Автор: Наймитенко С. КІТ-119а

Дата: 20 квітня 2020

# **Лабораторна робота №11**

**ШАБЛОННІ КЛАСИ**

Мета роботи: 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 класспадкоємець).

.

**Опис змінних**

Int\*\* array – масив елементів;

Int min\_value – мінімальний елемент масиву;

Int index – індекс елемента котрий ви хочете повернути;

MyClass obj – об’єкт шаблонного класу;

Bool flag – флаг для введення вірного індексу елемента;

String name – назва книги;

String author – ім’я автора книги;

Int year\_creating – рік написання книги

Опис функцій

bool Test\_Check\_index() – тест функції перевірки індексу данного елементу;

bool Test\_Min\_value() - визначення мінімального елементу в масиві;

void arrayfloat() – функція для типу float;

Book\* createArray() – створення масиву типу користувача;

float\* Create\_arr() – створення масиву стандартного типу;

T2 Min\_value(T1 array, T2 min\_value) – визначення мінімального значення в масиві;

T1 Sort(T1 array, int choice) – сортуванння масиву;

int Check\_index(T1\* array, T1 value) – перевірка індексу елемента;

void Output\_array(T1 array) – виведення на екран;

**Текст програми**

Файл Base\_Class.h

#pragma once

#include"MyClass.h"

class Book

{

protected:

string name;

string author;

int year\_creating;

public:

Book();

Book(string name, string surname, int age);

Book(const Book& obj);

virtual ~Book();

friend ostream& operator<<(ostream& output, const Book& obj) noexcept;

friend istream& operator>>(istream& input, Book& obj) noexcept;

bool operator==(const Book obj) const noexcept;

bool operator<(const Book obj) const noexcept;

bool operator>(const Book obj) const noexcept;

virtual string getInfo() const ;

};

Файл Inheritor\_Class.h

#pragma once

#include"Base\_Class.h"

class Info final : public Book

{

private:

int size;

public:

Info();

Info(string name, string surname, int age, int course);

Info(const Info& obj);

~Info() override final;

string getInfo() const override final;

};

Файл MyClass.h

#pragma once

#define CRTDBG\_MAP\_ALLOC

#include <crtdbg.h>

#define DEBUG\_NEW new(\_NORMAL\_BLOCK, FILE, \_\_LINE)

#include <iostream>

#include <string>

#include <iomanip>

#include <sstream>

using std::ostream;

using std::istream;

using std::string;

using std::cout;

using std::cin;

using std::endl;

using std::setw;

using std::exception;

using std::istringstream;

using std::stringstream;

#define SIZE 5

template <typename T>

bool A\_more(T a, T b) { return a > b; }

template <typename T>

bool B\_more(T a, T b) { return b > a; }

template <typename T>

class MyClass

{

protected:

T\*\* array;

public:

typedef bool (func)(T, T);

void Create\_arr()

{

typedef bool (func)(T, T);

cout << "Заполните масив данными. Пожалуйста, не повторяйтесь." << endl;

T\* arr = new T[SIZE];

for (int i = 0; i < SIZE; i++)

{

cin >> arr[i];

}

for (size\_t i = 0; i < SIZE; i++)

{

array[i] = &arr[i];

}

}

void Output\_array()

{

cout << "Вывод масива на экран" << endl;

for (int i = 0; i < SIZE; i++)

{

cout << setw(3) << \*array[i];

}

cout << endl;

}

int Check\_index(T value)

{

int index = SIZE + 1;

for (int i = 0; i < SIZE; i++)

{

if (\*array[i] == value)

{

index = i;

i = SIZE;

}

}

if (index == SIZE + 1)

throw 1;

return index;

}

void Sort (func condition)

{

auto temp = array[0];

cout << "Проиходит сортировка масива" << endl;

for (size\_t i = 0; i < SIZE; i++)

{

for (size\_t j = 0; j < SIZE; j++)

