**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**ЧЕРНІВЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**

**КУРСОВА РОБОТА**

на тему:

**«Розробити структуру об'єкта для**

**подання й розрахунку обсягу й площі**

**поверхні тетраедра»**

Виконав:

студент ІІІ-го курсу 31КІ групи

Бурла Г.Г.

(Прізвище та ініціали)(підпис)

Науковий керівник:

(Прізвище та ініціали)(підпис)

Оцінка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Чернівці – 2023

**ЗМІСТ**

**ВСТУП** ......................................................................................................................... 3

#### РОЗДІЛ І. ЗАГАЛЬНА ХАРАКТЕРИСТИКА ТЕТРАЕДРА ........................... 6

1.1. Аналіз тетраедра .................................................................................... 6

1.2. Огляд і аналіз формул площі і об’єму для знаходження тетраедра 14

**РОЗДІЛ ІІ. КОНСТРУКТОРСЬКИЙ РОЗДІЛ** .................................................. 17

2.1.Особливості ООП. ................................................................................. 17

1. 2.Вибір мови ............................................................................................. 19
   1. Вибір середовища програмування ..................................................... 22

**РОЗДІЛ ІІІ. ПРОГРАМА ТА ЇЇ ОСОБЛИВОСТІ** ............................................ 24

1. 1.Аналіз програми .................................................................................... 24

3.2. Створення програми ............................................................................ 26

**ВИСНОВКИ ..............................................................................................................** 30

**СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ .....................................................** 31

**ДОДАТКИ ................................................................................................................. 32**

**ВСТУП**

**Об'єктно - орієнтоване програмування (ООП)** – це модель програмування яка базується на стверджені того, що програма це сукупність об’єктів які взаємодіють між собою. Кожен об’єкт в цій моделі є незалежним, і він здатний отримувати, обробляти дані та відправляти ці дані іншим об’єктам. В ООП використано моделі успадкування, модульності, поліморфізму та інкапсуляції.

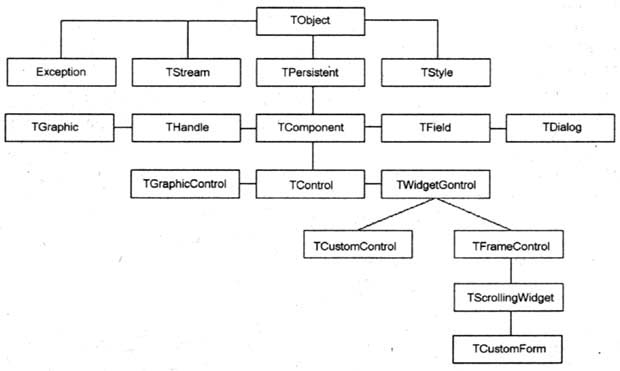
Основним поняттям ООП є об'єкт. **Об'єкт** можна визначити як певну сукупність даних(характеристик об'єкта) та методів роботи з ними. Для класифікації об'єктів у ООП використовують класи. **Клас** служить зразком для створення обєкту, тобто об'єкт є нічим іншим, ніж копією класу.

Кожен об'єкт має процедури і функції(те що він уміє виконувати, наприклад,завантажувати файл, відображати картинку і т.д.), які служать для роботи з даними об'єкта. Ці процедури і функції називаються **методами**.

Існування ООП можливе завдяки трьом основним парадигмам **на яких базується саме ООП:**

- **Інкапсуляція**. Також відома як приховування даних. Зміст інкапсуляції полягає у приховуванні від зовнішнього користувача деталей реалізації об'єкта, замість цього надаючи інтерфейс взаємодії з ним.

- **Успадкування.** Це означає, що об'єкти (класи) можуть переймати деякі властивості у своїх прабатьків. Як? Це залежить від тієї мови, на якому пишеться програма. Однак у будь-якому випадку картина та ж: це призводить до повторного використання вже написаного одного разу коду. Підкласи успадковують атрибути та поведінку своїх батьківських класів, і можуть мати нові власні атрибути. Тобто утворюється ієрархія з класів, де від основного класу(так званого, предка) походять усі інші класи. Приклад такого розгалуженого "дерева" зображено на схемі.

[](http://programming.in.ua/images/stories/articles/basis_progm/082.jpg)

[**C**](http://www.znannya.org/?view=concept:212)**++** — універсальна мова програмування високого рівня з підтримкою декількох парадигм програмування. Зокрема: об’єктно-орієнтованої та процедурної. Розроблена Б’ярном Страуструпом (англ. Bjarne Stroustrup) в AT&T Bell Laboratories (Мюррей-Хілл, Нью-Джерсі) у 1979 році та названа «С з класами». Страуструп перейменував мову у [C](http://www.znannya.org/?view=concept:212)++ у 1983 р. Базується на мові Сі. Визначена стандартом ISO/IEC 14882:2003.

У 1990-х роках С++ стала однією з найуживаніших мов програмування загального призначення.

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.

Нововведеннями С++ порівняно з С є:

підтримка об'єктно-орієнтованого програмування через класи;

підтримка узагальненого програмування через шаблони;

доповнення до стандартної бібліотеки;

додаткові типи даних;

обробка винятків;

простори імен;

вбудовані функції;

перевантаження операторів;

перевантаження імен функцій;

посилання і оператори управління вільно розподіленою пам'яттю.

У 1998 році ратифіковано міжнародний стандарт мови С++: ISO/IEC 14882 «Standard for the [C](http://www.znannya.org/?view=concept:212)++ Programming Language». Поточна версія цього стандарту — ISO/IEC 14882:2003.

- **Поліморфізм** означає залежність поведінки від класу, в якому ця поведінка викликається, тобто, два або більше класів можуть реагувати по різному на однакові повідомлення. Це спричинене зміною в одного з класів якогось методу(процедури, функції), шляхом запису іншого алгоритму. Як приклад, деяка комп'ютерна програма при натисканні клавіші Esc завершить роботу, інша ж програма після натискання кнопки Esc тільки відкриє меню даної програми.

