 МІНІСТЕРСТВО ОСВІТИ I НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ   
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ БІОМЕДИЧНОЇ ІНЖЕНЕРІЇ

КАФЕДРА БІОМЕДИЧНОЇ КІБЕРНЕТИКИ

**Практичне заняття №5**

з дисципліни «Об’єктно-орієнтоване програмування»

на тему: «Шаблони»

Варіант №11

**Виконав:**

студент гр. БС-52

Розюк П.В.

**Перевірив:**

ст. викладач каф. БМК

Радогуз А.В.

Зараховано від \_\_\_.\_\_\_.\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис викладача)

Київ-2017

**Завдання:**

1. Ознайомитися з тноретичними відомостями роботи із шаблонами функцій та шаблонами класів.
2. Відповідно до свого авріанту розробити шаблон класів абстрактної структури даних із визначенням заданих функцій та операцій ( в кожному шаблонному класі також має бути визначений конструктор копіювання та операція привласнення ) і відповідку діаграму класів в нотації UML, написати програму тестування. В якій перевіряється використання шаблона для стандартних типів даних.
3. Скласти і захистити звіт по роботі.

**🞏 Практичне заняття без зауважень** **🞏 Практичне заняття має зауваження:**

**🞏 несвоєчасний захист**

**🞏 присутні зауваження до UML діаграми:**

**🞏 діаграма класу не відповідає коду**

**🞏 виконані не за стандартом:**

**🞏 атрибути**

**🞏 відношення**

**🞏 багатостатність**

**🞏 інші зауваження:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**🞏 присутні зауваження до коду:**

**🞏 задача завдання вирішена хибно**

**🞏 код програми не компілюється**

**🞏 хибно задані специфікатори доступу**

**🞏 помилки у визначенні конструкторів / деструкторів**

**🞏 використано глобальні змінні**

**🞏 статичні змінні при роботі з масивами**

**🞏 оформлення коду**

**🞏 присутні зайві символи «{» та «}»**

**🞏 інші зауваження:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**🞏 невірні відповіді на запитання:**

**🞏 №1 🞏 №2 🞏 №3 🞏 №4 🞏 №5**

**🞏 №6 🞏 №7 🞏 №8 🞏 №9 🞏 №10**

**🞏 маються інші зауваження:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Лістинг програми:

List.h

#pragma once

#include <iostream>

#include <cstdlib>

#include <ctime>

#define MaxLev 2

using namespace std;

template <class T>

struct Node

{

T a;

Node \*\*x;

Node(T &a, int stage);

~Node();

};

template <class T>

class SkipList

{

private:

Node<T>\*first;

T a;

int stage;

public:

SkipList();

int Check();

bool Number\_elements(T &value);

void AddElement(T &value);

void Delete(T &value);

void DeleteSkipList();

void Print();

~SkipList();

};

template <class T>

Node<T>::Node(T &a,int stage)

{

x = new Node <T> \*[stage + 1];

memset(x, 0, sizeof(Node\*)\*(stage + 1));

this->a = a;

}

template <class T>

Node<T>::~Node()

{

delete[]x;

}

template <class T>

int SkipList<T>::Check()

{

return stage;

}

template <class T>

SkipList<T>::SkipList()

{

first = new Node<T>(a,MaxLev);

stage = 0;

}

template <class T>

SkipList<T>::~SkipList()

{

delete first;

}

template <class T>

bool SkipList<T>::Number\_elements( T &value)

{

Node <T>\*h = first;

for (int i = stage; i >= 0; i--)

{

while (h->x[i] != NULL && h->x[i]->a < value)

h = h->x[i];

}

h = h->x[0];

return h != NULL && h->a == value;

}

template <class T>

void SkipList<T>::Print()

{

if (SkipList::Check() == 0)

cout << "\nThe List is Empty!\n";

else

{

Node<T> \*h = first->x[0];

while (h != NULL)

{

cout << h->a;

h = h->x[0];

if (h != NULL)

cout << ";";

}

cout << endl;

}

}

template <class T>

void SkipList<T>::Delete(T &value)

{

Node<T>\*h = first;

Node<T>\*z[MaxLev + 1];

memset(z, NULL, sizeof(Node<T>\*) \* (MaxLev + 1));

for (int i = stage; i >= 0; i--)

{

while (h->x[i] != NULL && h->x[i]->a < value)

h = h->x[i];

z[i] = h;

}

h = h->x[0];

if (h->a == value)

{

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

{

if (z[i]->x[i] != h)

break;

z[i]->x[i] = h->x[i];

}

delete h;

while (stage > 0 && first->x[stage] == NULL)

{

stage--;

}

}

}

template <class T>

void SkipList<T>::AddElement(T &value)

{

Node<T> \*h = first;

Node<T> \*z[MaxLev + 1];

memset(z, NULL, sizeof(Node<T>\*) \* (MaxLev + 1));

for (int i = stage; i >= 0; i--)

{

while (h->x[i] != NULL && h->x[i]->a < value)

{

h = h->x[i];

}

z[i] = h;

}

h = h->x[0];

if (h == NULL || h->a != value)

{

int lvl = random();

if (lvl > stage)

{

for (int i = stage + 1; i <= lvl; i++)

{

z[i] = first;

