Звіт

Автор: Рябов О.В. КІТ-119а

Дата: 10 травня 2020

Лабораторна робота 11. ШАБЛОННІ КЛАСИ

Тема: шаблонні класи.
Мета: поширити знання у шаблонізації (узагальненні) на основі вивчення шаблонних класів та створення власних шаблонних типів.

1. Завдання до роботи

Загальне завдання. Модернізувати клас, що був розроблений у попередній роботи таким шляхом:

- зробити його шаблонним;
- додати поле – шаблонний масив;
- видалити з аргументів існуючих методів масив, а замість цього використовувати масив-поле класу.

Необхідно продемонструвати роботу програми як з використанням стандартних типів даних, так і типів, які створені користувачем.

Додаткове завдання на оцінку «відмінно»:

- продемонструвати роботу шаблонного класу, в масиві якого знаходиться ієрархія класів (тобто не тільки базовий клас, а ще й клас-спадкоємець).

2.1. Опис класів

Шаблонний клас: `TemplateClass`.
Клас, створений користувачем: `Student`.
Клас-спадкоємець: `Foreigner`.

2.2. Опис змінних

`string country` – країна іноземця (змінна класу `Foreigner`).
`string name` – ім’я студента (змінна класу `Student`).
`int age` – вік студента (змінна класу `Student`).
`T** array` – масив шаблоних даних (змінна класу `TemplateClass`).

2.3. Опис методів

`void OutputArr(int size) const` – виведення даних у консоль (метод класу `TemplateClass`).
`int FindEl(T element, int size) const` – знаходження елемента у масиві даних (метод класу `TemplateClass`).
`T* Sort(int size, bool choiseSort)` – сортування даних (метод класу `TemplateClass`).
`T FindMin(int size) const` – знаходження мінімального елементу в масиві (метод класу `TemplateClass`).

`TemplateClass(T** array) :array(array)` – конструктор класу з параметрами (метод класу `TemplateClass`).
`~TemplateClass()` – деструктор класу (метод класу `TemplateClass`).

`int LinesInFile(string) const` – знаходження кількості рядків у файлі (метод класу ArrayList).
`void OutputArr() const` – виведення даних у консоль (метод класу ArrayList).
`void Task()` – виконання індивідуального завдання (метод класу ArrayList).
`void SaveToFile(string) const` – виведення даних у файл (метод класу ArrayList).
`void DeleteArr()`– видалення масивів даних (метод класу ArrayList).

`~ArrayList()` – деструктор списку елементів (метод класу ArrayList).
`~Array()` – деструктор масиву (метод класу Array).
`Array()` – конструктор масиву (метод класу Array).

3. Текст програми

Foreigner.cpp

```cpp
#include "Foreigner.h"

string Foreigner::getInfo() const
{
	stringstream temp;
	temp.setf(std::ios::left);
	temp << setw(11) << name << setw(10) << age << setw(6) << country;

	return temp.str();
}

Foreigner::Foreigner():Student(), country("Зимбабве") {}
Foreigner::Foreigner(string country, string name, int age): country(country), Student(name, age){}
Foreigner::Foreigner(const Foreigner& other): Student(other), country(other.country){}
Foreigner::~Foreigner() {}
```

