Звіт

Автор: Чугунов В.Ю

КІТ-119а

Лабораторна робота №10

**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 усі необхідні дані передаються безпосередньо у функції. Клас має виконувати наступні дії:  
 - виводити вміст масиву на екран;  
 - визначати індекс переданого елементу в заданому масиві;  
 - сортувати елементи масиву;  
 - визначати значення мінімального елементу масиву.  
 Також необхідно продемонструвати роботу програми з використанням типів, створених користувачем.

2 РОЗРОБКА АЛГОРИТМІВ ФУНКЦІЙ

**Константи:**const int size = 5 – кількість елементів в масивах

**Змінні:**

Template<int> ITemp – об’єкт шаблонного класу, орієнтований на тип *int*

int intArr[size] – масив цілих чисел

Template<double> FTemp - об’єкт шаблонного класу, орієнтований на тип *double*

double doubleArr[size] – масив реальних чисел

Template<char> CTemp - об’єкт шаблонного класу, орієнтований на тип *char*

char charArr[size] – масив символів

Template<Point> PTemp - об’єкт шаблонного класу, орієнтований на тип *Point*

Point pointArr[size] – масив об’єктів типу *Point*

**Клас Point  
 private поля:**int x – змінна, що зберігає координату x

int y – змінна, що зберігає координату y

**Оператори:  
 ostream& operator<< ( ostream& os, const Point& obj )** Призначення: виведення інформації у форматі: (0; 0).

**const bool operator< ( const Point& obj ) const** Призначення: перевантаження оператора « < »

**const bool operator> ( const Point& obj ) const** Призначення: перевантаження оператора « > »

**const bool operator== ( const Point& obj ) const** Призначення: перевантаження оператора « == »

**Клас Template<typename T>**

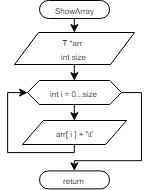
**Функції:  
 void ShowArray ( const T \*arr, const int size )** Призначення: виведення елементів масиву на екран.  
 Опис роботи: за допомогою циклу функція виводить кожен елемент масиву на екран.  
 Схема алгоритму подана на рис. 1.

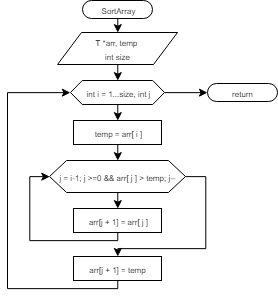
**void SortArray ( T \*arr, const int size )** Призначення: сортування масиву.  
 Опис роботи: функція використовує алгоритм сортування методом вставки.  
 Схема алгоритму подана на рис. 2.

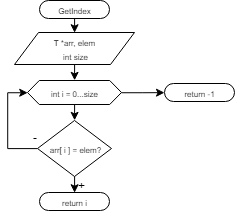
**Int GetIndex ( const T \*arr, const T elem, cons tint size)** Призначення: повернення індексу вхідного елементу.  
 Опис роботи: використовуючи цикл функція порівнює всі елементи масиву зі вхідним елементом. Якщо знаходить, то повертає індекс цього елементу, а якщо ні, то повертає -1.  
 Схема алгоритму подана на рис. 3.

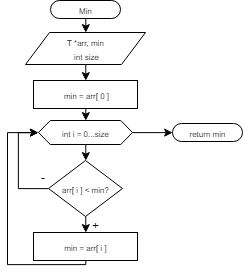
**T Min ( const T \*arr, const int size )** Призначення: пошук мінімального значення масиву.  
 Опис роботи: використовуючи цикл функція порівнює всі елементи масиву між собою, та заносить мінімальне у змінну, яку згодом повертає.  
 Схема алгоритму подана на рис. 4.

**Схеми алгоритмів функцій та методів**

  
Рисунок 1 – Схема алгоритму функції ShowArray

  
Рисунок 2 – Схема алгоритму функції SortArray

  
Рисунок 3 – Схема алгоритму функції GetIndex

  
Рисунок 4 – Схема алгоритму функції Min

3 ТЕКСТ ПРОНРАМИ

**Текст файлу main.cpp**#include "Template.h"

#include "Point.h"

int main()

{

const int size = 5;

cout << "#############################################\n";

Template<int> ITemp;

int intArr[size] = { 5, 84, 2, 46, 35 };

cout << "Array of integers:\n";

ITemp.ShowArray(intArr, size);

ITemp.SortArray(intArr, size);

cout << "\nSorted array:\n";

ITemp.ShowArray(intArr, size);

cout << "\nGet index by value: 46 = ";

printf("intArr[%d]\n", ITemp.GetIndex(intArr, 46, size));

cout << "Minimal value: " << ITemp.Min(intArr, size) << endl;

cout << "#############################################\n";

Template<double> FTemp;

double doubleArr[size] = { 8.45, 6.16, 7.95, 3.65, 5.72 };

cout << "Array of real numbers:\n";

FTemp.ShowArray(doubleArr, size);

FTemp.SortArray(doubleArr, size);

cout << "\nSorted array:\n";

FTemp.ShowArray(doubleArr, size);

cout << "\nGet index by value: 6.49 = ";

printf("floatArr[%d]\n", FTemp.GetIndex(doubleArr, 6.49, size));

cout << "Minimal value: " << FTemp.Min(doubleArr, size) << endl;

cout << "#############################################\n";

Template<char> CTemp;

char charArr[size] = "Hola";

cout << "Array of chars:\n";

CTemp.ShowArray(charArr, size);

CTemp.SortArray(charArr, size);

cout << "\nSorted array:\n";

CTemp.ShowArray(charArr, size);

cout << "\nGet index by value: o = ";

printf("charArr[%d]\n", CTemp.GetIndex(charArr, 'o', size));

cout << "Minimal value: " << CTemp.Min(charArr, size) << endl;

cout << "#############################################\n";

Template<Point> PTemp;

Point pointArr[size] = { {0, 1}, {5, 6}, {3, 4}, {2, 1}, {1, 7} };

cout << "Array of points:\n";

PTemp.ShowArray(pointArr, size);

PTemp.SortArray(pointArr, size);

cout << "\nSorted array:\n";

PTemp.ShowArray(pointArr, size);

cout << "\nGet index by value: (3 ; 4) = ";

printf("pointArr[%d]\n", PTemp.GetIndex(pointArr, {3, 4}, size));

cout << "Minimal value: " << PTemp.Min(pointArr, size) << endl;

cout << "#############################################\n";

system("PAUSE");

return 0;

}

**Текст файлу Template.h**#pragma once

#include <iostream>

using std::cout;

using std::cin;

using std::endl;

template<typename T>

class Template

{

public:

Template();

void ShowArray(const T \*arr, const int size);

void SortArray(T \* arr, const int size);

int GetIndex(const T \*arr, const T elem, const int size);

T Min(const T \*arr, const int size);

};

template<typename T>

inline Template<T>::Template()

{

}

template<typename T>

void Template<T>::ShowArray(const T \* arr, const int size)

{

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

cout << arr[i] << '\t';

}

template<typename T>

void Template<T>::SortArray(T \* arr, const int size)

{

T temp;

for (int i = 1, j; i < size; i++)

{

temp = arr[i];

for (j = i - 1; j >= 0 && arr[j] > temp; j--)

arr[j + 1] = arr[j];

arr[j + 1] = temp;

}

}

template<typename T>

int Template<T>::GetIndex(const T \* arr, const T elem, const int size)

{

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

if (arr[i] == elem)

return i;

return -1;

}

template<typename T>

inline T Template<T>::Min(const T \* arr, const int size)

{

T min = arr[0];

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

if (arr[i] < min)

min = arr[i];

return min;

}

**Текст файлу Point.h**#pragma once

#include <iostream>

using std::ostream;

class Point

{

public:

Point();

Point(int a, int b);

friend ostream& operator<< (ostream& os, const Point& obj);

const bool operator< (const Point& obj) const;

const bool operator> (const Point& obj) const;

const bool operator== (const Point& obj) const;

private:

int x;

int y;

};

ostream& operator<< (ostream& os, const Point& obj);

**Текст файлу Point.cpp**#include "Point.h"

Point::Point()

{

x = 0;

y = 0;

}

Point::Point(int a, int b)

{

x = a;

y = b;

}

const bool Point::operator<(const Point & obj) const

{

return (this->x + this->y) < (obj.x + obj.y);

}

const bool Point::operator==(const Point & obj) const

{

return (this->x + this->y) == (obj.x + obj.y);

}

const bool Point::operator>(const Point & obj) const

{

return (this->x + this->y) > (obj.x + obj.y);

}

ostream & operator<<(ostream & os, const Point & obj)

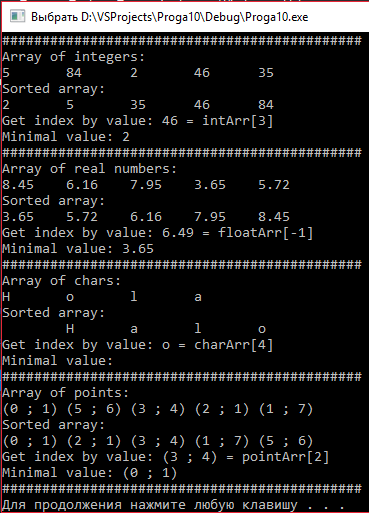
{

os << '(' << obj.x << " ; " << obj.y << ')';

return os;

}

4 РЕЗУЛЬТАТ РОБОТИ ПРОГРАМИ

  
Рисунок 5 – Результат роботи програми

ВИСНОВКИ

При виконанні лабораторної роботи було написано програму, яка може працювати з масивами будь-яких типів даних.

Використано шаблонний клас, який виконує поставлені задачі та може працювати з будь-якими типами даних.  
 Набуто практичних навичок щодо розроблення шаблонних класів.