

Міністерство освіти і науки України
Національний університет «Львівська політехніка»
Кафедра ЕОМ



Звіт
до лабораторної роботи № 8
з дисципліни: «Програмування, частина 1 (ОАП)»
на тему: «ШАБЛОНИ»
Варіант № 14

Підготув:
студент групи КІ-103
Мишак М.А.
Перевірила:
Гузинець Н. В.

Мета:

познайомитися із створенням шаблонів.

Теорія:

Шаблони являють собою схематичний опис побудови класів та функцій. Використовуючи шаблони, з'являється можливість створювати узагальнені специфікації для класів та функцій, що найчастіше носять назву параметризованих класів (generic classes) та параметризованих функцій (generic functions). Шаблони не прив'язані до конкретних типів даних і описують алгоритми, незалежно від типів даних. Дані алгоритми мають функціонувати однаково для різних типів даних. Такий опис дозволяє описати один раз функції, методи чи класи і на їх базі генерувати функції, методи і класи для кожного конкретного набору параметрів, що економить зусилля і час розробки програмного забезпечення. Після визначення загального шаблону, якщо для одного, кількох або всіх параметрів поведінка класу чи функції відрізнятиметься від описаної в загальному шаблоні, то створюється спеціалізація для конкретного набору параметрів. Спеціалізація може бути звичайною (неявною), явною або частковою.

Параметризовані функції

Для виконання схожих операцій над різними типами даних часто використовуються перевантажені функції. Якщо ж для кожного типу даних повинні виконуватися ідентичні операції, то більш компактним і зручним рішенням є використання параметризованих (шаблонних) функцій. При цьому програміст повинен написати лише один опис шаблону функції. Базуючись на типах аргументів, використаних при виклику цієї функції, компілятор буде автоматично генерувати об'єктні коди функцій, що оброблятимуть кожен тип даних.

Параметризовані класи

Визначаючи параметризований клас, ми створюємо його каркас (шаблон), що описує усі алгоритми, які використовуються класом. Фактичний тип даних, над яким проводитимуться маніпуляції, буде вказаний в якості параметру при конкретизації об'єктів цього класу. Компілятор автоматично згенерує відповідний об'єкт на основі вказаного типу. Іншими словами класи породжують об'єкти, а шаблонні класи – класи.

Завдання:

Контейнерний клас описує та забезпечує набір дій над даними параметризованого масиву, розмірність якого визначається під час роботи програми. Усі обчислення та перетворення повинні бути реалізовані у вигляді методів класу.

Варіант 14

Дана прямокутна матриця у якій розташувати елементи в рядках матриці по зростанню.

Лістинги (тексти) програм:

Main.cpp

```
#include "Matrix.h"
#include <iostream>
#include <windows.h>

int main() {
    SetConsoleOutputCP(1251);
    SetConsoleCP(1251);

    size_t n, m;
    std::cout << "Введіть розміри матриці (рядки і стовпці): ";
    std::cin >> n >> m;

    Matrix<int> mat(n, m);

    std::cout << "Введіть елементи матриці:\n";
    mat.input();

    std::cout << "Початкова матриця:\n";
    mat.print();

    // Сортуювання за зростанням
    mat.sortRows(true);

    std::cout << "Матриця після сортування рядків по зростанню:\n";
    mat.print();

    return 0;
}
```

Matrix.h

```
#pragma once
#include <vector>
#include <iostream>

template <typename T>
class Matrix {
private:
    std::vector<std::vector<T>> matrix;
    size_t rows, cols;

    // Бульбашкове сортування одного рядка
    void SortRow(std::vector<T>& row, bool ascending) {
        size_t n = row.size();
        for (size_t i = 0; i < n; ++i) {
            for (size_t j = 0; j < n - 1; ++j) {
                if ((ascending && row[j] > row[j + 1]) ||
                    (!ascending && row[j] < row[j + 1])) {
                    T temp = row[j];
                    row[j] = row[j + 1];
                    row[j + 1] = temp;
                }
            }
        }
    }

public:
    Matrix(size_t r, size_t c, const T& value = T())
        : rows(r), cols(c), matrix(r, std::vector<T>(c, value)) {}
};
```

```

    }

    void set(size_t i, size_t j, const T& value) {
        if (i < rows && j < cols)
            matrix[i][j] = value;
    }

    T get(size_t i, size_t j) const {
        if (i < rows && j < cols)
            return matrix[i][j];
        return T();
    }

    void input(std::istream& in = std::cin) {
        for (size_t i = 0; i < rows; ++i)
            for (size_t j = 0; j < cols; ++j)
                in >> matrix[i][j];
    }

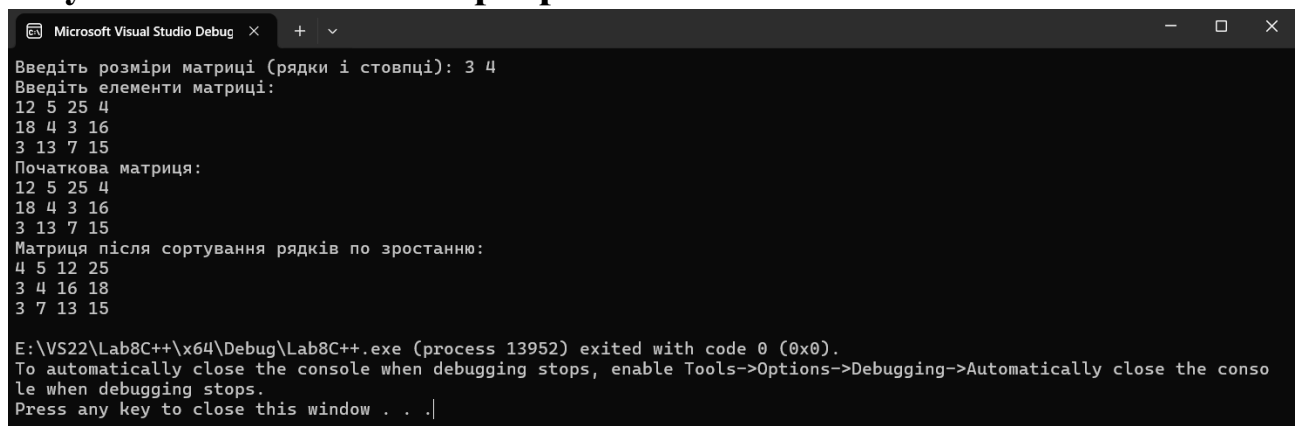
    void print(std::ostream& out = std::cout) const {
        for (const auto& row : matrix) {
            for (const auto& x : row)
                out << x << ' ';
            out << '\n';
        }
    }

    // Сортування кожного рядка без std::sort
    void sortRows(bool ascending = true) {
        for (auto& row : matrix)
            SortRow(row, ascending);
    }

    size_t numRows() const { return rows; }
    size_t numCols() const { return cols; }
};

```

Результати виконання програм:



```

Microsoft Visual Studio Debug
Введіть розміри матриці (рядки і стовпці): 3 4
Введіть елементи матриці:
12 5 25 4
18 4 3 16
3 13 7 15
Початкова матриця:
12 5 25 4
18 4 3 16
3 13 7 15
Матриця після сортування рядків по зростанню:
4 5 12 25
3 4 16 18
3 7 13 15
E:\VS22\Lab8C++\x64\Debug\Lab8C++.exe (process 13952) exited with code 0 (0x0).
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .|

```

Висновок: