­МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

КАФЕДРА КОМП’ЮТЕРНИХ НАУК

**КУРСОВА РОБОТА**

З ДИСЦИПЛІНИ

«ОСНОВИ ОБ'ЄКТНО-ОРІЄНТОВАНОГО ПРОГРАМУВАННЯ»

Викладач Тиркусова Н.В.

Студент Штогрін В.О.

Група ІН-11

Варіант 20

**Суми – 2022**

**ЗМІСТ**

[ПОСТАНОВКА ЗАДАЧІ 3](#_Toc122808005)

[ТЕОРЕТИЧНИЙ МАТЕРІАЛ 5](#_Toc122808006)

[ОПИС СТРУКТУРИ ДАНИХ 7](#_Toc122808007)

[ОПИС ФУНКЦІЙ 10](#_Toc122808008)

[ІНСТРУКЦІЯ ЩОДО РОБОТИ З ПРОГРАМОЮ 11](#_Toc122808009)

[ОПИС ФАЙЛІВ ТА ЇХ ПРИЗНАЧЕННЯ 12](#_Toc122808010)

[СПИСОК ВИКОРИСТАНИХ БІБЛІОТЕК 13](#_Toc122808011)

[ПРИКЛАД РОБОТИ ПРОГРАМИ 14](#_Toc122808012)

[ГРАФІК ТАБУЛЮВАННЯ ОБЧИСЛЮВАЛЬНОГО ПАРАМЕТРУ 18](#_Toc122808013)

[ВИСНОВКИ 19](#_Toc122808014)

[СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 20](#_Toc122808015)

[ДОДАТОК(код програми) 21](#_Toc122808016)

# **ПОСТАНОВКА ЗАДАЧІ**

Як відомо з фізики, молекула газу проходить певну відстань, перш ніж зіткнеться з іншою молекулою. Середню довжину вільного перебігу обчислюють за формулою

де N – кількість молекул;

δ – діаметр молекули.

Кількість молекул у заданому об’ємі обчислюють за формулою

де p – тиск газу;

T\_г – температура газу.

Під час описання класу передбачити такі елементи даних: середню довжину вільного пробігу, діаметр молекули, кількість молекул, тиск газу, температуру газу.

Також передбачити методи:

уведення даних із файлу;

розрахунок середньої довжини вільного пробігу;

виведення в інший файл вхідних даних і довжини пробігу.

Передбачити перевантажені операції присвоєння, порівняння (==,!=,<,<=,>,>=), а також конструктори: пустий, з ініціалізацією та копіювання.

Навести приклад програми, в якій застосовують усі конструктори, методи та перевантажені операції.

У контрольному прикладі можна використати такі значення параметрів:

Дані зчитувати з файлу. Результат розрахунку вивести в інший файл разом із вхідними даними.

Обрати один, будь-який, параметр, що має зазначений діапазон значень, та протабулювати «Обчислювальний параметр» за зміни параметра від початкового значення до кінцевого з кроком Δ. Результати обчислень вивести у файл. Побудувати графік одержаної залежності.

**ТЕОРЕТИЧНИЙ МАТЕРІАЛ**

C++- мова програмування високого рівня з підтримкою декількох парадигм програмування: об'єктно-орієнтованої, узагальненої та процедурної.

При створенні С++ прагнули зберегти сумісність з мовою С. Більшість програм на С справно працюватимуть і з компілятором С++. С++ має синтаксис, заснований на синтаксисі С. С++ - це мова програмування загального призначення з ухилом у бік системного програмування, яка підтримує абстракцію даних, об'єктно-орієнтоване програмування та узагальнене програмування.

Мова С++ багато в чому є надмножиною С. Нові можливості С++ включають оголошення у вигляді виразів, перетворення типів у вигляді функцій, оператори new і delete, тип bool, посилання, розширене поняття константності, функції, що підставляються, аргументи за умовчанням, перевизначення, простори імен, класи (включаючи і всі пов'язані з класами можливості, такі як успадкування, функції-члени (методи), віртуальні функції, абстрактні класи і конструктори), перевизначення операторів, шаблони, оператор ::, обробку винятків, динамічну ідентифікацію і багато що інше.

