class Iterator {
    private:
        const HashTable& hashTable;
        int currentBucket;
        Node<T>* currentNode;

    public:
        Iterator(const HashTable& ht, int bucket, Node<T>* node) : hashTable(ht),
currentBucket(bucket), currentNode(node) {}
        bool operator!=(const Iterator& other) const {
            return currentNode != other.currentNode;
        }
        Iterator& operator++() {
            if (currentNode->next != nullptr) {
                currentNode = currentNode->next;
            }
            else {
                ++currentBucket;
                while (currentBucket < hashTable.tableSize &&
hashTable.table[currentBucket] == nullptr) {
                    ++currentBucket;
                }
                if (currentBucket < hashTable.tableSize)
                    currentNode = hashTable.table[currentBucket];
                else
                    currentNode = nullptr;
            }
            return *this;
        }
        Node<T>* operator->() const {
            return currentNode;
        }
        bool isEnd() const
        {
            return currentNode == nullptr;
        }
    };

    HashTable(int size);
    ~HashTable();
    void add(const string& key, const T& value);
    void del(const string& key);

    Iterator begin() const {
        int bucket = 0;
        while (bucket < tableSize && table[bucket] == nullptr) {
            ++bucket;
        }
        if (bucket < tableSize) {
            return Iterator(*this, bucket, table[bucket]);
        } else {
            return end();
```

```
        }
    }
    Iterator end() const {
        return Iterator(*this, tableSize, nullptr);
    }

    void printChain(Node<T>* head) const;
    void print() const;
    int getSize() const;
    float getFullnessCoefficient() const;

private:
    int hashFunction(const string& key) const;
};
```

Вміст файлу HashTable.cpp

```
#include "HashTable.h"

template <typename T>
HashTable<T>::HashTable(int size): tableSize(size) {
    table = new Node<T>*[tableSize];
    for (int i = 0; i < tableSize; ++i) {
        table[i] = nullptr;
    }
}

template<typename T>
HashTable<T>::~HashTable() {
    for (int i = 0; i < tableSize; ++i) {
        Node<T>* currentNode = table[i];
        while (currentNode != nullptr) {
            Node<T>* nextNode = currentNode->next;
            delete currentNode;
            currentNode = nextNode;
        }
    }
    delete[] table;
}

template<typename T>
void HashTable<T>::add(const string &key, const T &value)
{
    int bucket = hashFunction(key);
    Node<T>* newNode = new Node<T>(key, value);
    if (table[bucket] == nullptr) {
        table[bucket] = newNode;
    } else {
        Node<T>* currentNode = table[bucket];
        while (currentNode->next != nullptr) {
            currentNode = currentNode->next;
        }
        currentNode->next = newNode;
    }
    amountOfElements++;
}

template<typename T>
void HashTable<T>::del(const string &key)  {
    int bucket = hashFunction(key);
    Node<T>* currentNode = table[bucket];
    Node<T>* prevNode = nullptr;
```

```
    while (currentNode != nullptr && currentNode->key != key) {
        prevNode = currentNode;
        currentNode = currentNode->next;
    }

    if (currentNode == nullptr) {
        return;
    }

    if (prevNode == nullptr) {
        table[bucket] = currentNode->next;
    } else {
        prevNode->next = currentNode->next;
    }

    delete currentNode;
    amountOfElements--;
}

template <typename T>
void HashTable<T>::printChain(Node<T> *head) const  {
    if(head == NULL) {
        cout << endl;
        return;
    }
    cout<<'['<<head->key<<':'<<head->value<<']';
    if(head->next != NULL)
        cout<<"->";
    printChain(head->next);
}

template <typename T>
void HashTable<T>::print() const {
    for (int i = 0; i < tableSize; ++i) {
        cout<<i<<':';
        printChain(table[i]);
    }
    cout<<endl;
}

template<typename T>
int HashTable<T>::hashFunction(const string&key) const  {
    return hash<string>{}(key) % tableSize;
}

template<typename T>
float HashTable<T>::getFullnessCoefficient() const {
    return (float)amountOfElements/tableSize;
}

template<typename T>
int HashTable<T>::getSize() const {
    return this->tableSize;
}
```

Вміст файлу MainFunctions.h