В C++ є і динамічний поліморфізм, коли функція, що викликається, визначається під час виконання. Для цього функції-члени повинні бути віртуальними. 5 Чисто віртуальною функцією називається функція-член, яка не визначена в базовому класі, а тільки в нащадках: Об’єктом курсового проекту на тему “Розробити структуру об'єкта для подання й розрахунку об’єму й площі поверхні тетраедра” є формули знаходження площі і об’єму тетраедра. Предметом курсового проекту на тему “Розробити структуру об'єкта для подання й розрахунку об’єму й площі поверхні тетраедра ” є дослідження, вивчення та реалізація алгоритму по розрахунці площі і об’єму тетраедра. Метою написання курсового проекту на тему “Розробити структуру об'єкта для подання й розрахунку об’єму й площі поверхні тетраедра ” є створення програмного продукту, який реалізує знаходження площі і об’єму тетраедра. В першому розділі даного курсового проекту докладно описано про тетраедр, а також про алгоритм знаходження його площі і об’єму. В другому розділі даного курсового проекту докладно описано мову програмування С++, і про її можливості, які використовувались під час створення програми. В третьому розділі даного курсового проекту описано і роз’яснено код програми по заданому алгоритму

**РОЗДІЛ І. ЗАГАЛЬНА ХАРАКТЕРИСТИКА ТЕТРАЕДРА**

* 1. **Аналіз тетраедра**

Тетраедр - це багатогранник з чотирма гранями, шістьма ребрами та чотирма вершинами. Це тривимірна фігура, утворена кількома багатокутниками, які в даному випадку є трикутниками.

Тетраедр характеризується тим, що є найпростішим із багатогранників і єдиним, що має менше п’яти сторін.

Варто згадати, що тетраедр - це піраміда з трикутною основою.

**Елементи тетраедра**

* **Обличчя:** Вони є сторонами тетраедра, які, як ми вже згадували, є трикутниками (ABC, ADC, ADB і BDC.
* **Краї:** Це об'єднання двох граней: AB, AC, AD, BC, CD і DB.
* **Вершини:** Це ті точки, де стикаються ребра: A, B, C і D.
* **Двогранний** кут: Він утворений об'єднанням двох граней.
* **Кут** багатогранника: Він складається з сторін, які збігаються в одній вершині.

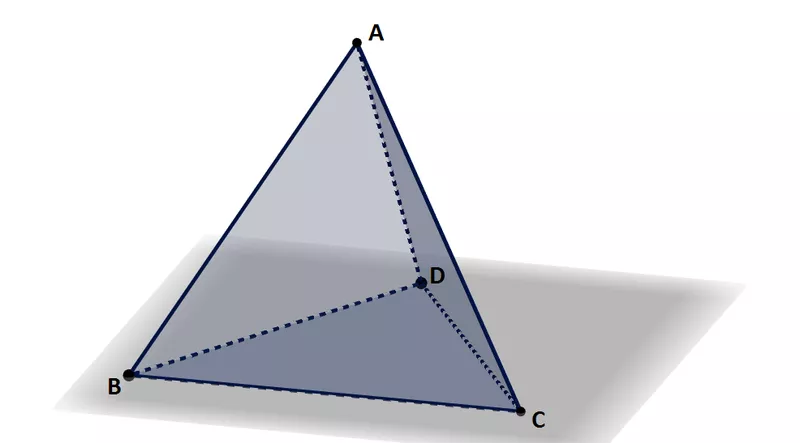


Рис 1.1 Тетраедр

**Площа та об’єм тетраедра**

Щоб знати характеристики тетраедра, ми можемо розрахувати:

* Площа: Довелося б додати площу чотирьох трикутників, що складають багатогранник. У цьому сенсі ми повинні пам’ятати, що площа трикутника обчислюється множенням основи на висоту і діленням на 2 (A = bxh / 2)
* Об'єм: Це було б розраховано за такою формулою

У формулі b - будь-яка грань багатогранника, а h - висота або відрізок, що з'єднує b з протилежною вершиною. Крім того, висота перпендикулярна до основи (вони утворюють прямий кут або розмір 90º).

**Властивості тетраедра**

* Паралельні площини, які проходять через два перехресних ребра, утворюють описаний паралелепіпед.
* Відмітною властивістю тетраедра є те, що медіани і бімедіани фігури зустрічаються в одній точці. Важливо, що остання ділить медіани у відношенні 3: 1, а бімедіани – навпіл.
* Площина розділяє тетраедр на дві рівні за обсягом частини, якщо проходить через середину двох перехресних ребер.

**Описаний паралелепіпед** Паралельні площини, що проходять через три пари перехресних ребер тетраедра, визначають описаний близько тетраедра паралелепіпед.

**Розподіл тетраедра на дві рівні частини** Площина, що проходить через середини двох перехресних ребер тетраедра, ділить його на дві рівні за обсягом частини

**Примітка:** гіперплоскость, що проходить через центри тяжкості протилежних елементів симплекса, ділить його на рівні по гіпероб'ёму частини.

**Бімедіани** Бімедіани тетраедра перетинаються в тій же самій точці, що і медіани тетраедра.

**Бімедіанами тетраедра називають** відрізки, що з'єднують середини його перехресних ребер (що не мають спільних вершин).

**Лемма про тризуб** Центри сфер, які проходять через три вершини і інцентр, лежать на сфері, центр якої збігається з центром описаної сфери. Также це твердження вірне і для зовнішніх інцентрів.

**Ортоцентр** Площини, які проходять через середину ребра і перпендикулярні протилежному ребру, перетинаються в одній точці (ортоцентр)

**Види тетраєдра**

Видове різноманіття фігури досить широко. Тетраедр може бути:

-правильним, тобто в основі рівносторонній трикутник;

-рівногранним

-ортоцентричним, коли висоти мають спільну точку перетину;

-прямокутним, якщо плоскі кути при вершині нормальні;

-пропорційним, все бі висоти рівні;

-каркасних, якщо присутній сфера, яка стосується ребер;

-інцентричним, тобто відрізки, опущені з вершини в центр вписаного кола протилежній грані, мають спільну точку перетину; цю точку називають центром тяжіння тетраедра.

**Рівногранний тетраедр**

Всі грані його представляють собою рівні між собою трикутники.

Розгортку рівногранного тетраедра є трикутник, розділений трьома середніми лініями на чотири рівних трикутника. У рівногранного тетраедра основи висот, середини висот і точки перетину висот граней лежать на поверхні однієї сфери (сфери 12 точок) .

Властивості рівногранного тетраедра:

• Всі його грані рівні (конгруентні).

• Перехресні ребра попарно рівні.

• Тригранні кути рівні.

• Протилежні двогранні кути рівні. 8

• Два плоских кута, що спираються на одне ребро, рівні.

• Сума плоских кутів при кожній вершині дорівнює 180 °.

• Розгортка тетраедра - трикутник або паралелограм.

• Описаний паралелепіпед прямокутний.

• Тетраедр має три осі симетрії.