}

stage = lvl;

}

h = new Node<T>(value, lvl);

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

{

h->x[i] = z[i]->x[i];

z[i]->x[i] = h;

}

}

}

float frand()

{

return (float)rand() / RAND\_MAX;

}

int random()

{

static bool first = true;

if (first)

{

srand((unsigned)time(NULL));

first = false;

}

int lvl = (int)(log(frand()) / log(1. - 0.5));

return lvl < MaxLev ? lvl : MaxLev;

}

template <class T>

void SkipList<T>::DeleteSkipList()

{

if (SkipList::Check() == 0)

cout << "\nThe List is Empty!\n";

else

{

delete first;

first = new Node<T>(a, MaxLev);

stage = 0;

}

}

**Main.cpp**

#include <iostream>

#include "List.h"

using namespace std;

void main()

{

SkipList<int> A;

int k, x;

while (1)

{

system("cls");

cout << "1.Add Element" << endl;

cout << "2.Delete Element" << endl;

cout << "3.Search Element" << endl;

cout << "4.Print List " << endl;

cout << "5.Delete list " << endl;

cout << "6.Exit " << endl << endl;

cout << "Input your choice : ";

cin >> k;

cout << endl;

switch (k)

{

case 1:

cout << "Input the element: ";

cin >> x;

A.AddElement(x);

if (A.Number\_elements(x))

system("pause");

break;

case 2:

cout << "Input the element: ";

cin >> x;

if (!A.Number\_elements(x))

{

cout << "Element not found" << endl;

system("pause");

break;

}

A.Delete(x);

if (!A.Number\_elements(x))

cout << "Element has been deleted!" << endl;

system("pause");

break;

case 3:

cout << "Input the element to search: ";

cin >> x;

if (A.Number\_elements(x))

cout << "Element " << x << " is in the list" << endl;

else

cout << "Element not found" << endl;

system("pause");

case 4:

A.Print();

system("pause");

break;

case 5:

A.DeleteSkipList();

cout << " The list has been deleted " << endl;

system("pause");

break;

case 6:

exit(1);

break;

default:

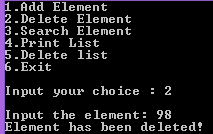
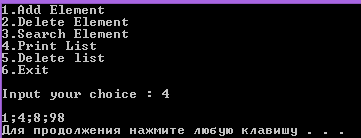
cout << "Error" << endl;

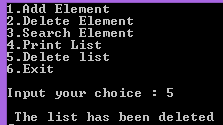
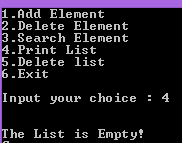
system("pause");

}

}

}



**UML-діаграма:**



**Контрольні питання**

1. **Для чого використовуються шаблони?**

Щоб відокремити загальний алгоритм від його реалізації стосовно до конкретних типів даних.

1. **Що таке шаблон класів, як його визначити в C++?**

Шаблон класу- опис, за яким компілятор створює клас для заданих вхідних параметрів.

template <class Type>

class ClassName

{ //тіло класу };

При створенні об’єкту необхідно вказати спеціалізацію.

ClassName <int> obj;

1. **Що таке шаблон функцій, як його визначити в C++?**

Шаблон функції- узагальнений опис функції (можна використовувати одну й ту саму функцію для обробки даних різних типів)

template <class Type>

тип\_поверт.знач ім’я\_функції(список параметрів)

{ //тіло функції }

1. **Що таке конкретизація шаблона? В яких випадках необхідне виконання явної конкретизації шаблона?**

Конкретизація шаблона- явне вказання компілятору типу параметра шаблона. Явна конкретизація необхідна в тому випадку, коли компілятор не може реалізувати тип, для якого він не може конкретизувати функцію або для створення класу на основі шаблону класу.

1. **Що таке спеціалізація шаблона, коли необхідно її виконувати?**

Спеціалізація шаблонів – це створення в коді окремої реалізації функції для визначених типів даних або реалізації шаблона класів для конкретних значень його параметрів. Її треба виконувати, коли компілятор не здатний побудувати функцію по шаблону, або коли шаблон класів не придатний для конкретизації визначеним типом даних або якщо конкретизація шаблону класів не ефективна по реалізації.

1. **Для чого використовують параметри за замовченням для шаблону класів?**

Щоб при створенні об’єкту не вказувати спеціалізацію, коли ми хочемо використовувати той тип даних, який стоїть за замовчуванням. В іншому випадку потрібно виконувати спеціалізацію.

1. **Чи можна викликати параметризовану функцію без параметрів?**

Можна, але коли компілятор сам може визначити тип вхідних даних.

1. **Чи може бути порожнім список параметрів шаблона? Відповідь пояснити.**

Ні, так як при цьому втрачається можливість параметризації і тому компілятор не може працювати з даною функцією.

1. **Чи можна за допомогою шаблона створити функцію з таким самим ім’ям, як і в явно визначеної функції?** **Відповідь пояснити.**

Так. Коли в коді викликається функція, то спочатку відбувається пошук по явно визначеним функціям, потім пошук по шаблонних функціях, а в разі відсутності підходящої функції відбувається спроба приведення типів у явно визначеної функції.

1. **Чи можуть шаблони класів містити віртуальні функції?**

Ні.