Деякі особливості С++ пізніше були перенесені в С, наприклад ключові слова const і inline, оголошення в циклах for і коментарі в стилі С++ («//»). У пізніших реалізаціях С також були представлені можливості, яких немає в С++, наприклад макроси vararg і покращена робота з масивами-параметрами.

Програма завжди починається дерективами препроцесору (#), які обробляється ним до початку компіляції. Задачею препроцесора є доповнення тексту програми бібліотечними функціями чи об‟єктами С++, описаними у відповідних заготовочних (header) файлах. Програма на мові С++ складається із набору окремих функцій. Обов’язковою є функція main() – вхідна точка програми (місце її розміщення у програмі значення немає).

Віртуальна функція (віртуальний метод) - це метод класу, поведінка якого може бути перевизначена у класах нащадках методами з такою самою сигнатурою. Таким чином, конкретна реалізація потрібного методу буде обрана під час виконання в залежності від типу. Чиста віртуальна функція – це функція, що використовується у абстрактних класах та не має визначення.

Абстрактний клас – це клас, який містить або наслідує без перевизначення хоча б одну чисту віртуальну функцію. Абстрактний клас у наслідувані використовується, як базовий для дочірніх класів. Об’єкт такого класу створити неможливо.

Інкапсуляція - це концепція об'єктно-орієнтованого програмування, яка з'єднує дані та функції, які керують ними, у одній структурі. Цей механізм захищає внутрішні дані від зовнішнього доступу та неправильного використання. Інкапсуляція даних призвела до розробки ідеї приховування даних, яка дозволяє контролювати доступ до внутрішньої інформації у системі.

Наслідування - це техніка, за допомогою якої один клас набуває властивостей і методів іншого класу. Клас, який набуває властивостей іншого класу, називається дочірнім класом, а клас, властивості якого набуваються, називається батьківським. Коли один клас набуває або успадковує інший клас, тоді всі властивості та методи базового класу доступні для похідного класу, щоб ми могли використовувати цей код повторно.

Поліморфізм – це властивість програмного коду змінювати свою поведінку в залежності від ситуації, яка виникає при виконанні програми. В контексті реалізації, поліморфізм – це технологія виклику віртуальних функцій, що реалізовані в ієрархічно зв’язаних класах. Ієрархія класів формується на основі механізму спадковості.

**ОПИС СТРУКТУРИ ДАНИХ**

|  |  |  |  |
| --- | --- | --- | --- |
| Ім’я параметра у формулі | Назва | Тип | Призначення |
|  | l | double | середня довжина вільного перебігу молекули |
|  | N | double | кількість молекул |
| δ | d | double | діаметр молекули |
| *Тг* | Тg | double | температура газу |
|  | p | double | тиск газу |

*Таблиця 1. Використані змінні*

*Таблиця 2. Батьківський клас Object*

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Модифікатор доступу | Тип даних | Призначення |
| l | protected | double | обчислювальний параметр |

|  |  |  |
| --- | --- | --- |
| Ім’я методу | Тип  результату | Призначення |
| enter\_file | void | введення даних |
| calc\_l | void | розрахунок обчислювального параметра |
| output\_file | void | виведення вхідних даних і результату |
| output\_l | const double | повертає значення обчислюваного параметру |

*Таблиця 3. Дочірній клас Mileage\_molecule*

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Модифікатор доступу | Тип даних | Призначення |
| N | private | double | кількість молекул |
| d | private | double | діаметр молекули |
| Тg | private | double | температура газу |
| p | private | double | тиск газу |

|  |  |  |
| --- | --- | --- |
| Ім’я методу | Тип  результату | Призначення |
| enter\_file | void | введення даних |
| calc\_l | void | розрахунок обчислювального параметра |
| output\_file | void | виведення вхідних даних і результату |
| calc\_N | double | розрахунок N |
| output\_l | const double | повертає значення обчислюваного параметру |