Main.cpp

```cpp
#include "TemplateClass.h"
#include "Student.h"
#include "Header.h"
#include "Foreigner.h"

void ArrayInt(int);
void ArrayFloat(int);
void ArrayClass(int);
Student** CreateArray(int);

int main()
{
	setlocale(LC_ALL, "Rus");
	const int SIZE = 5;
	int choise = 0;
```

```
	while (choise != 4)
	{
		cout << "С массивом какого типа данных работать?" << endl;
		cout << "1. int\n2. float\n3. Пользовательский тип данных\n4. Завершить работу\n";
		cout << "==================\nВаш выбор: ";
		cin >> choise;

		if (choise == 1)
			ArrayInt(SIZE);
		else if (choise == 2)
			ArrayFloat(SIZE);
		else if (choise == 3)
			ArrayClass(SIZE);
		else if (choise > 4 || choise < 1)
			cout << endl << "Неверный номер. Повторите попытку." << endl;
		else
			choise = 4;
	}

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

	return 0;
}
void ArrayInt(int SIZE)
{
	int* arraySize = new int[SIZE];
	int** arrayInt = new int* [SIZE];
	int choise = 0, result = 0;

	for (size_t i = 0; i < SIZE; i++)
	{
		arraySize[i] = rand();
		arrayInt[i] = &arraySize[i];
	}

	TemplateClass<int> element(arrayInt);
	cout << endl << "Массив элементов типа int:" << endl;
	element.OutputArr(SIZE);

	choise = element.EnterEl(choise);
	result = element.FindEl(choise, SIZE);
	if (result == -1)
		cout << "Элемент с индексом " << choise << " отсутствует." << endl;
	else
		cout << "Индекс элемента " << choise << " : " << result << endl;

	choise = element.ChoiseSort(choise);
	element.Sort(SIZE, choise - 1);
	cout << endl << "Результат сортировки:" << endl;
	element.OutputArr(SIZE);

	result = element.FindMin(SIZE);
	cout << "Минимальный элемент в массиве: " << result << endl;

	delete[] arraySize;
	delete[] arrayInt;
}
void ArrayFloat(int SIZE)
{
	float* arraySize = new float[SIZE];
	float** arrayFloat = new float* [SIZE];
	float choise = 0, result = 0;

	for (size_t i = 0; i < SIZE; i++)
	{
		arraySize[i] = (rand() % 101 - 50) / 10.0;
		arrayFloat[i] = &arraySize[i];
	}
```

```
	TemplateClass<float> element(arrayFloat);
	cout << endl << "Массив элементов типа float:" << endl;
	element.OutputArr(SIZE);

	choise = element.EnterEl(choise);
	result = element.FindEl(choise, SIZE);
	if (result == -1)
		cout << "Элемент с индексом " << choise << " отсутствует." << endl;
	else
		cout << "Индекс элемента " << choise << " : " << result << endl;

	choise = element.ChoiseSort(choise);
	element.Sort(SIZE, choise - 1);
	cout << endl << "Результат сортировки:" << endl;
	element.OutputArr(SIZE);

	result = element.FindMin(SIZE);
	cout << "Минимальный элемент в массиве: " << result << endl;

	delete[] arraySize;
	delete[] arrayFloat;
}
void ArrayClass(int SIZE)
{
	Student guy;
	Student** ArrayClass = CreateArray(SIZE);
	TemplateClass<Student> element(ArrayClass);
	int choise = 0, result = 0;

	cout << setw(6) << "Имя" << setw(10) << "Возраст" << setw(10) << "Страна" << endl;
	element.OutputArr(SIZE);

	cout << "Введите элемент, индекс которого хотите получить: ";
	cin >> guy;

	result = element.FindEl(guy, SIZE);
	if (result == -1)
		cout << "Элемент с индексом " << guy << " отсутствует." << endl;
	else
		cout << "Индекс элемента: " << result << endl;

	while (choise <= 0 || choise > 3)
	{
		cout << endl << "Сортировать по:" << endl;
		cout << "1) Возрастанию\n2) Убыванию\n3) Не сортировать\n";
		cout << "Ваш выбор: ";
		cin >> choise;

		if (choise <= 0 || choise > 3)
			cout << "Ошбика. Неверная команда. Повторите попытку." << endl;
	}

	if (choise == 1) choise = 1;
	else if (choise == 2) choise = 0;

	element.Sort(SIZE, choise);
	cout << endl << "Результат сортировки:" << endl;
	element.OutputArr(SIZE);

	guy = element.FindMin(SIZE);
	cout << "Минимальный элемент в массиве: " << guy << endl;

	for (size_t i = 0; i < SIZE; i++)
	{
		delete ArrayClass[i];
	}
	delete[] ArrayClass;
}
```

```
Student** CreateArray(int size)
{
	Student** array = new Student * [size];

	for (size_t i = 0; i < size; i++)
	{
		if (i == 0)
		{
			*(array + i) = new Student();
		}
		else if (i == 1)
		{
			*(array + i) = new Foreigner("США", "Джим", 22);
		}
		else if (i == 2)
		{
			*(array + i) = new Student("Алексей", 17);
		}
		else if (i == 3)
		{
			*(array + i) = new Foreigner("Португалия", "Георг", 20);
		}
		else
		{
			*(array + i) = new Student("Павел", 28);
		}
	}

	return array;
}
```

Student.cpp

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

string Student::getInfo() const
{
	stringstream temp;
	temp.setf(std::ios::left);
	temp << setw(11) << name << setw(6) << age;

	return temp.str();
}

ostream& operator<<(ostream& output, const Student& stud) noexcept
{
	output.setf(std::ios::left);
	output << stud.getInfo();

	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;
}
```

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

Student::Student(): name("Петров"), 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

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

void Test_FindEl(TemplateClass<int>&, int);
void Test_Sort(TemplateClass<int>&, int);
void Test_FindMin(TemplateClass<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()
{
	const int size = 5;
	int values[size] = { 1, -5, 0, 22, 236};
	int** array = new int* [size];
	for (size_t i = 0; i < size; i++)
	{
		array[i] = &values[i];
	}
	TemplateClass<int> element(array);

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

	delete[] array;

	return;
}

void Test_FindEl(TemplateClass<int>& element, int size)
{
	if (element.FindEl(22, size) == 3)
		cout << "Тест нахождения элементов\t\tвыполнен успешно.\n";
	else
		cout << "Тест нахождения элементов\t\tне выполнен успешно.\n";
}
void Test_Sort(TemplateClass<int>& element, int size)
{
	if (element.FindEl(-5, size) == 1)
		cout << "Тест сортировки\t\t\t\tвыполнен успешно.\n";
	else
		cout << "Тест сортировки\t\t\t\tне выполнен успешно.\n";
}
void Test_FindMin(TemplateClass<int>& element, int size)
```

```
{
	if (element.FindMin(size) == -5)
		cout << "Тест нахождения минимального элемента\tвыполнен успешно.\n";
	else
		cout << "Тест нахождения минимального элемента\tне выполнен успешно.\n";
}
```

