Звіт

Автор: Рябов О.В. КІТ-119а

Дата: 29 квітня 2020

Лабораторна робота 10. ШАБЛОННІ ФУНКЦІЇ

Тема. Шаблонні функції.
Мета: отримати базові знання про шаблонізацію (узагальнення) на основі шаблонних функцій.

1. Завдання до роботи

Загальне завдання. 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.

2.1. Опис класів

Клас, що містить шаблонні функції: MyClass.
Клас, як приклад даних, створений користувачем: Student.

2.2. Опис змінних

`string name` – поле імені студента (змінна класу Student).
`int age` – вік студента (змінна класу Student).
`Student* array` – масив елементів класу Student (змінна класу Student).

2.3. Опис методів

`Student* createArray(int) noexcept` – створення масиву даних (метод класу Student).
`friend ostream& operator<<(ostream&, const Student) noexcept` – перевантаження оператора << (метод класу Student).
`friend istream& operator>>(istream&, Student&) noexcept` – перевантаження оператора >> (метод класу Student).
`bool operator==(const Student) const noexcept` – перевантаження оператора == (метод класу Student).
`bool operator<(const Student) const noexcept` – перевантаження оператора < (метод класу Student).
`bool operator>(const Student) const noexcept` – перевантаження оператора > (метод класу Student).

`Student()` – конструктор за замовчуванням (метод класу `Student`).
`Student(string, int)` – конструктор з параметрами (метод класу `Student`).
`Student(const Student&)` – конструктор копіювання (метод класу `Student`).
`~Student()` – деструктор класу (метод класу `Student`).
`~MyClass()` – деструктор класу (метод класу `MyClass`).

`template<class T> void OutputArr(T, int) const` – шаблонна функція виведення масиву у консоль (метод класу `MyClass`).
`template<class T> void FindEl(T*, int, T) const` – шаблонна функція знаходження елемента у масиві (метод класу `MyClass`).
`template<class T> T* Sort(T*, int, bool)` – шаблонна функція сортування масиву (метод класу `MyClass`).
`template<class T> T FindMin(T*, int) const` – шаблонна функція знаходження мінімального елементу у масиві (метод класу `MyClass`).
`template<class T> T EnterEl(T) const` – шаблонна функція введення елемента (метод класу `MyClass`).
`template<class T> T ChoiseSort(T) const` – шаблонна функція вибору сортування масива (метод класу `MyClass`).

2.4. Опис функцій

`void Test_FindEl(MyClass, int*, int)` – функція тестування знаходження індекса елемента у масиві.
`void Test_Sort(MyClass, int*, int)` – функція сортування масива даних.
`void Test_FindMin(MyClass, int*, int)` – функція знаходження мінімального елемента у масиві.

3. Текст програми

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();
};
```

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;
```

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("Ivan", 20);
		return stud;
	}
	else if (value == 2)
	{
		Student stud("John", 24);
		return stud;
	}
	else if (value == 3)
	{
		Student stud("Roman", 21);
		return stud;
	}
	else if (value == 4)
	{
		Student stud("Susan", 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() {}
```

MyClass.h

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

class MyClass
{
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;
	}

	~MyClass();
};

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

	return choise;
}

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



MyClass::~MyClass() {}
```

Main.cpp

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

void Func();

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

	if (_CrtDumpMemoryLeaks()) cout << endl << "Есть утечка памяти." << endl;
	else cout << endl << "Утечка памяти отсутствует." << endl;

	return 0;
}

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

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

	int* arrayInt = new int[SIZE] { 5, 0, -1, 255, 100 };
	float* arrayFloat = new float[SIZE] {0.55, 100, 36.66, 14.69, 5};

	while (command != 4)
	{
		cout << "\nС каким массивом мы работаем?\n1) int\n2) float\n";
		cout << "3) Собственный тип данных\n4) Выйти из программы.\n";
		cout << "Ваш выбор: ";
		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)
			{
```

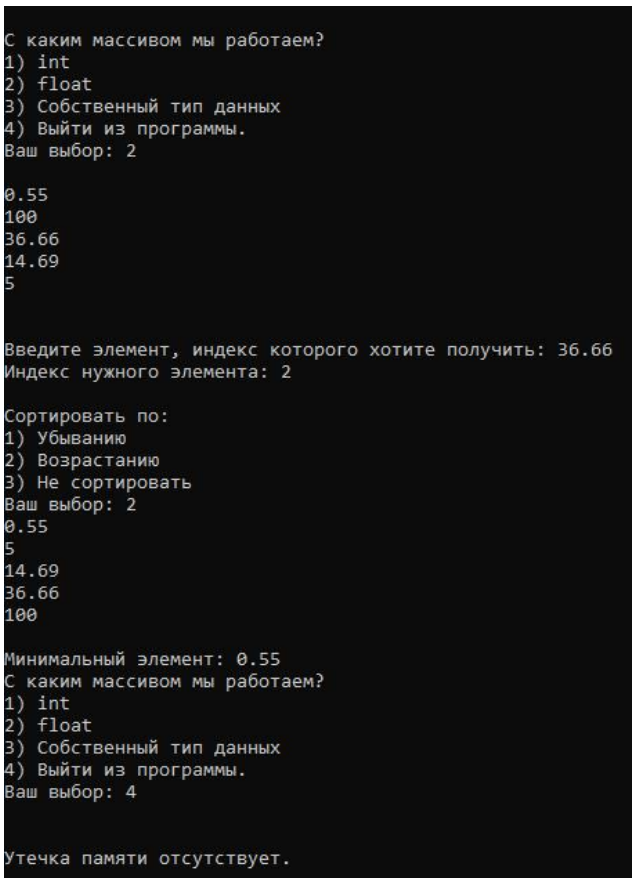
```
                        arrayFloat = element.Sort(arrayFloat, SIZE, true);
                        element.OutputArr(arrayFloat, SIZE);
                }
                else if (choiseSort == 2)
                {
                        arrayFloat = element.Sort(arrayFloat, SIZE, false);
                        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;
}
```

Test.cpp

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

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

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

    Func();

    if (_CrtDumpMemoryLeaks()) cout << endl << "Есть утечка памяти." << endl;
    else cout << endl << "Утечка памяти отсутствует." << endl;

    return 0;
}
```

```
void Func()
{
	MyClass 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(MyClass element,int* array, int size)
{
	int expected = 3;
	element.FindEl(array, size, 22);
}
void Test_Sort(MyClass 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(MyClass 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 роботи програми



5. Висновки

При виконанні даної лабораторної роботи було набуто практичного досвіду роботи з шаблонними функціями.

Програма протестована, витоків пам’яті немає, виконується без помилок.