• Загальні перпендикуляри перехресних ребер попарно перпендикулярні.

• Середні лінії попарно перпендикулярні.

• Периметри граней рівні.

• Площі граней рівні.

• Висоти тетраедра рівні.

• Відрізки, що з'єднують вершини з центрами тяжіння протилежних граней, рівні.

• Радіуси описаних близько граней кіл рівні.

• Центр тяжкості тетраедра збігається з центром описаної сфери.

• Центр тяжкості збігається з центром вписаного сфери.

• Центр описаної сфери збігається з центром вписаного.

• Вписана сфера стосується граней в центрах описаних близько цих граней кіл.

• Сума зовнішніх одиничних нормалей (одиничних векторів, перпендикулярних до граней), дорівнює нулю.

• Сума всіх двогранних кутів дорівнює нулю.

• Центри позавписаних сфер лежать на описаної сфері.

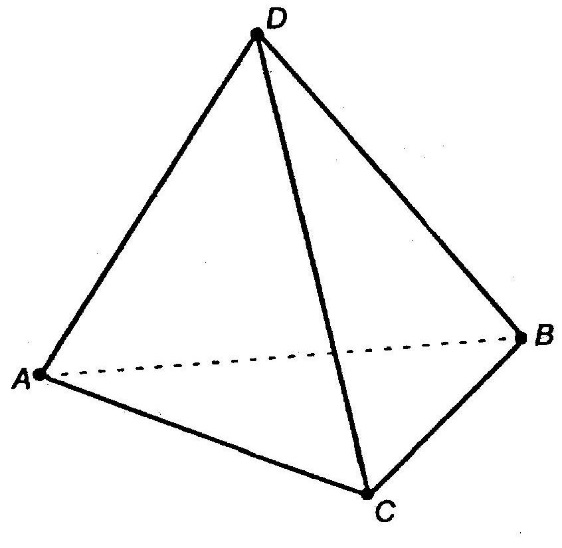


Рис 1.2 Рівногранний тетраедр

**Ортоцентричний тетраедр**

Всі висоти, опущені з вершин на протилежні грані, перетинаються в одній точці.

• Висоти тетраедра перетинаються в одній точці.

• Підстави висот тетраедра є ортоцентром граней.

• Кожні два протилежних ребра тетраедра перпендикулярні.

• Суми квадратів протилежних ребер тетраедра рівні.

• Відрізки, що з'єднують середини протилежних ребер тетраедра, рівні. 9

• Створенні косинуси протилежних двогранних кутів рівні.

• Сума квадратів площ граней вчетверо менше суми квадратів творів протилежних ребер.

• У ортоцентричного тетраедра окружності 9 точок (окружності Ейлера) кожної грані належать одній сфері (сфері 24 точок).

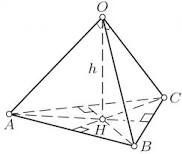


Рис 1.3 Ортоцентричний тетраедр

**Прямокутний тетраедр**

**Прямокутний тетраедр** — це [чотиригранник](https://uk.wikipedia.org/wiki/%D0%A7%D0%BE%D1%82%D0%B8%D1%80%D0%B8%D0%B3%D1%80%D0%B0%D0%BD%D0%BD%D0%B8%D0%BA) у якого всі ребра, прилеглі до однієї з вершин, [перпендикулярні](https://uk.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%BF%D0%B5%D0%BD%D0%B4%D0%B8%D0%BA%D1%83%D0%BB%D1%8F%D1%80%D0%BD%D1%96%D1%81%D1%82%D1%8C) між собою.

У прямокутному тетраедрі завжди три прилеглі грані будуть [прямокутними трикутниками](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D1%8F%D0%BC%D0%BE%D0%BA%D1%83%D1%82%D0%BD%D0%B8%D0%B9_%D1%82%D1%80%D0%B8%D0%BA%D1%83%D1%82%D0%BD%D0%B8%D0%BA), а остання грань буде [довільним трикутником](https://uk.wikipedia.org/wiki/%D0%A2%D1%80%D0%B8%D0%BA%D1%83%D1%82%D0%BD%D0%B8%D0%BA) і називається *базою*.

Прямокутний тетраедр виходить відсіканням тетраедра площиною від прямокутного паралелепіпеда.

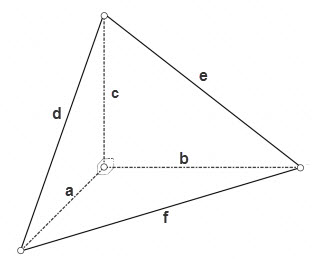


Рис 1.4 Прямокутний тетраедр

**Каркасний тетраедр**

Каркасним називається тетраедр, для якого існує сфера, що стосується всіх шести ребер тетраедра. Не всякий каркасний- тетраедр.

Наприклад, легко зрозуміти, що не можна побудувати сферу, що стосується всіх ребер рівногранного тетраедра, якщо описаний паралелепіпед "довгий".

Це тетраедр, який відповідає будь-якому з наступних умов:

• існує сфера, яка торкається всіх ребер,

• суми довжин перехресних ребер рівні,

• суми двогранні кутів при протилежних ребрах рівні,

• окружності, вписані в межі, попарно торкаються,

• всі чотирикутники, що виходять на розгортці тетраедра, - описані,

• перпендикуляри, восставленний до граней з центрів вписаних в них кіл, перетинаються в одній точці.

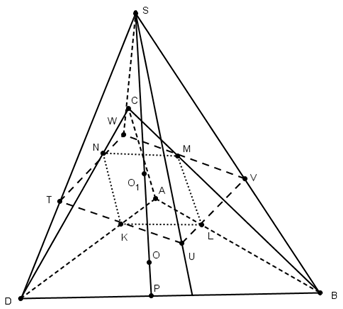


Рис 1.5 Каркасний тетраедр

**Пропорційний тетраедр**

У цього типу бівисоти рівні. Властивості пропорційного тетраедра:

• Бівисоти рівні. Бівисотамі тетраедра називають загальні перпендикуляри до двох схрещуючих його ребрах (ребрах, які не мають спільних вершин).

• Проекція тетраедра на площину, перпендикулярну будь-якій бімедіані, є ромб.

Бімедіанами тетраедра називають відрізки, що з'єднують середини його перехресних ребер (що не мають спільних вершин).

• Грані описаного паралелепіпеда рівновеликі.