{

if (condition(\*array[i], \*array[j]))

{

temp = array[i];

array[i] = array[j];

array[j] = temp;

}

}

}

}

T Min\_value()

{

T min\_value = \*array[0];

for (int i = 1; i < SIZE; i++)

{

if (\*array[i] < min\_value)

min\_value = \*array[i];

}

cout << "Минимальный елемент в масиве " << min\_value << endl;

return min\_value;

}

MyClass(T\*\* array)

{

this->array = array;

}

~MyClass()

{

delete[] array;

}

};

Файл Base\_Class.cpp

#include "Base\_Class.h"

Book::Book() : name("TOP\_BOOK"), author("Pushkin"),year\_creating(1999) {}

Book::Book(string name, string author, int year\_creating) : name(name), author(author), year\_creating(year\_creating) {}

Book::Book(const Book& obj) : name(obj.name), author(obj.author), year\_creating(obj.year\_creating) {}

Book:: ~Book() {}

string Book::getInfo() const

{

stringstream temp;

temp << setw(15) << name << setw(20) << author << setw(6) << year\_creating << endl;

return temp.str();

}

ostream& operator<<(ostream& output, const Book& obj) noexcept

{

output.setf(std::ios::left);

output << obj.getInfo();

return output;

}

istream& operator>>(istream& input, Book& obj) noexcept

{

input >> obj.year\_creating;

return input;

}

bool Book::operator==(const Book obj) const noexcept

{

return this->year\_creating == obj.year\_creating;

}

bool Book::operator<(const Book obj) const noexcept

{

return this->year\_creating < obj.year\_creating;

}

bool Book::operator>(const Book obj) const noexcept

{

return this->year\_creating > obj.year\_creating;

}

Файл Inheritor\_Class.cpp

#include"Inheritor\_Classr.h"

Info::Info() : Book(), size(2) {}

Info::Info(string name, string author, int year\_creating, int size) : Book(name, author, year\_creating), size(size) {}

Info::Info(const Info& obj) : Book(obj),size(obj.size) {}

Info::~Info() {}

string Info::getInfo() const

{

stringstream temp;

temp << setw(15) << name << setw(20) << author << setw(6) << year\_creating << setw(6) << size<<endl;

return temp.str();

}

Файл Test.cpp

#include"MyClass.h"

bool Test\_Check\_index();

bool Test\_Min\_value();

int main()

{

setlocale(LC\_ALL, "Rus");

bool result;

result = Test\_Check\_index();

if (result) /\*\*< перевірка тесту\*/

cout << "Test\_Check\_index пройден успешно!" << endl;

else

cout << "Test\_Check\_index не пройден!" << endl;

result = false;

result = Test\_Min\_value();

if (result) /\*\*< перевірка тесту\*/

cout << "Test\_Min\_value пройден успешно!" << endl;

else

cout << "Test\_Min\_value не пройден!" << endl;

return 0;

}

bool Test\_Check\_index()

{

int\*\* array = new int\*[SIZE];

int nums[SIZE] = { 2, 3, 1, -4, 0 };

for (size\_t i = 0; i < SIZE; i++)

{

array[i] = &nums[i];

}

MyClass<int> obj(array);

if (2 == obj.Check\_index(1))

{

return true;

}

else

{

return false;

}

}

bool Test\_Min\_value()

{

int\*\* array = new int\* [SIZE];

int nums[SIZE] = { 1, 4, 13, 23, 0 };

for (size\_t i = 0; i < SIZE; i++)

{

array[i] = &nums[i];

}

MyClass<int> obj(array);

int value = 15;

if (0 == obj.Min\_value())

{

return true;

}

else

{

return false;

}

}

Файл Main.cpp

#include"MyClass.h"

#include"Inheritor\_Classr.h"

void arrayint();

void arrayclass();

Book\*\* createArray( );

int main()

