МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра ЕОМ



Звіт до лабораторних робіт

з ООП

Виконав:

ст. групи КІ-15

Ковальський Н.Б.

Прийняв:

викл. Козак Н.Б.

Львів-2020

ЛАБОРАТОРНА РОБОТА № 8.

ШАБЛОНИ

**Мета:** познайомитися із створенням шаблонів.

**ТЕОРЕТИЧНІ ВІДОМОСТІ**

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 шаблона, 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. Призначенням шаблонів є створення екземплярів (instantiating) шаблону, які вже є реальними функціями чи класами. При цьому відбувається прив'язування параметрів шаблону до даних визначеного типу. Цей процес називається конкретизацією. Cпроба компілятором створити екземпляр шаблону є генерацією програми. Тому зустрічаючи спробу створити екземпляр шаблону компілятор перемикається в режим його вивчення та запам'ятовування, а це - часові витирати. Типи загального призначення, якими оперують шаблони, називаються шаблонними типами (template type), а їх сукупність параметрами шаблона (template parameters). Параметри шаблону як множина шаблонних типів може містити також преозначені і вбудовані типи С++. Шаблонний тип Т є невизначеним узагальненим типом. По мірі використання шаблонів компілятор автоматично замінить тип Т іменем реального типу. Як правило, для імені шаблонного типу використовують ідентифікатори T чи Type. Проте це не обов'язково: ім'я можна декларувати будь-яким допустимим в С++ ідентифікатором. Шаблонний тип можна повноцінно використовувати в тілі шаблону, але це не є строгою вимогою.

Шаблон допускає використання параметрів, які ініціалізуються аргументами за замовчуванням, згідно з методологією оголошення і використання таких аргументів. Типи аргументів по замовчуванні можуть бути лише преозначеними або вбудованими. Використання шаблонних типів як аргументів по замовчуванню не допускається. Під шаблон пам'ять не виділяється. Якщо екземпляр шаблону не створюється, то компілятор навіть не буде транслювати код шаблону. Це спричинює труднощі з використанням файлів заголовків, які містять лише оголошення шаблонів, а їх реалізація знаходиться у сpp-файлі. Для подолання цих недоліків треба підключати сpp-файл, а не файл заголовку, або код шаблону вносити у файл заголовку.

**Варіант-12**

Дана прямокутна матриця. Визначити номер рядка, в якому знаходиться найдовша серія з різних елементів.

**Код програми:**

**main.cpp:**

#include <iostream>

#include <iomanip>

#include "matrix.h"

using namespace std;

int main(void)

{

Matrix<int> m;

m.init\_matrix();

m.count\_regretion();

system("pause");

return 0;

}

**matrix.h:**

#ifndef matrix\_h\_\_

#define matrix\_h\_\_

#include <iostream>

#include <iomanip>

using namespace std;

template<class T>

class Matrix

{

private:

T \*\*m\_iArr;

int size\_row;

int size\_col;

public:

Matrix();

~Matrix();

void init\_matrix();

int count\_regretion();

};

template <class T>

Matrix<T>::Matrix()

{

size\_col = 0;

size\_row = 0;

m\_iArr = new T \*[size\_row];

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

{

m\_iArr[i] = new T[size\_col];

for (int j = 0; j < size\_col; j++)

{

m\_iArr[i][j] = 0;

}

}

}

template<class T>

Matrix<T>::~Matrix()

{

for (int j = 0; j < size\_row; j++)

{

delete m\_iArr[j];

}

delete[] m\_iArr;

}

template<class T>

void Matrix<T>::init\_matrix()

{

if (&m\_iArr != 0)

{

for (int j = 0; j < size\_row; j++)

{

delete m\_iArr[j];

}

delete[] m\_iArr;

}

cout << "Введіть Кількість Рядків:";

cin >> size\_row;

cout << "Введіть Кількість Стовпців:";

cin >> size\_col;

m\_iArr = new T \*[size\_row];

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

{

m\_iArr[i] = new T[size\_col];

for (int j = 0; j < size\_col; j++)

{

m\_iArr[i][j] = 0;

}

}

cout << "Введіть Вашу Матрицю:" << endl;

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

for (int j = 0; j < size\_col; j++)

{

cout << "a[" << i << "]" << "[" << j << "]" << "=";

cin >> m\_iArr[i][j];

}

}

template<class T>

int Matrix<T>::count\_regretion()

{

int count\_regretion = 0;

int i=0,j=0,k[size\_row],res;

cout<<"\nВаша Матриця:\n";

for (i = 0; i<size\_row; i++)

{

k[i]=0; for (j = 0; j<size\_col; j++)

{

cout<<setw(4)<<m\_iArr[i][j];

}

cout<<endl;

}

for (i = 0; i<size\_row; i++)

{

for (j = 0; j<size\_col-1; j++)

{

if(m\_iArr[i][j]!=m\_iArr[i][j+1]) k[i]+=1;

}

}

res=k[0]; for(i = 0;i<size\_row;i++)

{

if(res<=k[i])

{ res=k[i]; count\_regretion=i+1;

}

}

cout<<"Результат:\n";

cout<<"\nНайдовша серія з розміщених підряд різних елементів знаходить у рядку №"<<count\_regretion<<endl;

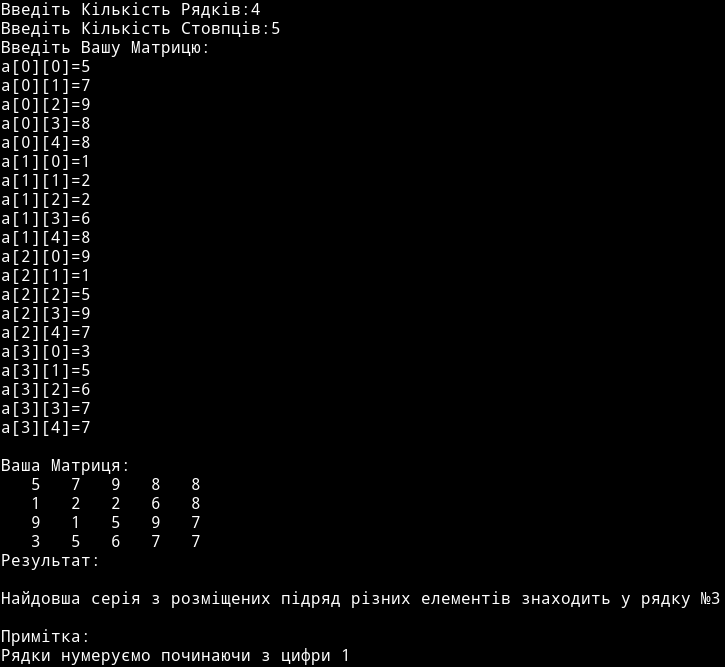
cout<<"\nПримітка:\nРядки нумеруємо починаючи з цифри 1"<<endl;

return 0;

}

#endif

**Результат**



**Висновок:** На даній лабораторній роботі я ознайомився із створенням шаблонів.