Автор: Татаренко А., КІТ-119а

Дата: 17 червня 2020

Лабораторна робота №10.

ШАБЛОННІ ФУНКЦІЇ

Тема. Шаблонні функції.

Мета – отримати базові знання про шаблонізацію (узагальнення) на основі шаблонних функцій.

1 Завдання до роботи

Індивідуальне завдання 19.

Створити клас, який не має полів, а всі необхідні дані передаються безпосередньо у функції. Клас має виконувати такі дії:
- виводити вміст масиву на екран;
- визначати індекс переданого елемента в заданому масиві;
- сортувати елементи масиву;
- визначати значення мінімального елемента масиву. При цьому необхідно продемонструвати роботу програми як з використанням стандартних типів даних, так і типів, створених користувачем.

2 Розробка алгоритму розв’язання задачі.

2.1 Опис змінних

Arr stud_array; class Student; class Arr;

Класи, методи, функції, конструктори

3 Код програми

Header.h

```
#pragma once
#define _CRT_SECURE_NO_WARNINGS
#define CRTDBG_MAP_ALLOC
#include <crtdbg.h>
#define DEBUG_NEW new(_NORMAL_BLOCK, FILE, __LINE)

#include <clocale>
#include <iostream>
#include <string>
#include <regex>
#include <iomanip>

using std::cin;
using std::cout;
using std::endl;
using std::string;
using std::regex;
using std::regex_search;
using std::ostream;
using std::istream;

using std::setw;
```

MyClass.h

```
#pragma once
#include "Header.h"

class Class
{
public:
	template<class T>
	void OutputArr(T, int) const;

	template<class T>
	void FindEl(T*, int, T) const;

	template<class T>
	T* Sort(T*, int, bool);

	template<class T>
	T FindMin(T*, int) const;

	template<class T>
	T EnterEl(T) const;

	template<class T>
	T ChoiseSort(T) const;

	template <typename T>
	static bool SortAsc(const T& a, const T& b) noexcept
	{
		return a > b;
	}

	template <typename T>
	static bool SortDesc(const T& a, const T& b) noexcept
	{
		return a < b;
```

```
	}

	~Class();
};

template<class T>
inline void Class::OutputArr(T array, int size) const
{
	for (size_t i = 0; i < size; i++)
	{
		cout << array[i] << "  ";
		cout << endl;
	}
	cout << endl;
}

template<class T>
inline T Class::EnterEl(T choise) const
{
	cout << endl << "Введите элемент, индекс которого хотите получить: ";
	cin >> choise;

	return choise;
}

template<class T>
inline T Class::ChoiseSort(T choise) const
{
	choise = 0;
	while (choise <= 0 || choise > 3)
	{
		cout << endl << "Сортировать по:" << endl;
		cout << "1) Убыванию\n2) Возрастанию\n3) Не сортировать\n";
		cout << "Ваш выбор: ";
		cin >> choise;
	}

	return choise;
}

template<class T>
inline void Class::FindEl(T* array, int size, T value) const
{
	bool FindEl = 0;
	for (size_t i = 0; i < size; i++)
	{
		if (array[i] == value)
		{
			cout << "Индекс нужного элемента: " << i << endl;
			FindEl = 1;
		}
	}
	if (FindEl == 0)
	{
		cout << "Нужного элемента в массиве нет." << endl;
	}

}

template<class T>
inline T* Class::Sort(T* array, int size, bool choiseSort)
{
	bool sort;
	T temp;
	bool pr;
```

```
	Class object;

	do
	{
		pr = 0;
		for (size_t i = 0; i < size - 1; i++)
		{
			if (choiseSort == 0)
			{
				sort = object.SortAsc(array[i], array[i + 1]);
			}
			else if (choiseSort == 1)
			{
				sort = object.SortDesc(array[i], array[i + 1]);
			}
			if (sort)
			{
				temp = array[i];
				array[i] = array[i + 1];
				array[i + 1] = temp;
				pr = 1;
			}
		}
	} while (pr);

	return array;
}

template<class T>
inline T Class::FindMin(T* array, int size) const
{
	T temp = array[0];
	for (size_t i = 1; i < size; i++)
	{
		if (array[i] < temp)
		{
			temp = array[i];
		}
	}
	//cout << endl << "Минимальный элемент: " << temp << endl << endl;

	return temp;
}

Class::~Class()
{
}
```