• Виконуються співвідношення:

4а2а1 2 -(b 2 +b1 2 – c 2 –c1 2 ) 2= 4b 2b1 2 – (c 2+c1 2 -a 2 -a1 2 ) 2=4c 2 c1 2 – (a 2+a1 2 –b 2 - (b1) 2 ) 2 - де а і а1, b і b1,c і c1 - довжини протилежних ребер.

• Для кожної пари протилежних ребер тетраедра площини, проведені через одне

з них і середину другого, перпендикулярні.

• В описаний паралелепіпед пропорційного тетраедра можна вписати сферу.

**Інцентричнийй тетраедр**

У цього типу відрізки, що з'єднують вершини тетраедра з центрами кіл, вписаних в протилежні грані, перетинаються в одній точці. Властивості інцентричного тетраедра:

• Відрізки, що з'єднують центри тяжкості граней тетраедра з протилежними вершинами (медіани тетраедра), завжди перетинаються в одній точці. Ця точка - центр тяжкості тетраедра.

• Зауваження. Якщо в останньому умови замінити центри тяжкості граней на ортоцентром граней, то воно перетвориться в нове визначення ортоцентричного тетраедра. Якщо ж замінити їх на центри вписаних в межі кіл, які називаються іноді інцентрами, ми отримаємо визначення нового класу тетраедрів - інцентрічених.

• Відрізки, що з'єднують вершини тетраедра з центрами кіл, вписаних в протилежні грані, перетинаються в одній точці.

• бісектриси кутів двох граней, проведеним до загального ребру цих граней, мають загальну підставу.

• Твори довжин протилежних ребер рівні.

• Трикутник, утворений другими точками перетину трьох ребер, що виходять з однієї вершини, з будь-якою сферою, що проходить через три кінця цих ребер, є рівностороннім.

**Правильний тетраедр**

Це рівногранний тетраедр, у якого всі грані - правильні трикутники. Є одним з п'яти тіл Платона.

Властивості правильного тетраедра:

• Всі ребра тетраедра рівні між собою,

• Всі грані тетраедра рівні між собою, 11

• Периметри і площі всіх граней рівні між собою.

• Правильний тетраедр є одночасно ортоцентричним, каркасним, рівногранним, інцентричним і пропорційним.

• Тетраедр є правильним, якщо він належить до двох будь-яких видів тетраедрів з перерахованих: ортоцентричний, каркасний, рівногранний, інцентричний і пропорційний.

• Тетраедр є правильним, якщо він є рівногранним і належить до одного з наступних видів тетраедрів: ортоцентричний, каркасний, інцентричний і пропорційний.

• У правильний тетраедр можна вписати октаедр, притому чотири (з восьми) межі октаедра будуть суміщені з чотирма гранями тетраедра, всі шість вершин октаедра будуть суміщені з центрами шести ребер тетраедра.

• Правильний тетраедр складається з одного вписаного октаедра (в центрі) і чотирьох тетраедрів (по вершинах), причому ребра цих тетраедрів і октаедра вдвічі менше ребер правильного тетраедра.

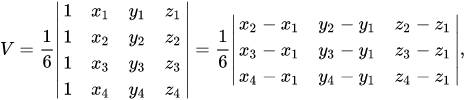
• Перехресні ребра правильного тетраедра взаємно перпендикулярні.

**1.2. Огляд і аналіз формул площі і об’єму для знаходження тетраедра**

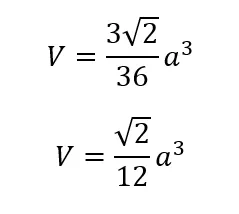
Об’єм тетраедра

Об'єм тетраедра (з урахуванням знака), вершини якого знаходяться в точках

r1(x1,y1,z1), r2(x2,y2,z2), r3(x3,y3,z3), r4(x4,y4,z4), дорівнює



Формула об’єму правильного тетраедра:

****

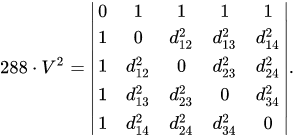
**Або**



Де S – площа будь-якої грані�, а H - це висота, �,що ,щ що опустили на цю грань,

V – об’єм тетраедра.

* Обєм тетраєдра через довжину ребер визначається за допомогою визначника Келі – Менгера



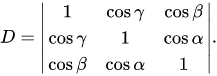
* Ця формула має плоский аналог для площі трикутника у вигляді варіанту формули Герона через аналоговій визначник
* Об’єм тетраедра через довжини двох протилежних ребер a і b , як перехресних ліній, які віддалені на відстань h одна від одної і утворюють один з одним кут φ, Знаходиться за формулою:



* Об’єм тетраедра через довжини трьох його ребер a , b і c , що виходять з однієї вершини і утворюють між собою попарно відповідно плоскі кути , Знаходиться за формулою



Де



* Аналогом для площині останньої формули є формула площі трикутника через довжини двох його сторін a і b , що виходять з однієї вершини і утворюють між собою кут

**Площа тетраедра**

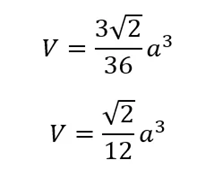
Формула площі правильного тетраедра:



де



Отже, для вирахування площі і об’єму тетраедра, я буду використовувати формули площі і об’єму правильного тетраедра:



