ДНІПРОВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ОЛЕСЯ ГОНЧАРА

Факультет прикладної математики

Кафедра обчислювальної математики та математичної кібернетики

Лабораторна робота №2

з дисципліни «**Об’єктно-орієнтоване програмування**»

Виконав

студент групи ПА-22-2

Початенко
Дмитро

Постанова задачі

Постановка задачі

Розробити об'єктно-орієнтовану бібліотеку для роботи зі структурами даних за однією з нижченаведених тем у відповідності з нижченаведеними вимогами. Властивості та методи для класів розробити у відповідності з відомими визначеннями відповідних структур даних. Скласти тести для перевірки працездатності бібліотеки. Скласти програму, що демонструє можливості розробленої бібліотеки.

Загальні вимоги

В незалежності від індивідуального варіанта повинні бути реалізовані наступні можливості:

1. Реалізація методів ініціалізації (конструктор за замовчуванням та конструктор з параметрами), копіювання (конструктор копіювання), індексації (перевантаження []), присвоювання (перевантаження =), візуалізації, збереження (на диск) та відновлення, діалогового керування, "розумного доступу" (перевантаження ->), а також псевдо змінних (забезпечення можливості виду: f(x)=const).
2. Перевантаження (спільне використання) потокового введення/виведення. (введення з файл, виведення в файл)
3. Створення та використання файла бібліотеки (*.lib, або *.obj).
4. Повторне використання класів без їх перекомпіляції ("ReUse", тобто наслідування).
5. Застосування вказаної структури даних для розв'язання типової задачі.

Результати виконання лабораторної роботи повинні бути викладені у вигляді звіту.

В ході демонстрації роботи, програма дозволяє користувачеві в діалоговому режимі (за допомогою меню) виконувати наступні операції:

1. Створювати об'єкти класу (у довільній кількості).
2. Вводити/виводити об'єкти на диск.
3. Візуалізувати об'єкти.
4. Виконувати операції над об'єктами.

Асоціативний масив з сортуванням. Забезпечити доступ к елементам за індексом та символьним представленням. Використання псевдозмінних для асоціативного масиву аналогічно використанню псевдозмінних для стека (див. варіант 1).

Хід роботи

Інтерфейс программи:

```
Menu:
1. Add an element
2. Remove an element
3. Display all elements
4. Save elements to file
5. Sort elements by value
6. Get element by key
7. Exit
Enter your choice:
```

Додавання елементу:

```
Menu:
1. Add an element
2. Remove an element
3. Display all elements
4. Save elements to file
5. Sort elements by value
6. Get element by key
7. Exit
Enter your choice: 1
Enter key (string): r
Enter value (integer): 24
```

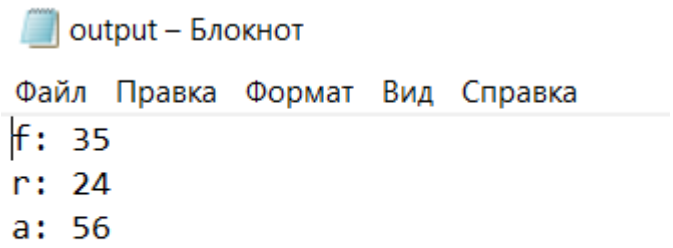
Візуалізація списку:

```
Menu:
1. Add an element
2. Remove an element
3. Display all elements
4. Save elements to file
5. Sort elements by value
6. Get element by key
7. Exit
Enter your choice: 3
Elements in the array:
f: 35
r: 24
p: 56
```

Збереження у файл:

```
Menu:
1. Add an element
2. Remove an element
3. Display all elements
4. Save elements to file
5. Sort elements by value
6. Get element by key
7. Exit
Enter your choice: 4
Elements saved to output.txt
```

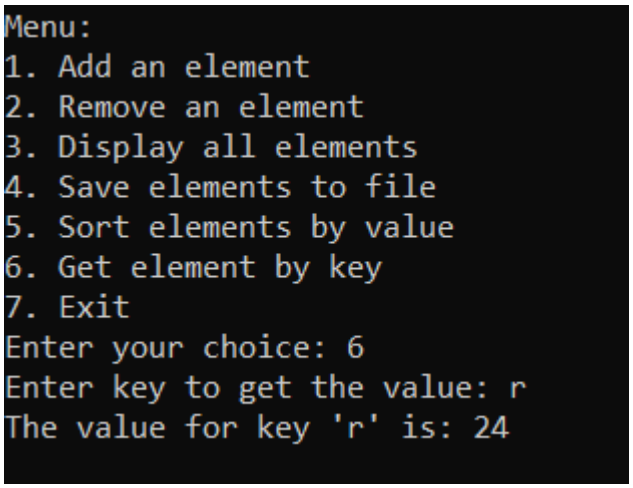
Збереженний файл:



