Звіт

Автор: Капелька Я.І. КІТ-119а Дата: 29 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 **Індивідуальне завдання**:

Зробити клас з попередньої лабораторної роботи шаблонним.

2. Опис класів, змінних, методів та функцій

2.1 Опис класів

`template <typename t> class Function_class` – шаблонний клас призначений для сортування масивів даних різних типів.

`class drob` – клас призначений для демонстрування роботи іншого класа типами даних створених користувачем.

`class drob2 : public drob` – клас нащадок класу drob.

2.2 Опис змінних

`t** data` – поле класу funcclass масив даних довільного типу.
`int len` – поле класу funcclass довжина масиву.
`int q` – змінна що визначає з даними якого типу буде працювати програма. Отримується від користувача.
`int n` – змінна що визначає розмір масив. Отримується від користувача.
`int t` – змінна що використовується для реалізації вибору в сторонніх операціях.
`Function_class f` – змінна, що використовується для визова функцій, які є методами класу.
`int* data` – масив типу int.
`float* data` – масив типу float.
`drob* data` – масив типу drob.
`float g` – змінна що використовується для отримання даних типу float від користувача.
`drob m` – змінна що використовується для отримання даних типу drob від користувача.

2.3 Опис методів

`template <typename t> void print_to_Screen(t* data, int l)` – метод класу funcclass, виводить на екран зміст даного масиву.
`template <typename t> int find_index(t* data, int l, t el)` – метод класу funcclass, знаходить індекс даного елемента у даному масиві.
`template <typename t> void Sort(t* data, int l, int s)` – метод класу funcclass, сортує масив у порядку спадання або зростання.
`template <typename t> t find_min(t* data, int l)` – метод класу funcclass, знаходить найменший елемент масиву.
`template <typename t> t find_max(t* data, int l)` – метод класу funcclass, знаходить найбільший елемент масиву.
`drob& operator=(const drob& d)` – метод класу drob, перевантаження оператору присваювання.
`drob(int f, int s)` – конструктор з параметрами класу drob.
`drob()` – конструктор класу drob.
`drob2(int f, int s, char c)` – конструктор з параметрами класу drob2.
`drob2()` – конструктор класу drob2.

2.4 Опис функцій

`std::ostream& operator<<(std::ostream& os, const drob& d)` – перевантаження оператору виведення для класу drob.
`std::istream& operator>>(std::istream& is, drob& d)` – перевантаження оператору введення для класу drob.
`bool operator==(const drob& d1, const drob& d2)` – перевантаження оператору порівняння для класу drob.
`bool operator!=(const drob& d1, const drob& d2)` – перевантаження іншого оператору порівняння для класу drob.
`bool operator>(const drob& d1, const drob& d2)` – перевантаження іншого оператору порівняння для класу drob.
`bool operator<(const drob& d1, const drob& d2)` – перевантаження іншого оператору порівняння для класу drob.
`std::ostream& operator<<(std::ostream& os, const drob2& d)` – перевантаження оператору виведення для класу drob2.
`std::istream& operator>>(std::istream& is, drob2& d)` – перевантаження оператору введення для класу drob2.
`bool operator==(const drob2& d1, const drob2& d2)` – перевантаження оператору порівняння для класу drob2.
`bool operator!=(const drob2& d1, const drob2& d2)` – перевантаження іншого оператору порівняння для класу drob2.
`bool operator>(const drob2& d1, const drob2& d2)` – перевантаження іншого оператору порівняння для класу drob2.
`bool operator<(const drob2& d1, const drob2& d2)` – перевантаження іншого оператору порівняння для класу drob2.

`void menu()` – функція меню.

3 Текст програми

Лабораторная работа 11.cpp

```
#define _CRTDBG_MAP_ALLOC
#include "Function_class.h"
int main()
{
	setlocale(LC_ALL, "Russian");
	menu();
	if (_CrtDumpMemoryLeaks())
	{
		std::cout << "Утечка памяти обнаружена." << "\n";
	}
	else
	{
		std::cout << "Утечка памяти не обнаружена." << "\n";
	}
}
```

