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

```cpp
#include <iostream>
#include "HashTable.cpp"

int main() {
    HashTable<string> hashTable(6);

    hashTable.add("key1", "first");
    hashTable.add("key10","second");
    hashTable.add("key3","third");
    hashTable.add("key4","fourth");
    hashTable.add("key5","fifth");
    hashTable.add("test","inserted");
    hashTable.add("key1231","seventh");
    hashTable.print();

    cout<<"-------------For loop-------------"<<endl;
    for(auto item = hashTable.begin(); item!=hashTable.end(); ++item)
    {
        cout<<"Key: "<<item->key<<" Value: "<<item->value<<endl;
    }

    HashTable<string>::Iterator item = hashTable.begin();
    cout<<endl<<"-------------While loop-------------"<<endl;
    while(!item.isEnd())
    {
        cout<<"Key: "<<item->key<<" Value: "<<item->value<<endl;
        ++item;
    }

    cout<<endl<<"Fullness coefficient: "<<hashTable.getFullnessCoefficient()<<endl;
    return 0;
}
```

Вміст файлу HashTable.h

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

template <typename T>
struct Node {
    T key;
    T value;
    Node* next;
    Node(const T& 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 T& key, const T& value);
```

```
    void del(const T& 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;
    float getFullnessCoefficient() const;

private:
    int hashFunction(const T& 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 T &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 T &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 T &key) const  {
    return hash<T>{}(key) % tableSize;
}

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

РЕЗУЛЬТАТ ТЕСТУВАННЯ

```
0:[key3:third]
1:[test:inserted]
2:[key4:fourth]->[key5:fifth]
3:[key10:second]
4:[key1:first]->[key1231:seventh]
5:

-------------For loop-------------
Key: key3 Value: third
Key: test Value: inserted
Key: key4 Value: fourth
Key: key5 Value: fifth
Key: key10 Value: second
Key: key1 Value: first
Key: key1231 Value: seventh

-------------While loop-------------
Key: key3 Value: third
Key: test Value: inserted
Key: key4 Value: fourth
Key: key5 Value: fifth
Key: key10 Value: second
Key: key1 Value: first
Key: key1231 Value: seventh

Fullness coefficient: 1.16667

Process finished with exit code 0
```

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