*Таблиця 4. Функція main*

|  |  |  |
| --- | --- | --- |
| Змінна | Тип змінної | Значення |
| Obj\_1 | Object\* | об’єкт створений з використанням покажчика на базовий клас для прикладу використання пустого конструктора та роботи з файлами |
| Obj\_2 | Object\* | об’єкт створений з використанням покажчика на базовий клас для прикладу використання конструктора з ініціалізацією |
| Obj\_3 | Mileage\_molecule | об’єкт створений для прикладу використання пустого конструктора та роботи з файлами. Використаний для перевірки перевантажених операцій |
| Obj\_4 | Mileage\_molecule | об’єкт створений для прикладу використання конструктора з ініціалізацією. Використаний для перевірки перевантажених операцій |
| Obj\_5 | Mileage\_molecule | об’єкт створений для прикладу використання конструктора копії. Використаний для перевірки перевантажених операцій |
| ofs | ofstream | для потокового введення даних у файл |
| оfs1 | ofstream | для потокового введення даних у файл |
| l\_tabulate | Mileage\_molecule | об’єкт створений для табуляції |

# 

**ОПИС ФУНКЦІЙ**

*Методи дочірнього класу Mileage\_molecule:*

* Mileage\_molecule() – пустий конструктор
* Mileage\_molecule(double d, double p, double Tg) – конструктор з ініціалізацією.
* Mileage\_molecule(const Mileage\_molecule& Mm) – конструктор копіювання.
* ~Mileage\_molecule() – деструктор.
* void enter\_file(const char\* S) – функція для введення даних з файлу
* void calc\_l() – функція для розрахунку обчислювального параметру.
* const double output\_l() – функція, яка повертає значення обчислюваного параметру.
* double calc\_N() – функція для розрахунку Kg.
* void output\_file(const char\* S)– функція для виведення вхідних даних і результату у файл.

*Перевантажені операції:*

* void operator = (Mileage\_molecule const& Mm) – перевантажений оператор присвоювання.
* friend bool operator == (Mileage\_molecule const& A, Mileage\_molecule const& B) – перевантажений оператор рівності.
* friend bool operator! = (Mileage\_molecule const& A, Mileage\_molecule const& B) – перевантажений оператор нерівності.
* friend bool operator < (Mileage\_molecule const& A, Mileage\_molecule const& B) – перевантажений оператор <.
* friend bool operator <= (Mileage\_molecule const& A, Mileage\_molecule const& B) – перевантажений оператор <=.
* friend bool operator > (Mileage\_molecule const& A, Mileage\_molecule const& B) – перевантажений оператор >.
* friend bool operator >= (Mileage\_molecule const& A, Mileage\_molecule const& B) – перевантажений оператор >=.

# **ІНСТРУКЦІЯ ЩОДО РОБОТИ З ПРОГРАМОЮ**

Для створення програми ми використовували Microsoft Visual C++.

Для роботи з програмою спочатку потрібно ввести вхідні дані до файлу Obj\_1.txt та Obj\_3.txt.

Далі відкрити файл ConsoleApplication13.сpp, у якому знаходиться код програми і запустити його.

Після цього виводиться консоль з результатом роботи програми.

Результати знаходяться у файлі Obj\_1\_res.txt та Obj\_3\_res.txt відповідно.

Результати порівняння знаходяться у файлі comparison\_res.txt, а результати табуляції у файлі tabulate\_res.txt.

Якщо під час виконання програми вхідний файл не відкрився, то виводиться повідомлення про відповідну помилку. Якщо під час виконання програми вхідні дані неправильні, то програма повідомляє про помилку і так, поки всі дані не будуть коректні.

**ОПИС ФАЙЛІВ ТА ЇХ ПРИЗНАЧЕННЯ**

* ConsoleApplication13.сpp – код програми.
* ConsoleApplication13.exe – скомпільована програма.
* Obj\_1.txt – файл, в якому зберігаються вхiднi дані для першого прикладу.
* Obj\_1\_res.txt – файл, в який записується результат першого прикладу.
* Obj\_3.txt – файл, в якому зберігаються вхiднi дані для другого прикладу.
* Obj\_3\_res.txt – файл, в який записується результат другого прикладу.
* Comparison\_res.txt - файл, в який записується результат перевантажених операцій порівняння.
* tabulate\_res.txt – файл, в який записується результат табуляції обчислювального параметру.

**СПИСОК ВИКОРИСТАНИХ БІБЛІОТЕК**

**<iostream>** - заголовний файл з класами, функціями та змінними для організації введення-виведення в мові програмування C++.

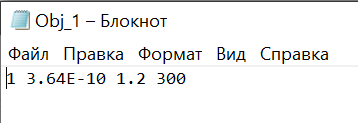
**<cmath>** - заголовний файл, який надає прототипи функцій, розроблених для виконання простих математичних операцій.