Function_class.h

```
#pragma once
#include <iostream>
template <typename t> class Function_class
{
public:
t** data;
int len;
void print_to_Screen();
int find_index(t element);
void Sort(int s);
t find_min();
t find_max();
};
class drob
{
public:
int chislitel;
int znaminaytel;
drob& operator= (const drob& d);
drob(int f, int s);
drob();
virtual char get_char() const;
};
class drob2 : public drob
{
public:
char c;
drob2& operator= (const drob2& d);
drob2(int f, int s, char c);
drob2();
virtual char get_char() const override;
};
std::ostream& operator<< (std::ostream& os, const drob& d);
std::istream& operator>> (std::istream& is, drob& d);
bool operator== (const drob& d1, const drob& d2);
bool operator!= (const drob& d1, const drob& d2);
bool operator> (const drob& d1, const drob& d2);
bool operator<(const drob& d1, const drob& d2);
std::ostream& operator<< (std::ostream& os, const drob2& d);
std::istream& operator>> (std::istream& is, drob2& d);
bool operator== (const drob2& d1, const drob2& d2);
bool operator!= (const drob2& d1, const drob2& d2);
bool operator>(const drob2& d1, const drob2& d2);
bool operator<(const drob2& d1, const drob2& d2);
void menu();
```

Function_class.cpp

```cpp
#include "Function_class.h"
template <typename t> void Function_class<t>::print_to_Screen()
{
	std::cout << "Вот ваш массив данных: " << "\n";
	for (int i = 0; i < this->len; i++)
	{
		std::cout << i + 1 << " " << *(data[i]) << "\n";
	}
}
template <typename t> int Function_class<t>::find_index(t element)
{
	for (int i = 0; i < this->len; i++)
	{
		if (*(data[i]) == element)
		{
			return i;
		}
	}
}
template <typename t> void Function_class<t>::Sort(int s)
{
	bool check = false;
	if (s >= 0)
	{
		do
		{
			check = false;
			for (int i = 0; i < this->len - 1; i++)
			{
				if (*(data[i]) < *(data[i + 1]))
				{
					std::swap(*(data[i]), *(data[i + 1]));
					check = true;
				}
			}
		} while (check);
	}
	else
	{
		do
		{
			check = false;
			for (int i = 0; i < this->len - 1; i++)
			{
				if (*(data[i]) > * (data[i + 1]))
				{
					std::swap(*(data[i]), *(data[i + 1]));
					check = true;
				}
			}
		} while (check);
	}
}
template <typename t> t Function_class<t>::find_min()
{
	t min = *(data[0]);
	for (int i = 0; i < this->len; i++) {
		if (min > * (data[i])) {
			min = *(data[i]);
		}
	}
	return min;
}
template <typename t> t Function_class<t>::find_max()
{
	t max = *(data[0]);
	for (int i = 0; i < this->len; i++)
	{
```

```
			if (max < *(data[i]))
			{
				max = *(data[i]);
			}
		}
		return max;
}
std::ostream& operator<<(std::ostream& os, const drob& d)
{
		return os << d.chislitel << "/" << d.znaminaytel << " " << d.get_char();
}
std::istream& operator>>(std::istream& is, drob& d)
{
		return is >> d.chislitel >> d.znaminaytel;
}
bool operator==(const drob& d1, const drob& d2)
{
		float dr1, dr2;
		dr1 = float(d1.chislitel) / float(d1.znaminaytel);
		dr2 = float(d2.chislitel) / float(d2.znaminaytel);
		if (dr1 == dr2)
		{
			return true;
		}
		else
		{
			return false;
		}
}
bool operator!=(const drob& d1, const drob& d2)
{
		return !(d1 == d2);
}
bool operator>(const drob& d1, const drob& d2)
{
		float dr1, dr2;
		dr1 = float(d1.chislitel) / float(d1.znaminaytel);
		dr2 = float(d2.chislitel) / float(d2.znaminaytel);
		if (dr1 > dr2) {
			return true;
		}
		else {
			return false;
		}
}
bool operator<(const drob& d1, const drob& d2)
{
		float dr1, dr2;
		dr1 = float(d1.chislitel) / float(d1.znaminaytel);
		dr2 = float(d2.chislitel) / float(d2.znaminaytel);
		if (dr1 < dr2)
		{
			return true;
		}
		else
		{
			return false;
		}
}
drob& drob::operator=(const drob& d)
{
		chislitel = d.chislitel;
		znaminaytel = d.znaminaytel;
		return *this;
}
drob::drob(int f, int s)
{
		chislitel = f;
		znaminaytel = s;
```

```
}
drob::drob() {
	chislitel = 1;
	znaminaytel = 2;
}
drob2& drob2::operator=(const drob2& d)
{
	chislitel = d.chislitel;
	znaminaytel = d.znaminaytel;
	c = d.c;
	return *this;
}
drob2::drob2(int f, int s, char c) : drob(f, s), c(c) {}
drob2::drob2() : drob(), c('c') {}
std::ostream& operator<<(std::ostream& os, const drob2& d)
{
	return os << d.chislitel << "/" << d.znaminaytel << " " << d.get_char();
}
std::istream& operator>>(std::istream& is, drob2& d)
{
	return is >> d.chislitel >> d.znaminaytel >> d.c;
}
bool operator==(const drob2& d1, const drob2& d2)
{
	float dr1, dr2;
	dr1 = float(d1.chislitel) / float(d1.znaminaytel);
	dr2 = float(d2.chislitel) / float(d2.znaminaytel);
	if (dr1 == dr2)
	{
		return true;
	}
	else
	{
		return false;
	}
}
bool operator!=(const drob2& d1, const drob2& d2)
{
	return !(d1 == d2);
}
bool operator>(const drob2& d1, const drob2& d2)
{
	float dr1, dr2;
	dr1 = float(d1.chislitel) / float(d1.znaminaytel);
	dr2 = float(d2.chislitel) / float(d2.znaminaytel);
	if (dr1 > dr2)
	{
		return true;
	}
	else
	{
		return false;
	}
}
bool operator<(const drob2& d1, const drob2& d2)
{
	float dr1, dr2;
	dr1 = float(d1.chislitel) / float(d1.znaminaytel);
	dr2 = float(d2.chislitel) / float(d2.znaminaytel);
	if (dr1 < dr2)
	{
		return true;
	}
	else
	{
		return false;
	}
}
char drob::get_char() const
```

```
{
	return 'd';
}
char drob2::get_char() const
{
	return this->c;
}
void menu() {
	std::cout << "Выберите тип данных который будет использованн в массиве: " << "\n";
	std::cout << "1 - int " << "\n";
	std::cout << "2 - дробное число (кастомный тип данных) " << "\n";