**РОЗДІЛ ІІ. КОНСТРУКТОРСЬКИЙ РОЗДІЛ**

**2.1.Особливості ООП**

**Об'є́ктно-орієнто́ване програмува́ння** (**ООП**, іноді *об'єктно-зорієнто́ване програмува́ння*; [англ.](https://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D1%96%D0%B9%D1%81%D1%8C%D0%BA%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0) *Object-oriented programming, OOP*) — одна з [парадигм](https://uk.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%B4%D0%B8%D0%B3%D0%BC%D0%B8_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) [програмування](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F), яка розглядає програму як множину [«об'єктів»](https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%27%D1%94%D0%BA%D1%82_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)), що взаємодіють між собою.  Основу ООП складають чотири основні концепції: [інкапсуляція](https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D0%BA%D0%B0%D0%BF%D1%81%D1%83%D0%BB%D1%8F%D1%86%D1%96%D1%8F), [успадкування](https://uk.wikipedia.org/wiki/%D0%A3%D1%81%D0%BF%D0%B0%D0%B4%D0%BA%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)), [поліморфізм](https://uk.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D1%96%D0%BC%D0%BE%D1%80%D1%84%D1%96%D0%B7%D0%BC_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)) та абстракція Однією з переваг ООП є краща [модульність](https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D1%83%D0%BB%D1%8C_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)) програмного забезпечення (тисячу функцій процедурної мови, в ООП можна замінити кількома десятками класів із своїми методами). Попри те, що ця парадигма з'явилась в 1960-х роках, вона не мала широкого застосування до [1990-х](https://uk.wikipedia.org/wiki/1990-%D1%82%D1%96), коли розвиток комп'ютерів та комп'ютерних мереж дав змогу писати надзвичайно об'ємне і складне програмне забезпечення, що змусило переглянути підходи до написання програм. Сьогодні багато [мов програмування](https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) або підтримують ООП ([PHP](https://uk.wikipedia.org/wiki/PHP), [Lua](https://uk.wikipedia.org/wiki/Lua)) або ж є цілком [об'єкто-орієнтованими](https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%27%D1%94%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D1%96%D1%94%D0%BD%D1%82%D0%BE%D0%B2%D0%B0%D0%BD%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) (зокрема, [Java](https://uk.wikipedia.org/wiki/Java_(%D0%BC%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)), [C#](https://uk.wikipedia.org/wiki/C_Sharp_(%D0%BC%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)), [C++](https://uk.wikipedia.org/wiki/C%2B%2B), [Python](https://uk.wikipedia.org/wiki/Python), [Ruby](https://uk.wikipedia.org/wiki/Ruby) і [Objective-C](https://uk.wikipedia.org/wiki/Objective-C), [ActionScript 3](https://uk.wikipedia.org/wiki/ActionScript), [Swift](https://uk.wikipedia.org/wiki/Swift_(%D0%BC%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)), [Vala](https://uk.wikipedia.org/wiki/Vala)).

Об'єктно-орієнтоване програмування сягає своїм корінням до створення мови програмування [Симула](https://uk.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BC%D1%83%D0%BB%D0%B0) в 1960-х роках, одночасно з посиленням дискусій про [кризу програмного забезпечення](https://uk.wikipedia.org/w/index.php?title=%D0%9A%D1%80%D0%B8%D0%B7%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F&action=edit&redlink=1). Через ускладнення апаратного та програмного забезпечення було дуже важко зберегти якість програм. Об'єкто-орієнтоване програмування частково розв'язує цю проблему шляхом наголошення на модульності програми.

На відміну від традиційних поглядів, коли програму розглядали як набір [підпрограм](https://uk.wikipedia.org/wiki/%D0%9F%D1%96%D0%B4%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B0), або як перелік [інструкцій](https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%86%D1%96%D1%8F) комп'ютеру, ООП-програми можна вважати сукупністю об'єктів. Відповідно до парадигми об'єктно-орієнтованого програмування, кожен об'єкт здатний отримувати [повідомлення](https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%D0%BC%D1%96%D0%BD_%D0%BF%D0%BE%D0%B2%D1%96%D0%B4%D0%BE%D0%BC%D0%BB%D0%B5%D0%BD%D0%BD%D1%8F%D0%BC%D0%B8), обробляти дані, та надсилати повідомлення іншим об'єктам. Кожен об'єкт — своєрідний незалежний автомат з окремим призначенням та відповідальністю..

В даному визначенні можна виділити три частини: 1) OOP використовує в якості базових елементів об'єкти, а не алгоритми (ієрархія "бути частиною", яка була визначена в розділі 1); 2) кожен об'єкт є екземпляром якогось певного класу; 3) класи організовані ієрархічно (див. Поняття про ієрархію "is а" там же). Програма буде об'єктно-орієнтованої тільки при дотриманні всіх трьох зазначених вимог. Зокрема, програмування, не заснований на ієрархічних відносинах, не відноситься до OOP, а називається програмуванням на основі абстрактних типів даних. Кожен стиль програмування має свою концептуальну базу. Кожен стиль вимагає свого умонастрої і способу сприйняття розв'язуваної задачі.

Для об'єктно-орієнтованого стилю концептуальна база - це об'єктна модель. Вона має чотири головні елементи:

• поліморфізм;

• інкапсуляція;

• модульність;

• Успадкування.

Ці елементи є головними в тому сенсі, що без будь-якого з них модель не буде об'єктно-орієнтованої.

Крім головних, є ще три додаткові елементи:

• типізація;

• паралелізм; 17

• збереженість.

Називаючи їх додатковими, ми маємо на увазі, що вони корисні в об'єктної моделі, але не обов'язкові.

Поліморфізм – властивість, яка дозволяє одне і те саме ім’я використовувати для вирішення декількох технічно різних задач, тобто основною метою поліморфізму є використання одного імені для задання загальних класу дій.

Інкапсуляція - це процес відділення один від одного елементів об'єкта, що визначають його пристрій і поведінку; інкапсуляція служить для того, щоб ізолювати контрактні зобов'язання абстракції від їх реалізації.

**Модульність** - це властивість системи, яка була розкладена на внутрішньо зв'язкові, але слабо пов'язані між собою модулі.

**Успадкування** – процес, завдяки якому один об’єкт може придбати властивості іншого, тобто наслідувати властивість іншого обєкту і додавати риси характерні тільки для нього самого.

**Типізація** - це спосіб захиститися від використання об'єктів одного класу замість іншого, або принаймні керувати таким використанням.

**Паралелізм** - це властивість, що відрізняє активні об'єкти від пасивних.

Збереженість - здатність об'єкта існувати в часі, переживаючи породив його процес, і (або) в просторі, переміщаючись зі свого первісного адресного простору.

Отже. без ООП, в сучасних реаліях програмування не обійтись.

**2.2 Вибір мови**

#### Розвиток і стандартизація мови

У [1985 році](https://ru.wikipedia.org/wiki/1985_%D0%B3%D0%BE%D0%B4) вийшло перше видання «[Мови програмування C ++»](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F_C%2B%2B_(%D0%BA%D0%BD%D0%B8%D0%B3%D0%B0)), що забезпечує перший опис цієї мови, що було надзвичайно важливо через відсутність офіційного стандарту. У [1989 році](https://ru.wikipedia.org/wiki/1989_%D0%B3%D0%BE%D0%B4) відбувся вихід C ++ версії 2.0. Його нові можливості включали множинне спадкування, абстрактні класи, статичні функції-члени, функції-константи і захищені члени. У [1990 році](https://ru.wikipedia.org/wiki/1990_%D0%B3%D0%BE%D0%B4) вийшло «Довідкове керівництво по C ++», покладене згодом в основу стандарту. Останні оновлення включали шаблони, виключення простору імен, нові способи приведення типів і булеві типи.