**<iomanip>** - заголовний файл, що містить засоби для форматування виведення.

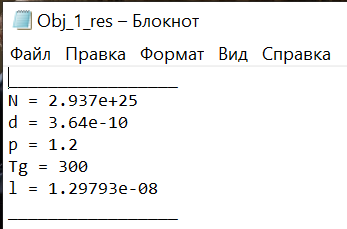
**<** **Windows.h>** - специфічний для Windows заголовний файл для мов програмування C і C++, який містить оголошення для всіх функцій в API Windows, всіх поширених макросів, що використовуються програмістами Windows, та всіх типів даних, що використовуються різними функціями та підсистемами.

**<fstream>** - заголовний файл, що надає функціонал для зчитування даних з файлу і для запису в файл.

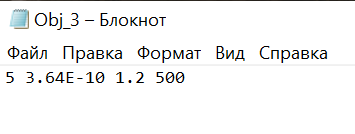
# **ПРИКЛАД РОБОТИ ПРОГРАМИ**



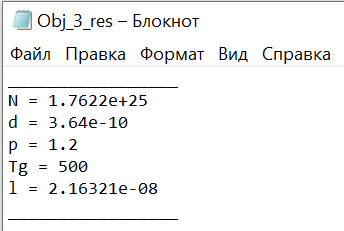
*Рисунок 1. Вхідні дані для прикладу №1.*



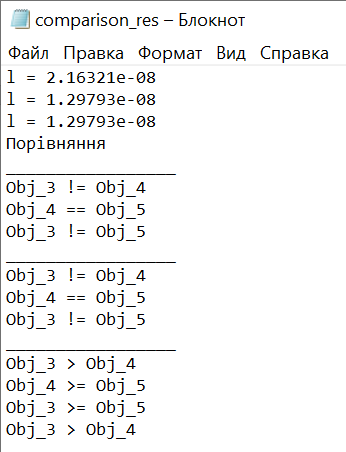
*Рисунок 2. Вихідні дані для прикладу №1.*



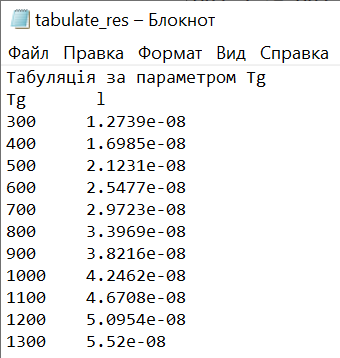
*Рисунок 3. Вхідні дані для прикладу №2.*



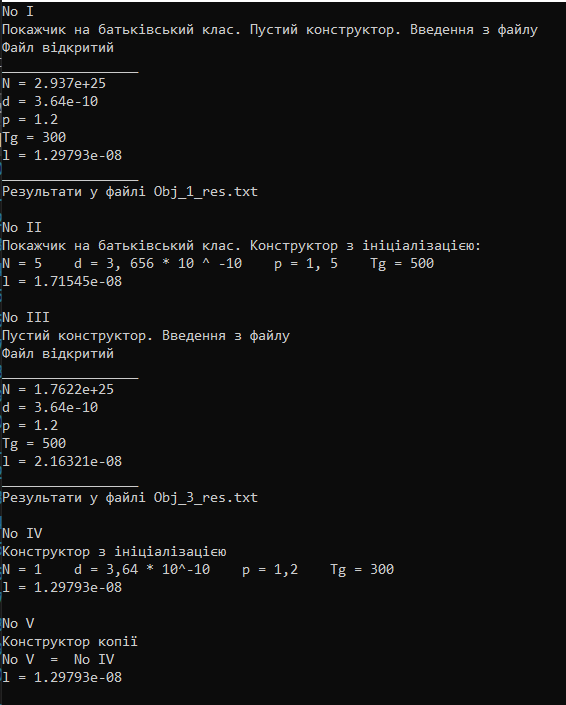
*Рисунок 4. Вихідні дані для прикладу №2.*