Foreigner.h

```
#pragma once
#include "Student.h"
class Foreigner final: public Student
{
private:
	string country;

public:
	string getInfo() const override final;
	friend ostream& operator<<(ostream&, const Foreigner) noexcept;

	Foreigner();
	Foreigner(string, string, int);
	Foreigner(const Foreigner&);
	~Foreigner() override final;
};
```

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>
#include <sstream>

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;
using std::stringstream;
```

Student.h

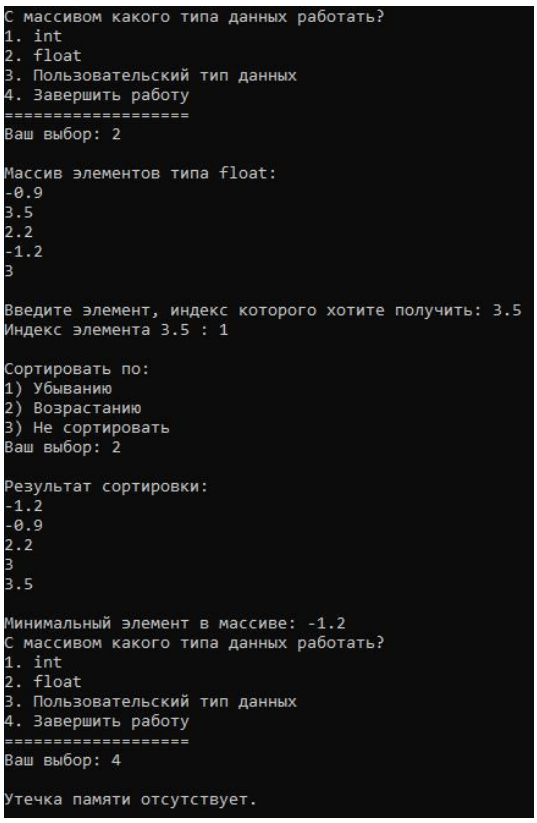
```
#pragma once
#include "Header.h"

class Student
{
protected:
	string name;
	int age;

public:
	virtual string getInfo() const;

	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&);
	virtual ~Student();
};
```

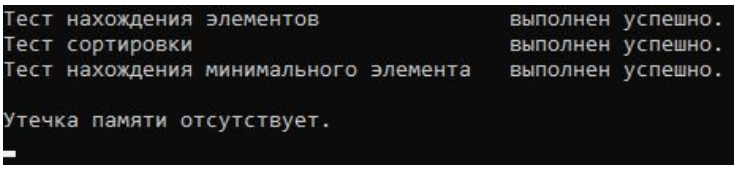
TemplateClass.h

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

template <typename T>
class TemplateClass
{
private:
	T** array;

public:
	void OutputArr(int size) const
	{
		for (size_t i = 0; i < size; i++)
			cout << *array[i] << endl;
		cout << endl;
	}

	int FindEl(T element, int size) const
	{
		for (size_t i = 0; i < size; i++)
			if (*array[i] == element)
				return i;
		return -1;
	}

	T* Sort(int size, bool choiseSort)
	{
		TemplateClass<T> object(T);
		bool sort = 0, pr = 0;
		T* temp = 0;

		do
		{
			pr = 0;
			for (size_t i = 0; i < size - 1; i++)
			{
				if (choiseSort == 0)
					sort = *array[i] < *array[i + 1];
				else if (choiseSort == 1)
					sort = *array[i] > * array[i + 1];
```

```
                        if (sort)
                        {
                                temp = *(array + i);
                                *(array + i) = *(array + i + 1);
                                *(array + i + 1) = temp;
                                pr = 1;
                        }
                }
        } while (pr);

        return temp;
}

T FindMin(int size) const
{
        T temp = *array[0];
        for (size_t i = 1; i < size; i++)
                if (*array[i] < temp)
                        temp = *array[i];
        return temp;
}

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

        return choise;
}

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

                if (choise <= 0 || choise > 3)
                        cout << "Ошибка. Неверная команда. Повторите попытку." << endl;
        }

        return choise;
}

TemplateClass(T** array) :array(array) {}
~TemplateClass() {};
};
```

4. Результати роботи про грами





5. Висновки

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

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