Стандартна бібліотека C ++ також розвивалася разом з ним. Першим додаванням до стандартної бібліотеці C ++ стали потоки введення-виведення, що забезпечують засоби для заміни традиційних функцій C printf і scanf. Пізніше самим значним розвитком стандартної бібліотеки стало включення в неї [Стандартної бібліотеки шаблонів](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82%D0%BD%D0%B0%D1%8F_%D0%B1%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0_%D1%88%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD%D0%BE%D0%B2) .

У [1998 році](https://ru.wikipedia.org/wiki/1998_%D0%B3%D0%BE%D0%B4) був опублікований стандарт мови ISO / IEC 14882: 1998

(відомий як C ++ 98), розроблений комітетом з стандартизації C ++ ( [ISO](https://ru.wikipedia.org/wiki/International_Organization_for_Standardization) / [IEC](https://ru.wikipedia.org/wiki/International_Electrotechnical_Commission) JTC1 / SC22 / WG21 working group). Стандарт C ++ не описує способи іменування об'єктів, деякі деталі обробки виключень і інші можливості, пов'язані з деталями реалізації, що робить несумісним об'єктний код, створений різними компіляторами. Однак для цього третіми особами створено безліч стандартів для конкретних [архітектур](https://ru.wikipedia.org/wiki/%D0%90%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%B0) і [операційних систем.](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0)

У [2003 році](https://ru.wikipedia.org/wiki/2003_%D0%B3%D0%BE%D0%B4) був опублікований стандарт мови ISO / IEC 14882: 2003, де були виправлені виявлені помилки і недоліки попередньої версії стандарту.

У [2005 році](https://ru.wikipedia.org/wiki/2005_%D0%B3%D0%BE%D0%B4) був випущений звіт Library Technical Report 1 (коротко званий TR1). Не будучи офіційно частиною стандарту, звіт описує розширення стандартної бібліотеки, які, як очікувалося авторами, повинні бути включені в наступну версію мови C ++. Ступінь підтримки TR1 поліпшується майже у всіх підтримуваних компіляторах мови C ++.

З2009 рок[у](https://ru.wikipedia.org/wiki/2009_%D0%B3%D0%BE%D0%B4) велася робота по оновленню попереднього стандарту. Попередньою версією нового стандарту спершу був C ++ 09, а через рік - C ++ 0x (згодом перейменований в C ++ 11), куди були включені доповнення в ядро мови і розширення стандартної бібліотеки, в тому числі велику частину TR1.

C ++ продовжує розвиватися, щоб відповідати сучасним вимогам. Одна з груп, що розробляють мову C ++ і напрямних комітету по стандартизації C ++ пропозиції щодо його поліпшення - це [Boost](https://ru.wikipedia.org/wiki/Boost) , яка займається, в тому числі,

вдосконаленням можливостей мови шляхом додавання в нього особливостей [метапрограмування](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%B0%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) .

Ніхто не має права на мову C ++, вона є вільною. Однак сам документ стандарту мови (за винятком чернеток) не доступний безкоштовно. В рамках процесу стандартизації, ISO випускає кілька видів видань.

#### Розвиток мови

У [1985 році](https://ru.wikipedia.org/wiki/1985_%D0%B3%D0%BE%D0%B4) вийшло перше видання «[Мови програмування C ++»](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F_C%2B%2B_(%D0%BA%D0%BD%D0%B8%D0%B3%D0%B0)), що забезпечує перший опис цієї мови, що було надзвичайно важливо через відсутність офіційного стандарту. У [1989 році](https://ru.wikipedia.org/wiki/1989_%D0%B3%D0%BE%D0%B4) відбувся вихід C ++ версії 2.0. Його нові можливості включали множинне спадкування, абстрактні класи, статичні функції-члени, функції-константи і захищені члени. У [1990 році](https://ru.wikipedia.org/wiki/1990_%D0%B3%D0%BE%D0%B4) вийшло «Довідкове керівництво по C ++», покладене згодом в основу стандарту. Останні оновлення включали шаблони, виключення простору імен, нові способи приведення типів і булеві типи.

Стандартна бібліотека C ++ також розвивалася разом з ним. Першим додаванням до стандартної бібліотеці C ++ стали потоки введення-виведення, що забезпечують засоби для заміни традиційних функцій C printf і scanf. Пізніше самим значним розвитком стандартної бібліотеки стало включення в неї [Стандартної бібліотеки шаблонів](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82%D0%BD%D0%B0%D1%8F_%D0%B1%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0_%D1%88%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD%D0%BE%D0%B2) .

У [1998 році](https://ru.wikipedia.org/wiki/1998_%D0%B3%D0%BE%D0%B4) був опублікований стандарт мови ISO / IEC 14882: 1998

(відомий як C ++ 98), розроблений комітетом з стандартизації C ++ ( [ISO](https://ru.wikipedia.org/wiki/International_Organization_for_Standardization) / [IEC](https://ru.wikipedia.org/wiki/International_Electrotechnical_Commission) JTC1 / SC22 / WG21 working group). Стандарт C ++ не описує способи іменування об'єктів, деякі деталі обробки виключень і інші можливості, пов'язані з деталями реалізації, що робить несумісним об'єктний код, створений різними компіляторами. Однак для цього третіми особами створено безліч стандартів для конкретних [архітектур](https://ru.wikipedia.org/wiki/%D0%90%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%B0) і [операційних систем.](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0)

У [2003 році](https://ru.wikipedia.org/wiki/2003_%D0%B3%D0%BE%D0%B4) був опублікований стандарт мови ISO / IEC 14882: 2003, де були виправлені виявлені помилки і недоліки попередньої версії стандарту.

У [2005 році](https://ru.wikipedia.org/wiki/2005_%D0%B3%D0%BE%D0%B4) був випущений звіт Library Technical Report 1 (коротко званий TR1). Не будучи офіційно частиною стандарту, звіт описує розширення стандартної бібліотеки, які, як очікувалося авторами, повинні бути включені в наступну версію мови C ++. Ступінь підтримки TR1 поліпшується майже у всіх підтримуваних компіляторах мови C ++.

З2009 рок[у](https://ru.wikipedia.org/wiki/2009_%D0%B3%D0%BE%D0%B4) велася робота по оновленню попереднього стандарту. Попередньою версією нового стандарту спершу був C ++ 09, а через рік - C ++ 0x (згодом перейменований в C ++ 11), куди були включені доповнення в ядро мови і розширення стандартної бібліотеки, в тому числі велику частину TR1.