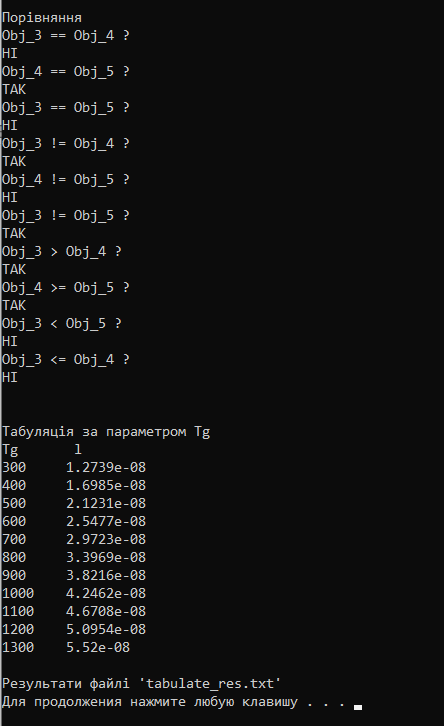
*Рисунок 5. Вихідні дані перевантажених операцій порівняння*



*Рисунок 6. Вихідні дані табуляції обчислювального параметру.*



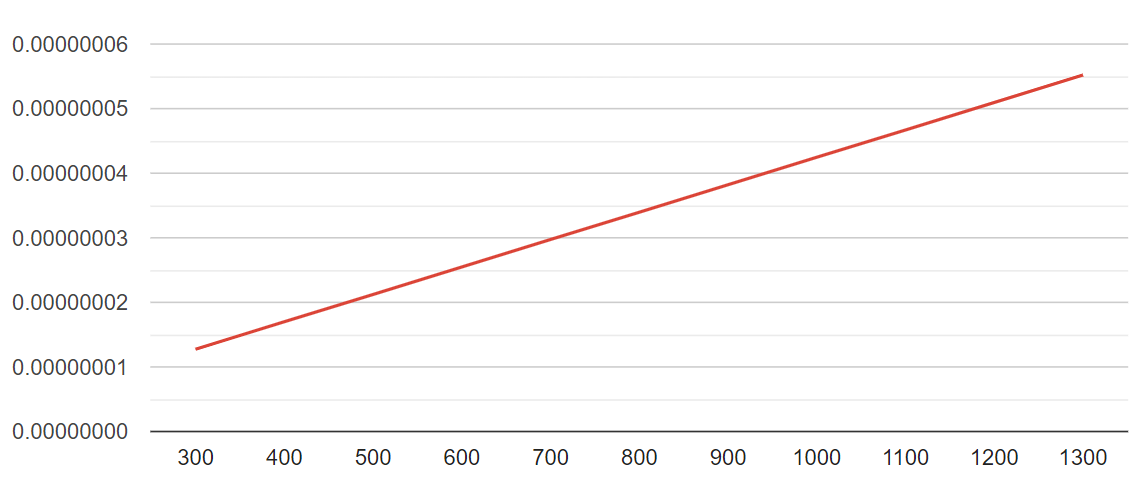
*Рисунок 7.1 Консоль.*



*Рисунок 7.2 Консоль.*

# **ГРАФІК ТАБУЛЮВАННЯ ОБЧИСЛЮВАЛЬНОГО ПАРАМЕТРУ**

*Рисунок 1. Графік табулювання обчислювального параметру.*



# **ВИСНОВКИ**

В курсовій роботі ми розглянули основні принципи, які лежать в основі технології об'єктно-орієнтованого програмування, такі як інкапсуляція, наслідування та поліморфізм. Ми також отримали базові знання про те, як описувати класи та створювати об'єкти як екземпляри цих класів. Під час виконання курсової роботи ми створили програму, яка розраховує середню довжину вільного перебігу молекули. Для цього ми описали абстрактний клас та передбачили такі віртуальні функції: введення даних із файлу, обчислення заданого параметра, виведення у файл результату і вхідних даних, геттер для значення обчислюваного параметру. Ми описали похідний клас та додали до нього елементи даних відповідно до варіанта. Описали пустий конструктор, з ініціалізацією і копіювання. Дані ми зчитували з файлів, а результати розрахунків виводили в інші файли. Також ми протабулювали обчислювальний параметр, а результати обчислень вивели у файл та побудували графік одержаної залежності на інтернет-ресурсі.

**СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ**

1. Програмування лінійних алгоритмів (мова С/С++) URL:<https://naurok.com.ua/programuvannya-liniynih-algoritmiv-mova-s-s-252911.html> (дата звернення: 24.12.2022).
2. Огляд і основи мови програмування C++ URL: <http://www.znannya.org/?view=Cpp_basics> (дата звернення: 24.12.2022).
3. Уроки програмування на С++ URL: <https://acode.com.ua/uroki-po-cpp/#toc-0> (дата звернення: 24.12.2022).
4. Лабораторний практикум URL: <https://mishchyk.files.wordpress.com/2018/03/d0bbd0b0d0b1d0bed180d0b0d182d0bed180d0bdd0b8d0b9_d0bfd180d0b0d0bad182d0b8d0bad183d0bc.pdf> (дата звернення: 24.12.2022).
5. Интерфейсы и абстрактные классы в С++ / ravesli. Ravesli URL: <https://ravesli.com/urok-168-chistye-virtualnye-funktsii-interfejsy-i-abstraktnye-klassy/> (дата звернення: 24.12.2022).
6. Урок #21 - Дружественные классы. itProger - Сообщество программистов URL: <https://itproger.com/course/cpp/21> (дата звернення: : 24.12.2022).
7. Графіки та діаграми URL:  <https://yequalx.com/ru/chart/line/%D0%97%D0%B0%D0%B3%D0%BE%D0%BB%D0%BE%D0%B2%D0%BE%D0%BA,%D0%97%D0%B0%D0%B3%D0%BE%D0%BB%D0%BE%D0%B2%D0%BE%D0%BA;300,1.2739e-8;400,1.6985e-8;500,2.1231e-8;600,2.5477e-8;700,2.9723e-8;800,3.3969e-8;900,3.8216e-8;1000,4.2462e-8;1100,4.6708e-8;1200,5.0954e-8;1300,5.52e-8#w:900;h:400;c:db4437> (дата звернення: : 24.12.2022).

**ДОДАТОК(код програми)**

#include <iostream>

#include<cmath>

#include <iomanip>

#include<Windows.h>

#include<fstream>

#define pi 3,14

using namespace std;

//абстрактний клас

class Object {

protected:

//oбчислювальний параметр

double l;

public:

//чисті віртуальні функції

virtual void enter\_file(const char\* S) = 0;

virtual void calc\_l() = 0;

virtual void output\_file(const char\* S) = 0;

virtual const double output\_l() = 0;

virtual ~Object() {}

};

//похідний клас

class Mileage\_molecule : public Object {

double N = 0;

double d = 0;

double p = 0;

double Tg = 0;

public:

//конструктори

Mileage\_molecule() {};

Mileage\_molecule(double d, double p, double Tg);

Mileage\_molecule(const Mileage\_molecule& Mm);

void enter\_file(const char\* S);

void calc\_l();

const double output\_l();

double calc\_N();

void output\_file(const char\* S);

//перевантажені операції

void operator=(Mileage\_molecule const& Mm);

friend bool operator==(Mileage\_molecule const& A, Mileage\_molecule const& B);

friend bool operator!=(Mileage\_molecule const& A, Mileage\_molecule const& B);

friend bool operator<(Mileage\_molecule const& A, Mileage\_molecule const& B);

friend bool operator<=(Mileage\_molecule const& A, Mileage\_molecule const& B);

friend bool operator>(Mileage\_molecule const& A, Mileage\_molecule const& B);

friend bool operator>=(Mileage\_molecule const& A, Mileage\_molecule const& B);

~Mileage\_molecule() {}

};

int main(){

//для використання української мови

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

cout << "No I" << endl;

cout << "Покажчик на батьківський клас. Пустий конструктор. Введення з файлу" << endl;

Object\* Obj\_1 = new Mileage\_molecule();

Obj\_1->enter\_file("Obj\_1.txt");

Obj\_1->calc\_l();

Obj\_1->output\_file("Obj\_1\_res.txt");

cout << endl;

cout << "No II" << endl;

cout << "Покажчик на батьківський клас. Конструктор з ініціалізацією:" << endl << "N = 5 d = 3, 656 \* 10 ^ -10 p = 1, 5 Tg = 500" << endl;

Object\* Obj\_2 = new Mileage\_molecule(3.656 \* pow(10, -10), 1.5, 500);