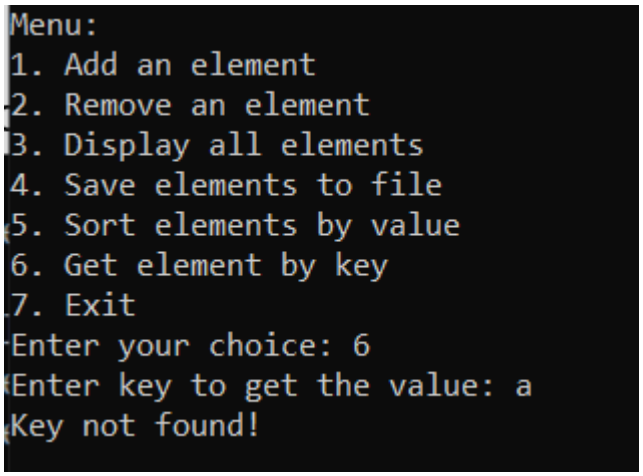
Сортування елементів:

```
Menu:
1. Add an element
2. Remove an element
3. Display all elements
4. Save elements to file
5. Sort elements by value
6. Get element by key
7. Exit
Enter your choice: 5
Elements sorted by value:
r: 24
f: 35
p: 56
```

Пошук за ключем:



Помилка пошуку:



Код программи

Додаток А

```
#include <iostream>
#include <map>
#include <string>
#include <algorithm>
#include <fstream>
#include <vector>

// Базовий клас для роботи з асоціативними масивами
class AssociativeArray {
protected:
	std::map<std::string, int> data;  // Використання std::map для асоціативного масиву

public:
	// Конструктор за замовчуванням
	AssociativeArray() {}

	// Деструктор
	virtual ~AssociativeArray() {}

	// Метод додавання елемента
	void addElement(const std::string& key, int value) {
	data[key] = value;
	}

	// Метод видалення елемента
	void removeElement(const std::string& key) {
	data.erase(key);
	}

	// Віртуальний метод для сортування
	virtual void sortElements() = 0;

	// Метод для виведення всіх елементів
	void displayElements() const {
	for (const auto& element : data) {
	std::cout << element.first << ": " << element.second << std::endl;
	}
	}

	// Метод для збереження елементів у файл
	void saveToFile(const std::string& filename) const {
	std::ofstream file(filename);
	if (file.is_open()) {
	for (const auto& element : data) {
	file << element.first << ": " << element.second << std::endl;
	}
	file.close();
	}
	else {
```

```
        std::cout << "Unable to open file" << std::endl;
        }
        }

        // Метод для отримання значення за ключем
        int getElement(const std::string& key) const {
        auto it = data.find(key);
        if (it != data.end()) {
        return it->second;
        }
        else {
        std::cerr << "Key not found!" << std::endl;
        return -1; // Повертаємо -1, якщо ключ не знайдено
        }
        }
};

// Похідний клас, що реалізує сортування елементів за значеннями
class SortedAssociativeArray : public AssociativeArray {
public:
        // Реалізація сортування за значеннями
        void sortElements() override {
        // Копіюємо дані у вектор для сортування
        std::vector<std::pair<std::string, int>> elements(data.begin(), data.end());

        // Сортування вектора за значеннями
        std::sort(elements.begin(), elements.end(), [](const auto& a, const auto& b) {
        return a.second < b.second; // Сортування за значенням
        });

        // Виведення відсортованих елементів
        std::cout << "Elements sorted by value:\n";
        for (const auto& element : elements) {
        std::cout << element.first << ": " << element.second << std::endl;
        }
        }
};

int main() {
        SortedAssociativeArray arr; // Створення об'єкта похідного класу
        int choice;
        std::string key;
        int value;

        do {
        // Виведення меню для користувача
        std::cout << "\nMenu:\n";
        std::cout << "1. Add an element\n";
        std::cout << "2. Remove an element\n";
        std::cout << "3. Display all elements\n";
        std::cout << "4. Save elements to file\n";
        std::cout << "5. Sort elements by value\n";
```

```
std::cout << "6. Get element by key\n";
std::cout << "7. Exit\n";
std::cout << "Enter your choice: ";
std::cin >> choice;

switch (choice) {
case 1:
std::cout << "Enter key (string): ";
std::cin >> key;
std::cout << "Enter value (integer): ";
std::cin >> value;
arr.addElement(key, value);
break;

case 2:
std::cout << "Enter key to remove: ";
std::cin >> key;
arr.removeElement(key);
break;

case 3:
std::cout << "Elements in the array:\n";
arr.displayElements();
break;

case 4:
arr.saveToFile("output.txt");
std::cout << "Elements saved to output.txt\n";
break;

case 5:
arr.sortElements();
break;

case 6:
std::cout << "Enter key to get the value: ";
std::cin >> key;
value = arr.getElement(key);
if (value != -1) {
std::cout << "The value for key '" << key << "' is: " << value << std::endl;
}
break;

case 7:
std::cout << "Exiting program...\n";
break;

default:
std::cout << "Invalid choice. Please try again.\n";
}

} while (choice != 7);
```

```
        return 0;
}
```