C ++ продовжує розвиватися, щоб відповідати сучасним вимогам. Одна з груп, що розробляють мову C ++ і напрямних комітету по стандартизації C ++ пропозиції щодо його поліпшення - це [Boost](https://ru.wikipedia.org/wiki/Boost) , яка займається, в тому числі,

вдосконаленням можливостей мови шляхом додавання в нього особливостей [метапрограмування](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%B0%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) .

Ніхто не має права на мову C ++, вона є вільною. Однак сам документ стандарту мови (за винятком чернеток) не доступний безкоштовно. В рамках процесу стандартизації, ISO випускає кілька видів видань.

### **2.3. Вибір середовища програмування**

Для виконання свого завдання я вибрав Dev-C++.

**Dev-C++** — [вільне](https://uk.wikipedia.org/wiki/%D0%92%D1%96%D0%BB%D1%8C%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%B5_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F)інтегроване середовище розробк[и](https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B5_%D1%81%D0%B5%D1%80%D0%B5%D0%B4%D0%BE%D0%B2%D0%B8%D1%89%D0%B5_%D1%80%D0%BE%D0%B7%D1%80%D0%BE%D0%B1%D0%BA%D0%B8) для мов програмування [С](https://uk.wikipedia.org/wiki/%D0%A1%D1%96_(%D0%BC%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F))[/С++.](https://uk.wikipedia.org/wiki/C%2B%2B) У дистрибутив входить компілятор [MinGW.](https://uk.wikipedia.org/wiki/MinGW) Сам DevC++ написаний на [Delphi.](https://uk.wikipedia.org/wiki/CodeGear_Delphi) Розповсюджується згідно з [GPL.](https://uk.wikipedia.org/wiki/GNU_General_Public_License)

Проект підтримується [SourceForge.](https://uk.wikipedia.org/wiki/SourceForge) Засновник проекту Колін Лаплас, компанія Bloodshed Software.

Свого часу був доступний Linux-порт, проте на теперішній час актуалізована тільки [Windows-](https://uk.wikipedia.org/wiki/Microsoft_Windows)версія.

На поточний момент не розробляється, замість нього активно розробляється порт інтерфейсу Dev-C++ на [wxWidgets](https://uk.wikipedia.org/wiki/WxWidgets) — [wxDev-C++.](https://uk.wikipedia.org/wiki/WxDev-C%2B%2B)

30 червня 2011 незалежним програмістом було випущено неофіційну версію 4.9.9.3, що включала новіший компілятор GCC 4.5.2, ресурси Windows SDK (Win32 та D3D). Було виправлено багато помилок та покращено стабільність. 27 серпня, після 5-річного перебування у стані бета-версії, розробником з блогу Орвела було випущено версію 5.0. Цей варіант має власну сторінку на SourceForge, починаючи з версії 5.0.0.5, оскільки попередні розробники не відповідають на запити

# **РОЗДІЛ ІІІ. ПРОГРАМА ТА ЇЇ ОСОБЛИВОСТІ**

## **3.1. Аналіз програми**

Програма призначена для визначення і розрахування площі і об’єму тетраедра. Розроблена дана програма за допомогою класів, з використанням головних парадигм ООП.

//-------------------Директиви та заготовочні файли-----------------------

#include <iostream> // підключення бібліотеки з класами, функціями і змінними для організації введення-виведення в мові програмування C ++.

#include <math.h> // підключення бібліотеки з [прототипами функцій,](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D1%82%D0%B8%D0%BF_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D1%96%D1%97) розроблених для виконання простих математичних операцій.

#include <fstream> // підключення бібліотеки що включає набір класів, методів і функцій, які надають інтерфейс для читання / запису даних в файл using namespace std; // це декларативна область, в рамках якої

визначаються різні ідентифікатори (імена типів, функцій, змінних, і т.д.).

//----------Абстрактний клас---------------// *class Figure { public:*

*virtual float Area() const = 0; // віртуальна функція для обрахування площі і ~Figure() {} };*

//----------Клас для вирахування площі тетраедра---------------// *class Tetraid\_ploshcha : public Figure { float a; public:*

*Tetraid\_ploshcha() : a(0) {}*

*Tetraid\_ploshcha(float a1) : a(a1) {}*

*virtual float Area() const { // функція для обрахування площі тетраедра return a\*a\*sqrt(3);*

*}*

*};*

*//----------Клас для вирахування об’єму тетраедра---------------// class Tetraid\_obiem : public Figure { float a; public:*

*Tetraid\_obiem() : a(0) {}*

*Tetraid\_obiem(float a1) : a(a1) {} virtual float Area() const { // функція для обрахування об’єму тетраедра return (a\*a\*a\*sqrt(2))/12;*

*}*

*};*

*//-------Створення динамічного масиву, та виділення для нього памяті-----*

*------//*

*Figure \*mas[2]; mas[0] = new Tetraid\_ploshcha(a); mas[1] = new Tetraid\_obiem(a);*

*//-------Вивід данних на екран консолі-----------// cout << "Ploshcha Tetraida "<<" = "<<mas[0]->Area()<<" sm kv;"<<endl; cout << "Obiem Tetraida "<<" = "<<mas[1]->Area()<<" sm kyb."<<endl;*

*//-------Запис в данних в файл-----------// ofstream ofs("output.txt"); ofs<<"Ploshcha Tetraida "<<" = "<<mas[0]->Area()<<" sm kv;"<<endl; ofs<<"Obiem Tetraida "<<" = "<<mas[1]->Area()<<" sm kyb."<<endl; ofs.close();*

*//-------Звільнення динамічної памяті-----------// for (int i=0;i!=2;i++) {*

*delete mas[i];*

*}*

#### 3.2.Створення програми

Для початку ми створюємо заголовний файл з розширенням .h для того, щоб нам було легше виявляти помилки, та виправляти їх. В даному файлі ми записуємо, так званий, початковий код: оголошуємо клас, функції та змінні. Заголовні файли ми створюємо до кожної формули, яку будемо використовувати в основній задачі та до абстрактної функції *(SizeFormula; SquareFormula; RadiusInsideFormula; RadiusOutsideFormula; FormulaInterface).*

Першим кроком в створенні заголовного файлу, типу *(SizeFormula; SquareFormula; RadiusInsideFormula; RadiusOutsideFormula)* ми підключаємо директиви:

#ifndef - у випадку, якщо ідентифікатор не був виявлений директивою

#define;

**#define - виявляє ідентифікатор.**

**Далі ми підключаємо абстрактну функцію "***FormulaInterface*".

Оголошуємо потрібний нам клас надаємо йому модифікатор доступу. Та додаємо в кінці файлу #endif.