Obj\_2->calc\_l();

cout << "l = " << Obj\_2->output\_l() << endl;

cout << endl;

ofstream ofs("comparison\_res.txt");

cout << "No III" << endl;

cout << "Пустий конструктор. Введення з файлу " << endl;

Mileage\_molecule Obj\_3;

Obj\_3.enter\_file("Obj\_3.txt");

Obj\_3.calc\_l();

Obj\_3.output\_file("Obj\_3\_res.txt");

ofs << "l = " << Obj\_3.output\_l() << endl;

cout << endl;

cout << "No IV" << endl;

cout << "Конструктор з ініціалізацією" << endl << "N = 1 d = 3,64 \* 10^-10 p = 1,2 Tg = 300 " << endl;

Mileage\_molecule Obj\_4(3.64 \* pow(10, -10), 1.2, 300);

Obj\_4.calc\_l();

cout << "l = " << Obj\_4.output\_l() << endl;

ofs << "l = " << Obj\_4.output\_l() << endl;

cout << endl;

cout << "No V" << endl;

cout << "Конструктор копії" << endl << "No V = No IV" << endl;

Mileage\_molecule Obj\_5(Obj\_4);

Obj\_5.calc\_l();

cout << "l = " << Obj\_5.output\_l() << endl;

ofs << "l = " << Obj\_5.output\_l() << endl;

cout << endl << endl;

//Порівняння

ofs << "Порівняння" << endl;

ofs << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

cout << "Порівняння" << endl;

cout << "Obj\_3 == Obj\_4 ?" << endl;

if (Obj\_3 == Obj\_4) {

cout << "ТАК" << endl;

ofs << "Obj\_3 == Obj\_4" << endl;

}

else {

cout << "НІ" << endl;

ofs << "Obj\_3 != Obj\_4" << endl;

}

cout << "Obj\_4 == Obj\_5 ?" << endl;

if (Obj\_4 == Obj\_5) {

cout << "ТАК" << endl;

ofs << "Obj\_4 == Obj\_5" << endl;

}

else {

cout << "НІ" << endl;

ofs << "Obj\_4 != Obj\_5" << endl;

}

cout << "Obj\_3 == Obj\_5 ?" << endl;

if (Obj\_3 == Obj\_5) {

cout << "ТАК" << endl;

ofs << "Obj\_3 == Obj\_5" << endl;

}

else {

cout << "НІ" << endl;

ofs << "Obj\_3 != Obj\_5" << endl;

}

ofs << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

cout << "Obj\_3 != Obj\_4 ?" << endl;

if (Obj\_3 != Obj\_4) {

cout << "ТАК" << endl;

ofs << "Obj\_3 != Obj\_4" << endl;

}

else {

cout << "НІ" << endl;

ofs << "Obj\_3 == Obj\_4" << endl;

}

cout << "Obj\_4 != Obj\_5 ?" << endl;

if (Obj\_4 != Obj\_5) {

cout << "ТАК" << endl;

ofs << "Obj\_4 != Obj\_5" << endl;

}

else {

cout << "НІ" << endl;

ofs << "Obj\_4 == Obj\_5" << endl;

}

cout << "Obj\_3 != Obj\_5 ?" << endl;

if (Obj\_3 != Obj\_5) {

cout << "ТАК" << endl;

ofs << "Obj\_3 != Obj\_5" << endl;

}

else {

cout << "НІ" << endl;

ofs << "Obj\_3 == Obj\_5" << endl;

}

ofs << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

cout << "Obj\_3 > Obj\_4 ?" << endl;

if (Obj\_3 > Obj\_4) {

cout << "ТАК" << endl;

ofs << "Obj\_3 > Obj\_4" << endl;

}

else {

cout << "НІ" << endl;

ofs << "Obj\_3 <= Obj\_4" << endl;

}

cout << "Obj\_4 >= Obj\_5 ?" << endl;

if (Obj\_4 >= Obj\_5) {

cout << "ТАК" << endl;

ofs << "Obj\_4 >= Obj\_5" << endl;

}

else {

cout << "НІ" << endl;

ofs << "Obj\_4 < Obj\_5" << endl;

}

cout << "Obj\_3 < Obj\_5 ?" << endl;

if (Obj\_3 < Obj\_5) {

cout << "ТАК" << endl;

ofs << "Obj\_3 < Obj\_5" << endl;

