 МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ   
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ БІОМЕДИЧНОЇ ІНЖЕНЕРІЇ

КАФЕДРА БІОМЕДИЧНОЇ КІБЕРНЕТИКИ

**Компьютерный практикум №5**

з дисципліни «Об'єктно-орієнтоване програмування»

# на тему: «Шаблони»

Варіант №7

**Виконав:**

студент гр. БС-03

Затуловский Г. А.

**Перевірив:**

ас. каф. БМК Бабенко В.О.

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

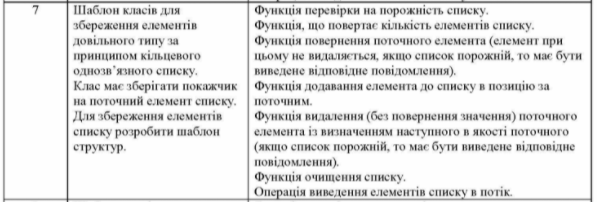
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

                      (підпис викладача)

Київ-2022

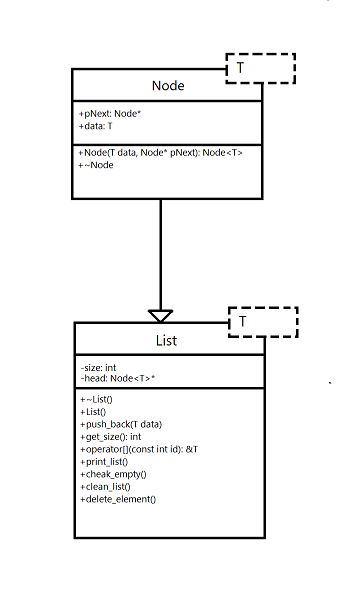
**Завдання:**

1. Ознайомитися з теоретичними відомостями роботи із шаблонами та шаблонами класів

2. Відповідно до свого варіанту розробити шаблон класiв абстрактной структури даних із вначенням заданих функцій та операцій (в кожному пшб ошому клас також мае бути визначений конструктор копіювання та операція привласнення) і відповідну діаграму класів в нотац UA.кати програму тестування, в якій перевіряється використання шаблона для стандартних типів даних:

3. Скласти і захистити звіт по роботі

**UML діаграма:**

****

**Код програми:**

**OOP.cpp**

#include "List.h"

#include <iostream>

#include <string>

using namespace std;

//template<typename T>

int main()

{

List<int> lst;

bool prog = true;

int choice = 1;

int input = 0;

cout << "1 - check for emptiness\n"

<< "2 - size list\n"

<< "3 - return the current element\n"

<< "4 - add new element\n"

<< "5 - delete element\n"

<< "6 - clean list\n"

<< "7 - print list\n"

<< "8 - exit\n" << endl;

while (prog)

{

cin >> choice;

switch (choice)

{

case 1:

lst.cheak\_empty();

break;

case 2:

cout << "List size: " << (lst.get\_size()) << endl;

break;

case 3:

cout << "Current index: ";

cin >> input;

cout<<"Current element: " << lst[input] << endl;

break;

case 4:

cout << "Input data: ";

cin >> input;

lst.push\_back(input);

break;

case 5:

cout << "Input index of element wich you want delete: ";

input =(lst.get\_size()+1);

while (input > lst.get\_size() && lst.get\_size() != 0)

{

cin >> input;

lst.delete\_element(input);

}

break;

case 6:

lst.clean\_list();

cout << "List cleared" << endl;

break;

case 7:

lst.print\_list();

break;

case 8:

lst.clean\_list();

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

prog = false;

break;

default:

cout << "Anoun command" << endl;

break;

}

}

return 0;

}

**List.h**

#pragma once

#include <iostream>

#include <string>

#include "Node.h"

using namespace std;

template<typename T>

class List

{

private:

int size;

Node<T>\* head;

public:

~List();

List();

void push\_back(T data);

int get\_size();