	std::cout << "Введите цисло соответствующее нужному типу данных: ";
	int q, n, t;
	std::cin >> q;
	if (q == 1)
	{
		Function_class<int> d;
		std::cout << "Введите размер массива: ";
		std::cin >> n;
		d.data = new int* [n];
		d.len = n;
		for (int i = 0; i < n; i++)
		{
			d.data[i] = new int(i + 1);
		}
		d.print_to_Screen();
		while (true)
		{
			std::cout << "Выберите нужное действие: " << "\n";
			std::cout << "1 - отсортировать массив" << "\n";
			std::cout << "2 - найти минимум или максимум" << "\n";
			std::cout << "3 - получить индекс элемента по его значению" << "\n";
			std::cout << "4 - показать массив" << "\n";
			std::cout << "5 - завершить работу программы" << "\n";
			std::cout << "Введите число соответствующее нужной операции: ";
			std::cin >> q;
			if (q == 1)
			{
				std::cout << "Выберите направление сортировки 1 - по возрастанию, 2 - по
убыванию: ";
					std::cin >> t;
				if (t == 1)
				{
					d.Sort(-1);
				}
				else if (t == 2)
				{
					d.Sort(1);
				}
				else
				{
					std::cout << "Введите правильное значение" << "\n";
				}
			}
			else if (q == 2)
			{
				std::cout << "Выберите, что будете искать 1 - минимум, 2 - максимум: ";
				std::cin >> t;
				if (t == 1)
				{
					std::cout << "Ваш минимум: " << d.find_min() << "\n";
				}
				else if (t == 2)
				{
					std::cout << "Ваш максимум: " << d.find_max() << "\n";
				}
				else
				{
					std::cout << "Введите правильное значение" << "\n";
```

```
                        }
                }
                else if (q == 3)
                {
                        std::cout << "Введите нужный элемент: ";
                        std::cin >> t;
                        std::cout << "Индекс вашего элемента: " << d.find_index(t) << "\n";
                }
                else if (q == 4)
                {
                        d.print_to_Screen();
                }
                else
                {
                        for (int i = 0; i < n; i++)
                        {
                                delete[] d.data[i];
                        }
                        delete[] d.data;
                        break;
                }
        }
}
else if (q == 2)
{
        Function_class<drob> d;
        drob m;
        std::cout << "Введите размер массива: ";
        std::cin >> n;
        d.data = new drob * [n];
        d.len = n;
        for (int i = 0; i < n; i++)
        {
                if (i % 2 == 0)
                {
                        d.data[i] = new drob(i + 1, i + 2);
                }
                else
                {
                        d.data[i] = new drob2(i + 1, i + 2, 'c');
                }
        }
        d.print_to_Screen();
        while (true)
        {
                std::cout << "Выберите нужное действие: " << "\n";
                std::cout << "1 - отсортировать массив" << "\n";
                std::cout << "2 - найти минимум или максимум" << "\n";
                std::cout << "3 - получить индекс элемента по его значению" << "\n";
                std::cout << "4 - показать массив" << "\n";
                std::cout << "5 - завершить работу программы" << "\n";
                std::cout << "Введите число соответствующее нужной операции: ";
                std::cin >> q;
                if (q == 1)
                {
                        std::cout << "Выберите направление сортировки 1 - по возрастанию, 2 - по
убыванию: ";
                                std::cin >> t;
                        if (t == 1)
                        {
                                d.Sort(-1);
                        }
                        else if (t == 2)
                        {
                                d.Sort(1);
                        }
                        else {
                                std::cout << "Введите правильное значение" << "\n";
                        }
```

```
                }
                else if (q == 2)
                {
                        std::cout << "Выберите, что будете искать 1 - минимум, 2 - максимум: ";
                        std::cin >> t;
                        if (t == 1)
                        {
                                std::cout << "Ваш минимум: " << d.find_min() << "\n";
                        }
                        else if (t == 2)
                        {
                                std::cout << "Ваш максимум: " << d.find_max() << "\n";
                        }
                        else
                        {
                                std::cout << "Введите правильное значение" << "\n";
                        }
                }
                else if (q == 3)
                {
                        std::cout << "Введите нужный элемент, сначала введите числитель в виде
целого числа, а потом знаминатель в таком же виде : ";
                                std::cin >> m;
                        std::cout << "Индекс вашего элемента: " << d.find_index(m) << "\n";
                }
                else if (q == 4)
                {
                        d.print_to_Screen();
                }
                else
                {
                        for (int i = 0; i < n; i++)
                        {
                                delete[] d.data[i];
                        }
                        delete[] d.data;
                        break;
                }
        }
    }
}
```

