Автор: Татаренко А., КІТ-119а

Дата: 17 червня 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

Індивідуальне завдання 19.

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.

2 Розробка алгоритму розв’язання задачі.

2.1 Опис змінних

Arr stud_array; class Student; class Arr;

Класи, методи, функції, конструктори

3 Код програми

Foreign.h

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

class Foreign final : public Student
{
private:
	string country;

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

	Foreign();
	Foreign(string, string, int);
	Foreign(const Foreign&);
	~Foreign() 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;
```

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
{
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();
};
```

TClass.h

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

template <typename T>
class TClass
{
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)
	{
```

```
		TClass<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 << "Неверная команда. Повторите попытку." << endl;
		}

		return choise;
	}

	TClass(T** array) :array(array) {}
	~TClass() {};

};
```

Foreign.cpp

```
#include "Foreign.h"

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

	return temp.str();
}

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

	return output;
}

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

main.cpp

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

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

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

	while (choise != 4)
	{
		cout << "\nВыберете массив:\n1) типа int\n2) типа float\n";
		cout << "3) Свой тип данных\n4) Выйти\n";
		cout << "Массив: ";
		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 << "Неверная команда. Повторите попытку." << endl;
		else
			choise = 4;
	}
```

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

	system("PAUSE");
	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];
	}

	TClass<int> element(arrayInt);
	cout << endl << "Массив 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];
	}

	TClass<float> element(arrayFloat);
	cout << endl << "Массив 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);
	TClass<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) Убыванию\n2) Возрастанию\n3) Не сортировать\n";
		cout << "Ваш выбор: ";
		cin >> choise;

		if (choise <= 0 || choise > 3)
			cout << "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 Foreign("США", "Джим", 22);
		}
		else if (i == 2)
		{
			*(array + i) = new Student("Алексей", 17);
		}
		else if (i == 3)
		{
			*(array + i) = new Foreign("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;
}

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

```
#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

Массив типа int:
41
18467
6334
26500
19169

Введите элемент, индекс которого хотите получить: 41
Индекс элемента 41 : 0

Сортировать по:
1) Убыванию
2) Возрастанию
3) Не сортировать
Ваш выбор: 1

Результат сортировки:
26500
19169
18467
6334
41

Минимальный элемент в массиве: 41

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 шаблонізації (узагальненні) на основі вивчення шаблонних класів та створення власних шаблонних типів. Було створенно меню за варіантами вибору типу масиву. Реалізовані методи роботи з масивом.