Student.h

```
#pragma once
#include "Header.h"

class Student
{
private:
	string name;
	int age;
	Student* array;

public:
	Student* createArray(int) noexcept;
	Student students(int) const noexcept;
```

```
	friend ostream& operator<<(ostream&, const Student) noexcept;
	friend istream& operator>>(istream&, Student&) noexcept;

	bool operator==(const Student) const noexcept;
	bool operator<(const Student) const noexcept;
	bool operator>(const Student) const noexcept;

	Student();
	Student(string, int);
	Student(const Student&);
	~Student();
};
```

main.cpp

```
#include "MyClass.h"
#include "Student.h"
#include "Header.h"

void Func();

int main()
{
	setlocale(LC_ALL, "ru");
	Func();

	if (_CrtDumpMemoryLeaks())
		cout << endl << "Обнаружена утечка памяти!" << endl;
	else
		cout << endl << "Утечки не обнаруженно!" << endl;

	system("PAUSE");
	return 0;
}

void Func()
{
	const int SIZE = 5;
	regex expresion("([\\d]+)");
	Class element;
	Student student1;
	Student* arrayStud = student1.createArray(SIZE);

	float elementToFind = 0;
	int command = 0;
	int choiseSort = 0;
	char charToFind = '*';

	int* arrayInt = new int[SIZE] { 7, 0, -7, 666, 999 };
	float* arrayFloat = new float[SIZE] {3.14, 89, 6.6, 3.33, 17};

	while (command != 4)
	{
		cout << "\nВыберете массив:\n1) типа int\n2) типа float\n";
		cout << "3) Свой тип данных\n4) Выйти\n";
		cout << "Массив: ";
		cin >> command;
		cout << endl;

		if (command == 1)
		{
			element.OutputArr(arrayInt, SIZE);
			elementToFind = element.EnterEl(elementToFind);
			element.FindEl(arrayInt, SIZE, (int)elementToFind);
```

```
                        choiseSort = element.ChoiseSort(choiseSort);
                        if (choiseSort == 1)
                        {
                                arrayInt = element.Sort(arrayInt, SIZE, true);
                                element.OutputArr(arrayInt, SIZE);
                        }
                        else if (choiseSort == 2)
                        {
                                arrayInt = element.Sort(arrayInt, SIZE, false);
                                element.OutputArr(arrayInt, SIZE);
                        }
                        cout << "Минимальный элемент: " << element.FindMin(arrayInt, SIZE);
                }
                else if (command == 2)
                {
                        element.OutputArr(arrayFloat, SIZE);
                        elementToFind = element.EnterEl(elementToFind);
                        element.FindEl(arrayFloat, SIZE, elementToFind);
                        choiseSort = element.ChoiseSort(choiseSort);
                        if (choiseSort == 1)
                        {
                                arrayFloat = element.Sort(arrayFloat, SIZE, true);
                                element.OutputArr(arrayFloat, SIZE);
                        }
                        else if (choiseSort == 2)
                        {
                                arrayFloat = element.Sort(arrayFloat, SIZE, false);
                                element.OutputArr(arrayFloat, SIZE);
                        }
                        cout << "Минимальный элемент: " << element.FindMin(arrayFloat,
SIZE);
                }
                else if (command == 3)
                {
                        element.OutputArr(arrayStud, SIZE);
                        elementToFind = element.EnterEl(elementToFind);
                        Student student1("Ivanov", elementToFind);
                        element.FindEl(arrayStud, SIZE, student1);
                        choiseSort = element.ChoiseSort(choiseSort);

                        if (choiseSort == 1)
                        {
                                arrayStud = element.Sort(arrayStud, SIZE, true);
                                element.OutputArr(arrayStud, SIZE);
                        }
                        else if (choiseSort == 2)
                        {
                                arrayStud = element.Sort(arrayStud, SIZE, false);
                                element.OutputArr(arrayStud, SIZE);
                        }
                        cout << "Минимальный элемент: " << element.FindMin(arrayStud, SIZE);
                }
                if (command >= 5)
                {
                        cout << "Неверная команда. Повторите попытку." << endl;
                }
        }


        delete[] arrayInt;
        delete[] arrayFloat;
        delete[] arrayStud;

        return;
}
```