tests.cpp

```
#define _CRTDBG_MAP_ALLOC
#include "Function_class.h"
int main()
{
        setlocale(LC_ALL, "Russian");
        Function_class<int> f;
        std::cout << "Если сейчас дважды будет выведен массив в разном порядке, то функции сортировки
и вывода работают правильно." << "\n";
                int n = 10;
        f.data = new int* [n];
        f.len = n;
        for (int i = 0; i < f.len; i++)
        {
                f.data[i] = new int(i + 1);
        }
        f.Sort(1);
        f.print_to_Screen();
        f.Sort(-1);
        f.print_to_Screen();
        for (int i = 0; i < n; i++)
        {
                delete[] f.data[i];
        }
        delete[] f.data;
        if (_CrtDumpMemoryLeaks())
        {
                std::cout << "Утечка памяти обнаружена." << "\n";
```

```
	}
	else
	{
		std::cout << "Утечка памяти не обнаружена." << "\n";
	}
	std::cout << "Нажмите любую клавишу для завершения теста." << "\n";
	std::cin >> n;
}
```

4. Результати роботи програми

Результати роботи програми:

```
Выберите тип данных который будет использованн в массиве:
1 - int
2 - дробное число (кастомный тип данных)
Введите цисло соответствующее нужному типу данных: 1
Введите размер массива: 10
Вот ваш массив данных:
1 1
2 2
3 3
4 4
5 5
6 6
7 7
8 8
9 9
10 10
Выберите нужное действие:
1 - отсортировать массив
2 - найти минимум или максимум
3 - получить индекс элемента по его значению
4 - показать массив
5 - завершить работу программы
Введите число соответствующее нужной операции:
```

Результати тестів:

```
Если сейчас дважды будет выведен массив в разном порядке, то функции сортировки и вывода работают правильно.
Вот ваш массив данных:
1 10
2 9
3 8
4 7
5 6
6 5
7 4
8 3
9 2
10 1
Вот ваш массив данных:
1 1
2 2
3 3
4 4
5 5
6 6
7 7
8 8
9 9
10 10
Утечка памяти не обнаружена.
Нажмите любую клавишу для завершения теста.
```

5. Висновки

При виконанні даної лабораторної роботи було створено шаблонний клас, при цьому було використано ключове слово template, але на цей раз воно було розмішене перед оголошенням класу і перед його функціями реалізація яких записана поза тіло класу.

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