}

else {

cout << "НІ" << endl;

ofs << "Obj\_3 >= Obj\_5" << endl;

}

cout << "Obj\_3 <= Obj\_4 ?" << endl;

if (Obj\_3 <= Obj\_4) {

cout << "ТАК" << endl;

ofs << "Obj\_3 <= Obj\_4" << endl;

}

else {

cout << "НІ" << endl;

ofs << "Obj\_3 > Obj\_4" << endl;

}

cout << endl << endl;

//табуляція

ofstream ofs1("tabulate\_res.txt");

ofs1 << "Табуляція за параметром Tg" << endl;

ofs1 << "Tg \t l" << endl;

cout << "Табуляція за параметром Tg" << endl;

cout << "Tg \t l" << endl;

for (int delta = 300; delta <= 1300; delta += 100) {

Mileage\_molecule l\_tabulate(3.6 \* pow(10, -10), 1.25, delta);

l\_tabulate.calc\_l();

ofs1 << delta << "\t" << setprecision(5) << l\_tabulate.output\_l() << endl;

cout << delta << "\t" << setprecision(5) << l\_tabulate.output\_l() << endl;

}

cout << endl;

cout << "Результати файлi 'tabulate\_res.txt' " << endl;

ofs.close();

ofs1.close();

delete Obj\_1;

delete Obj\_2;

system("pause");

return 0;

}

Mileage\_molecule::Mileage\_molecule(double d, double p, double Tg) {

this->d = d;

this->p = p;

this->Tg = Tg;

}

Mileage\_molecule::Mileage\_molecule(const Mileage\_molecule& Mm) {

N = Mm.N;

d = Mm.d;

p = Mm.p;

Tg = Mm.Tg;

}

void Mileage\_molecule::enter\_file(const char\* S) {

ifstream ifs(S);

if (!ifs)

{

cout << "Файл не відкритий " << endl;

exit(1);

}

cout << "Файл відкритий " << endl;

ifs >> N >> d >> p >> Tg;

//cout << N << endl << d << endl << p << endl << Tg << endl;

}

void Mileage\_molecule::calc\_l() {

N = calc\_N();

if (d == 0 || N == 0) {

cout << "Помилка" << endl;

exit(2);

}

else {

l = 1.0 / (pi \* d \* d \* N \* sqrt(2));

}

}

double Mileage\_molecule::calc\_N() {

if (Tg == 0) {

cout << "Помилка" << endl;

exit(3);

}

else {

N = (6.02497 \* pow(10, 23) \* p) / (Tg \* 8.2056 \* pow(10, -5));

}

}

void Mileage\_molecule::output\_file(const char\* S) {

ofstream ofs(S);

ofs << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

ofs << "N = " << N << endl;

ofs << "d = " << d << endl;

ofs << "p = " << p << endl;

ofs << "Tg = " << Tg << endl;

ofs << "l = " << l << endl;

ofs << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

cout << "N = " << N << endl;

cout << "d = " << d << endl;

cout << "p = " << p << endl;

cout << "Tg = " << Tg << endl;

cout << "l = " << l << endl;

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

cout << "Результати y файлі " << S << endl;

}

const double Mileage\_molecule::output\_l() {

return l;

}

void Mileage\_molecule::operator=(Mileage\_molecule const& Mm) {

N = Mm.N;

d = Mm.d;

p = Mm.p;

Tg = Mm.Tg;

}

bool operator==(Mileage\_molecule const& A, Mileage\_molecule const& B) {

return A.l == B.l ? true : false;

}

bool operator!=(Mileage\_molecule const& A, Mileage\_molecule const& B) {

return A.l != B.l ? true : false;

}

bool operator<(Mileage\_molecule const& A, Mileage\_molecule const& B) {

return A.l < B.l ? true : false;

}

bool operator<=(Mileage\_molecule const& A, Mileage\_molecule const& B) {

return A.l <= B.l ? true : false;

}

bool operator>(Mileage\_molecule const& A, Mileage\_molecule const& B) {

return A.l > B.l ? true : false;

}

bool operator>=(Mileage\_molecule const& A, Mileage\_molecule const& B) {

return A.l >= B.l ? true : false;

}