```
#include "HashTable.cpp"

void intMenuCode(int size);
void stringMenuCode(int size);
void floatMenuCode(int size);
int showUserMenu();
```

Вміст файлу MainFunctions.cpp

```
#include "MainFunctions.h"
#include <iomanip>

void intMenuCode(int size)
{
    HashTable<int> hashTable(size);
    int menuChoose = 0;
    do {
        menuChoose = showUserMenu();
        switch(menuChoose)
        {
            case 1:
            {
                char choice = 'y';
                do {
                    if (choice == 'y') {
                        string key;
                        int elem;
                        cout << "Enter key(string):";
                        cin >> key;
                        cout << "Enter value:";
                        cin >> elem;
                        hashTable.add(key, elem);
                    } else if (choice != 'n')
                        cout << "Incorrect choice!" << endl;
                    cout << "Do you want to add element (y/n):";
                    cin >> choice;
                } while (choice != 'n');
                cout<<endl;
            }
                break;
            case 2:
            {
                string key;
                cout<<"Enter key:";
                cin>>key;
                hashTable.del(key);
                cout<<"Done"<<endl<<endl;
            }
                break;
            case 3:
                hashTable.print();
            break;
            case 4:
            {
                for (auto item = hashTable.begin(); item != hashTable.end(); ++item)
{
                    cout << "Key: " << item->key << " Value: " << item->value <<
endl;
                }
                cout<<endl;
```

```
            }
                break;
            case 5:
                cout << "Fullness coefficient: " << setprecision(3)<<
hashTable.getFullnessCoefficient() << endl<<endl;
                break;
            case 6:
                cout<<"Size of HashTable: "<<hashTable.getSize()<<endl<<endl;
                break;
            default:
                break;
        }
    }while(menuChoose!=0);
}

void stringMenuCode(int size)
{
    HashTable<string> hashTable(size);
    int menuChoose = 0;
    do {
        menuChoose = showUserMenu();
        switch(menuChoose)
        {
            case 1:
            {
                char choice = 'y';
                do {
                    if (choice == 'y') {
                        string key;
                        string elem;
                        cout << "Enter key(string):";
                        cin >> key;
                        cout << "Enter value:";
                        cin >> elem;
                        hashTable.add(key, elem);
                    } else if (choice != 'n')
                        cout << "Incorrect choice!" << endl;
                    cout << "Do you want to add element (y/n):";
                    cin >> choice;
                } while (choice != 'n');
                cout<<endl;
            }
                break;
            case 2:
            {
                string key;
                cout<<"Enter key:";
                cin>>key;
                hashTable.del(key);
                cout<<"Done"<<endl<<endl;
            }
                break;
            case 3:
                hashTable.print();
                break;
            case 4:
            {
                for (auto item = hashTable.begin(); item != hashTable.end(); ++item)
{
                    cout << "Key: " << item->key << " Value: " << item->value <<
endl;
                }
                cout<<endl;
            }
```

```
                break;
            case 5:
                cout << "Fullness coefficient: "
<<setprecision(3)<<hashTable.getFullnessCoefficient() << endl<<endl;
                break;
            case 6:
                cout<<"Size of HashTable: "<<hashTable.getSize()<<endl<<endl;
                break;
            default:
                break;
        }
    }while(menuChoose!=0);
}