Student.cpp

```
#include "Student.h"
#include "Header.h"

Student* Student::createArray(int size) noexcept
{
	array = new Student[size];

	for (size_t i = 0; i < size; i++)
	{
		array[i] = students(i);
	}

	return array;
}
Student Student::students(int value) const noexcept
{
	if (value == 0)
	{
		Student defaultStud;
		return defaultStud;
	}
	else if (value == 1)
	{
		Student stud("Peta", 20);
		return stud;
	}
	else if (value == 2)
	{
		Student stud("Ivan", 24);
		return stud;
	}
	else if (value == 3)
	{
		Student stud("Danil", 21);
		return stud;
	}
	else if (value == 4)
	{
		Student stud("Jhon", 26);
		return stud;
	}
}

ostream& operator<<(ostream& output, const Student stud) noexcept
{
	output.setf(std::ios::left);
	output << setw(12) << stud.name << setw(14) << stud.age;

	return output;
}
istream& operator>>(istream& input, Student& stud) noexcept
{
	input >> stud.age;

	return input;
}

bool Student::operator<(const Student stud) const noexcept
{
	return this->age < stud.age;
}
bool Student::operator>(const Student stud) const noexcept
```

```
{
	return this->age > stud.age;
}
bool Student::operator==(const Student stud) const noexcept
{
	return this->age == stud.age;
}

Student::Student() : name("Petrov"), age(18) {}
Student::Student(string name, int age) : name(name), age(age) {}
Student::Student(const Student& other) : name(other.name), age(other.age) {}

Student::~Student() {}
```

Test.cpp

```
#include "MyClass.h"
#include "Header.h"
#include "Student.h"

void Test_FindEl(Class, int*, int);
void Test_Sort(Class, int*, int);
void Test_FindMin(Class, int*, int);
void Func();

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

	Func();

	if (_CrtDumpMemoryLeaks())
		cout << endl << "Обнаружена утечка памяти!" << endl;
	else
		cout << endl << "Утечки не обнаруженно!" << endl;

	return 0;
}

void Func()
{
	Class element;
	int size = 10;
	int* array = new int[size] { 1, -5, 0, 22, 236, -523, 56423, -5634, -4235, 1000};

	Test_FindEl(element, array, size);
	Test_Sort(element, array, size);
	Test_FindMin(element, array, size);

	return;
}

void Test_FindEl(Class element, int* array, int size)
{
	int expected = 3;
	element.FindEl(array, size, 22);

	//if (expected == real) cout << "Тест нахождения элементов \t выполнен
успешно.\n";
	//else cout << "Тест нахождения элементов \t не выполнен успешно.\n";
}
void Test_Sort(Class element, int* array, int size)
{
	int beforeSort = array[0];
	array = element.Sort(array, size, 1);
```

```
	int afterSort = array[0];

	if (beforeSort != afterSort && afterSort == 56423) cout << "Тест
сортировки\t\t\t\t выполнен успешно.\n";
	else cout << "Тест сортировки\t\t\t\t не выполнен успешно.\n";
}
void Test_FindMin(Class element, int* array, int size)
{
	int temp = element.FindMin(array, size);

	if (temp == -5634) cout << "Тест нахождения минимального элемента\t выполнен
успешно.\n";
	else cout << "Тест нахождения минимального элемента\t не выполнен успешно.\n";
}
```

4 Результати тестування

```
Выберете массив:
1) типа int
2) типа float
3) Свой тип данных
4) Выйти
Массив: 1

7
0
-7
666
999


Введите элемент, индекс которого хотите получить: 2
Нужного элемента в массиве нет.

Сортировать по:
1) Убыванию
2) Возрастанию
3) Не сортировать
Ваш выбор: 2
-7
0
7
666
999

Минимальный элемент: -7
Выберете массив:
1) типа int
2) типа float
3) Свой тип данных
4) Выйти
Массив:
```

5 Опис результатів

При виконані лабораторної роботи було набуто практичні навички роботи з шаблонізацію (узагальненням) на основі шаблонних функцій. Було створенно меню за варіантами вибору типу масиву. Реалізовані методи роботи з масивом.