T& operator[](const int id);

void print\_list();

void cheak\_empty();

void clean\_list();

void delete\_element(int id);

};

template<typename T>

inline List<T>::~List()

{

}

template<typename T>

inline List<T>::List()

{

size = 0;

head = nullptr;

}

template<typename T>

inline void List<T>::push\_back(T data)

{

if (head == nullptr)

{

head = new Node<T>(data);

}

else

{

Node<T>\* current = this->head;

while (current->pNext != nullptr)

{

current = current->pNext;

}

current->pNext = new Node<T>(data);

}

size++;

}

template<typename T>

int List<T>::get\_size()

{

return size;

}

template<typename T>

inline T& List<T>::operator[](const int id)

{

int count = 0;

Node<T>\* current = this->head;

while (current != nullptr)

{

if (count == id)

{

return current->data;

}

current = current->pNext;

count++;

}

}

template<typename T>

inline void List<T>::print\_list()

{

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

{

cout << operator[](i)<< " -> ";

}

cout <<"NULL"<< endl;

}

template<typename T>

inline void List<T>::cheak\_empty()

{

if ((get\_size()) == 0)

{

cout << "The list is empty" << endl;;

}

else

{

cout << "The list is not empty" << endl;

}

}

template<typename T>

inline void List<T>::clean\_list()

{

while (size)

{

Node <T> \*temp = head;

head = head->pNext;

delete temp;

--size;

}

}

template<typename T>

inline void List<T>::delete\_element(int id)

{

Node<T>\* current = this->head;

int count = 0;

int i = 0;

id--;

while (current != nullptr && i!=1)

{

if (id == -1)

{

Node <T>\* temp = head;

head = head->pNext;

delete temp;

i++;

size--;

}

else if(count == id)

{

Node<T>\* temp = current->pNext;

current->pNext = temp->pNext;

delete temp;

size--;

i++;

}

current = current->pNext;

count++;

}

}

**Node.h**

#pragma once

#include <iostream>

#include <string>

using namespace std;

template<typename T>

class Node

{

private:

public:

Node\* pNext;

T data;

Node(T data, Node\* pNext = nullptr);

~Node();

};

template<typename T>

inline Node<T>::Node(T data, Node\* pNext)

{

this->data = data;

this->pNext = pNext;