{

setlocale(LC\_ALL, "Rus");

arrayint();

arrayclass();

}

void arrayint()

{

int\*\* array = new int\*[SIZE];

MyClass<int> obj(array);

obj.Create\_arr();

obj.Output\_array();

obj.Min\_value();

int value;

cout << "Введите елемент индекс которого вы хотите узнать" << endl;

int index;

bool flag = true;

while (flag)

{

try

{

cin >> value;

index = obj.Check\_index(value);

cout << "Индех елемента который вы хотели узнать " << index << endl;

flag = false;

}

catch (const int)

{

cout << "Такого елемента нет в масиве, поробуйте еще раз" << endl;

}

}

obj.Sort(A\_more);

obj.Output\_array();

}

void arrayclass()

{

Book object;

Book\*\* array = createArray();

MyClass<Book> obj(array);

obj.Output\_array();

obj.Min\_value();

int value;

cout << "Введите елемент индекс которого вы хотите узнать" << endl;

int index;

bool flag = true;

while (flag)

{

try

{

cin >> object;

index = obj.Check\_index(object);

cout << "Индех елемента который вы хотели узнать " << index << endl;

flag = false;

}

catch (const int)

{

cout << "Такого елемента нет в масиве, поробуйте еще раз" << endl;

}

}

obj.Sort(B\_more);

obj.Output\_array();

for (int i = 0; i < SIZE; i++)

{

delete array[i];

}

}

Book\*\* createArray()

{

Book\*\* array = new Book \* [SIZE];

for (size\_t i = 0; i < SIZE; i++)

{

if (i == 0)

{

array[i] = new Book();

}

else if (i == 1)

{

array[i] = new Info("voina\_i\_mir", "Kto\_to", 18, 100);

}

else if (i == 2)

{

array[i] = new Book("Elisey", "Leonidovich", 45);

}

else if (i == 3)

{

array[i] = new Info("Dmitry", "Aleksandrovich", 23, 4);

}

else if (i == 4)

{

array[i] = new Book("Oliver", "Lvovich", 32);

}

}

return array;

}

4 РЕЗУЛЬТАТ РОБОТИ ПРОГРАМИ

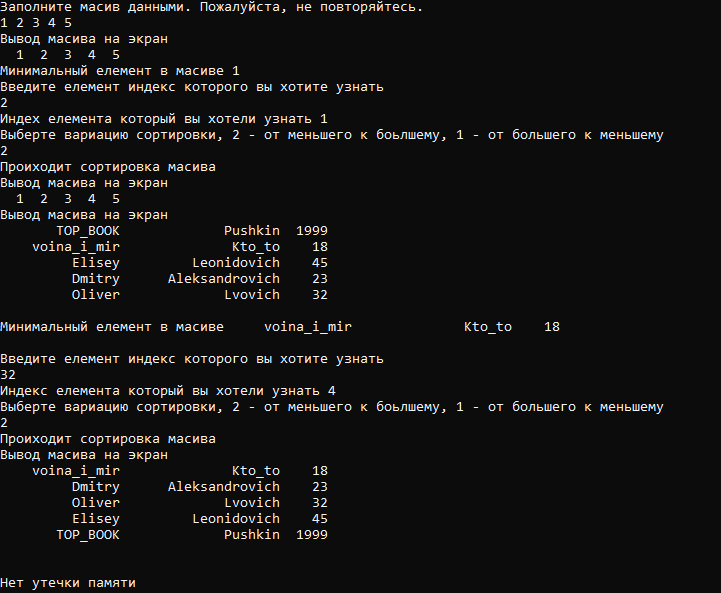


Рис. 1 - Результат роботи основоного завдання



Рис. 2 - Результат тестування методів

ВИСНОВКИ

Поширили знання у шаблонізації (узагальненні) на основі вивчення шаблонних класів та створення власних шаблонних типів. Виконали завдання згідно з варіантом 14 у списку.