void floatMenuCode(int size)
{
    HashTable<float> hashTable(size);
    int menuChoose = 0;
    do {
        menuChoose = showUserMenu();
        switch(menuChoose)
        {
            case 1:
            {
                char choice = 'y';
                do {
                    if (choice == 'y') {
                        string key;
                        float elem;
                        cout << "Enter key(string):";
                        cin >> key;
                        cout << "Enter value:";
                        cin >> elem;
                        hashTable.add(key, elem);
                    } else if (choice != 'n')
                        cout << "Incorrect choice!" << endl;
                    cout << "Do you want to add element (y/n):";
                    cin >> choice;
                } while (choice != 'n');
                cout<<endl;
            }
                break;
            case 2:
            {
                string key;
                cout<<"Enter key:";
                cin>>key;
                hashTable.del(key);
                cout<<"Done"<<endl<<endl;
            }
                break;
            case 3:
                hashTable.print();
                break;
            case 4:
            {
                for (auto item = hashTable.begin(); item != hashTable.end(); ++item)
{
                    cout << "Key: " << item->key << " Value: " << item->value <<
endl;
                }
                cout<<endl;
            }
                break;
```

```
            case 5:
                cout << "Fullness coefficient: " <<
setprecision(3)<<hashTable.getFullnessCoefficient() << endl<<endl;
                break;
            case 6:
                cout<<"Size of HashTable: "<<hashTable.getSize()<<endl<<endl;
                break;
            default:
                break;
        }
    }while(menuChoose!=0);
}

int showUserMenu()
{
    int choose = 0;
    cout<<"\tMenu"<<endl;
    cout<<"1 - Add Element"<<endl;
    cout<<"2 - Delete Element"<<endl;
    cout<<"3 - Show HashTable"<<endl;
    cout<<"4 - Show HashTable using iterator"<<endl;
    cout<<"5 - Show Fullness Coefficient"<<endl;
    cout<<"6 - Show Size Of Table"<<endl;
    cout<<"0 - Exit"<<endl;
    cout<<"Choose:";
    cin>>choose;
    return choose;
}
```

РЕЗУЛЬТАТ ТЕСТУВАННЯ

```
Enter size of hash table:3
Enter variable type(int,string,float):string
 Menu
1 - Add Element
2 - Delete Element
3 - Show HashTable
4 - Show HashTable using iterator
5 - Show Fullness Coefficient
6 - Show Size Of Table
0 - Exit
Choose:1
Enter key(string):qwe
Enter value:qwe
Do you want to add element (y/n):y
Enter key(string):asd
Enter value:asd
Do you want to add e
lement (y/n):y
Enter key(string):zxc
Enter value:zxc
Do you want to add element (y/n):n
```

```
        Menu
1 - Add Element
2 - Delete Element
3 - Show HashTable
4 - Show HashTable using iterator
5 - Show Fullness Coefficient
6 - Show Size Of Table
0 - Exit
Choose:3
0:[zxc:zxc]
1:[qwe:qwe]
2:[asd:asd]

        Menu
1 - Add Element
2 - Delete Element
3 - Show HashTable
4 - Show HashTable using iterator
5 - Show Fullness Coefficient
6 - Show Size Of Table
0 - Exit
Choose:4
Key: zxc Value: zxc
Key: qwe Value: qwe
Key: asd Value: asd
```

```
        Menu
1 - Add Element
2 - Delete Element
3 - Show HashTable
4 - Show HashTable using iterator
5 - Show Fullness Coefficient
6 - Show Size Of Table
0 - Exit
Choose:2
Enter key:qwe
Done

        Menu
1 - Add Element
2 - Delete Element
3 - Show HashTable
4 - Show HashTable using iterator
5 - Show Fullness Coefficient
6 - Show Size Of Table
0 - Exit
Choose:3
0:[zxc:zxc]
1:
2:[asd:asd]
```

```
        Menu
1 - Add Element
2 - Delete Element
3 - Show HashTable
4 - Show HashTable using iterator
5 - Show Fullness Coefficient
6 - Show Size Of Table
0 - Exit
Choose:5
Fullness coefficient: 0.667

        Menu
1 - Add Element
2 - Delete Element
3 - Show HashTable
4 - Show HashTable using iterator
5 - Show Fullness Coefficient
6 - Show Size Of Table
0 - Exit
Choose:6
Size of HashTable: 3
```

Посилання на GitHub: https://github.com/YuriiPatriuk12/Labs.git