}

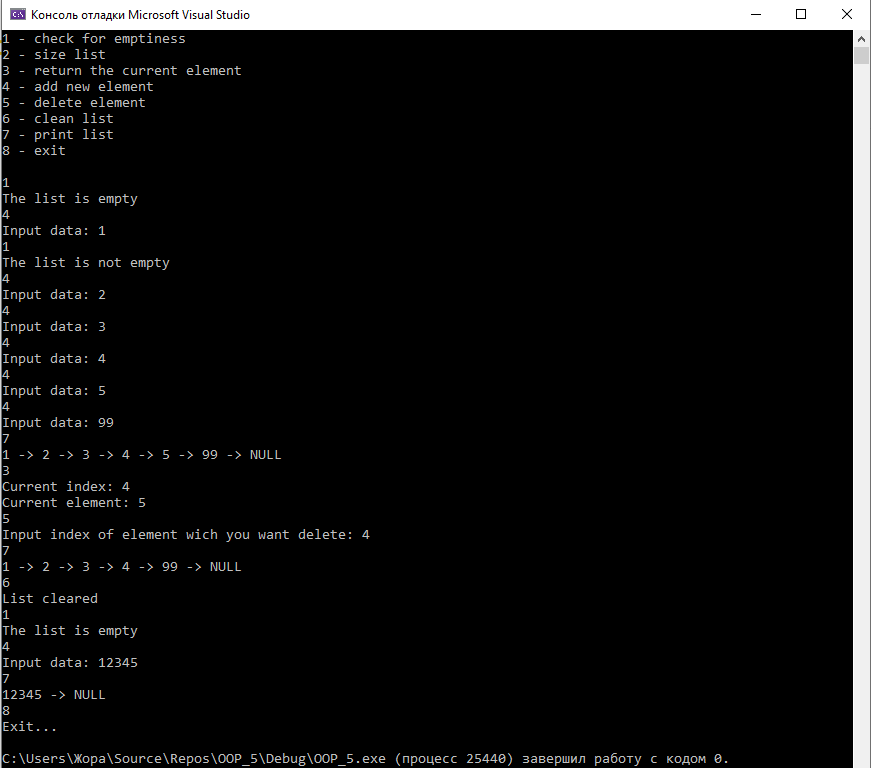
template<typename T>

inline Node<T>::~Node()

{

}

**Результати:**



**Контрольні запитання**

**1. Для чого використовують шаблони?**

Шаблон — специфікація форми подання та правил редагування елемента даних за допомогою рядка символів, у якій кожен символ вказує на допустимий вид символу або на предмет виконання редагування для відповідної позиції значення елемента.

**2 Що таке шаблон класів, як його визначити в С++?**

Шаблони класу є параметризованими типами. Шаблони класу використовують для створення сімейства класів, які впливають на тип.

Формат оголошення шаблона класів:

template<typename T1, T2, … >

class name

{

//визначення класу

};

**3. Що таке шаблон функцій, як його визначити в С++?**

Шаблони функцій - це узагальнений опис поведінки функцій, які можуть викликатися для об'єктів різних типів. Іншими словами, шаблон функції (шаблонна функція, узагальнена функція) є сімейством різних функцій (або опис алгоритму).

template< typename T >

void MySwap( T& a, T& b )

{

T c(a);

a = b;

b = c;

}

int main()

{

}

**4. Що таке конкретизація шаблона? В яких випадках необхідне виконання яинот конкретизан шаблона?**

Конкретизація шаблона – це створення за шаблоном функції, його конкретна реалізація для конкретних типів вхідних параметрів. Явна конкретизація шаблона – це явна вказівка компілятору типів параметрів шаблона, коли компілятор не може розпізнати тип, для якого він повинен конкретизувати функцію

**5. Що таке спеціалізація шаблона, коли й необхідно виконувати?**

Спеціалізація шаблону — Вона дозволяє задавати альтернативні реалізації шаблону, коли значення параметрів шаблону є певними наперед заданими типами.

Спеціалізація шаблону зазвичай використовується для:

* оптимізації — спеціалізована версія шаблону матиме оптимальніший код за рахунок знання особливостей типу;
* спрощення коду — винесення часткових випадків у окремі реалізації спростить код (подібно до використання патерну проектування стратегія).

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

У контексті оголошення шаблону функції, службове слово class не має жодного особливого смислового навантаження. Справа в тому, що аналогічна конструкція використовується також і для оголошення шаблону класу, де ключове слово class грає свою особливу роль.

У заголовку шаблону імена параметрів шаблону повинні бути унікальні. Параметрами шаблонів можуть бути: параметри-типи, параметри звичайних типів, параметри-шаблони. Для параметрів будь-якого типу можна вказувати значення за замовчуванням.

Таким чином, шаблон Buffer за замовчуванням визначено з типом int для елементів створюваного буфера і значенням 10 для параметра шаблона, який визначає розмір цього буфера.При наявності декількох параметрів шаблона всі параметри, що розташовані після першого параметра за замовчуванням, повинні бути теж параметрами за замовчуванням.

**7. Чи можна викликати параметризовану функцію без шраметрів?**

Так. У випадку, коли компілятор сам здатен визначити тип вхідних даних.

**8. Чи може бути порожнiм список параметрiв шаблона? Відповідь пояснити**

Ні. Список параметрів шаблонів не може бути пустим.

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

Так. Завдяки можливості перевантажити фунуцію.

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

Ні, шаблони класів містити віртуальні функції не можна.