Наш заголовний файл буде мати такий вигляд:

#ifndef SIZEFORMULA\_H

#define SIZEFORMULA\_H

#include "FormulaInterface.h" class SizeFormula : public FormulaInterface

{ public:

SizeFormula(const float length = 1); inline virtual void setLength(const float length) noexcept override final; inline virtual float calculate() const override final;

private:

float m\_length = -1;

};

#endif

Тепер ми підключаємо створений заголовний файл в основний код нашої формули, оголошуємо потрібні бібліотеки, клас, функцію та саме обчислення важливих формул:

*#include "SizeFormula.h"*

*#include <stdexcept>*

*#include <math.h>*

*SizeFormula::SizeFormula(const float length)*

*{ setLength(length);*

*} void SizeFormula::setLength(const float length) noexcept*

*{ m\_length = length;*

*} float SizeFormula::calculate() const*

*{ if (m\_length < 0.1f)*

*throw std::runtime\_error("Length cannot be less than 0.1!");*

*return 5.0f/12.0f \* (3.0f + static\_cast<float>(sqrt(5))) \* m\_length \* sqr(m\_length);*

*}*

Етапи створення заголовного файлу та підключення його в код, ми повторюємо з усіма формулами: *SizeFormula; SquareFormula;*

*RadiusInsideFormula; RadiusOutsideFormula; FormulaInterface.*

Залишилось найважче, а саме підключити всі, щойно створені, заголовні файли в основний код нашої програми *main.cpp,* оголосити всі функції, та вивести результат на екран. Починаємо з бібліотек, оголошення функцій, щоб програма могла їх розпізнавати, тим самим перевантажуючи кожну з них, вводимо сталі числа для кожної з формул та виводимо результат обчислених формул за допомогою "cout".

*#include <iostream>*

*#include "RadiusInsideFormula.h"*

*#include "RadiusOutsideFormula.h"*

*#include "SizeFormula.h" #include "SquareFormula.h" using namespace std; void result(const RadiusInsideFormula &formula); void result(const RadiusOutsideFormula &formula); void result(const SizeFormula &formula); void result(const SquareFormula &formula); void result(const FormulaInterface \*formula); int main()*

*{*

*RadiusInsideFormula rInside(5.0f);*

*RadiusOutsideFormula rOutside;*

*SizeFormula size(3.5f);*

*SquareFormula square;*

*FormulaInterface \*object = new SizeFormula(10.12f); rOutside.setLength(6.7f); square.setLength(1.0f); result(rInside); result(rOutside); result(size); result(square); result(object); delete object; return 0;*

*}*

*void result(const RadiusInsideFormula &formula)*

*{ cout << "Inside circle radius is " << formula.calculate() << ".\n"; }*

*void result(const RadiusOutsideFormula &formula)*

*{ cout << "Outside circle radius is " << formula.calculate() << ".\n";*

*}*

*void result(const SizeFormula &formula)*

*{ cout << "Size is " << formula.calculate() <<*

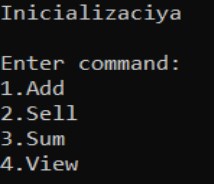
##### **3.3 .Будова програми**

Для написання програми використовувалася така бібліотека як #include <bits/stdc++.h> це бібліотека яка містить кожну стандартну бібліотеку що зменшує кількість програмного коду але збільшує час компеляції що є незначним для нас.

В даній програмі використовується два класи перший це батьківський клас

BaseClass та клас нащадок mSclad

Для батьківського класу встановлено модифікатор доступу public він дає доступ всім хто бачить визначення даного класу.



**ВИСНОВКИ**

Під час розробки програми я вивчив основні функції мови C++. Створивши програму, яка обчислює об'єм і площу правильного тетраедра, я дав можливість користувачеві обчислити об'єм і площу. У програмі застосовано методи і прийоми роботи з класами та різноманітні методи. Основне завдання – обчислити площу та об’єм правильного тетраедра, вивести його на екран консолі та записати у файл.

Також не можна не відзначити, що завдяки написанню курсової роботи я отримав хороший практичний досвід програмування.

Під час розробки програми я вивчив основні функції мови C++. Створивши програму, яка має функції бази даних, я дозволив користувачеві виконувати в ній основні дії. У програмі використовуються методи та прийоми роботи з класами, конструкторами та різними функціями. Головне завдання - знайти відсоток від продажу товару.

Його можна використовувати в будь-яких організаціях.

У 1990-х роках C++ стала однією з найуживаніших мов програмування загального призначення. Мову використовують для системного програмування, розробки прикладного програмного забезпечення, написання драйверів, потужних серверних та клієнтських програм, а також для розробки розважальних програм, наприклад, відеоігор.

**СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ**

1. [Todd Veldhuizen «Techniques for scientific C++»](http://osl.iu.edu/~tveldhui/papers/techniques/)
2. [A Brief Look at C++0x by Bjarne Stroustrup](http://www.artima.com/cppsource/cpp0x.html)
3. [Timothy A. Budd.](https://uk.wikipedia.org/wiki/%D0%A2%D1%96%D0%BC%D0%BE%D1%82%D1%96_%D0%91%D0%B0%D0%B4%D0%B4) *C++* For Java Programmers
4. Матизен В. Э., Дубровский. Із геометриї тетраедра [«Квант»](https://ru.wikipedia.org/wiki/%D0%9A%D0%B2%D0%B0%D0%BD%D1%82_(%D0%B6%D1%83%D1%80%D0%BD%D0%B0%D0%BB))
5. Понарін Я. П. Елементарна геометрія. Том 3. Трикутники та тетраедри
6. Заславський А. А. [Порівняльна геометрія трикутника та тетраедра](http://mi.mathnet.ru/mp141)
7. Граді Буч. Об'єктно-орієнтований аналіз і проектування з прикладами додатків на С ++.
8. [Ліберті Д. Навчись користуватись С++ за 21 день](http://www.cyberforum.ru/cpp-beginners/thread1440139.html)
9. [isocpp.org](https://isocpp.org/) [[Архівовано](https://web.archive.org/web/20180714125833/https:/isocpp.org/) 14 липня 2018 у [Wayback Machine](https://uk.wikipedia.org/wiki/Wayback_Machine).] — домашня сторінка Стандарту C++ в інтернеті — новини, статус і обговорення C++ на усіх компіляторах і платформах.(англ.)
10. [корисні ресурси по C++](https://isocpp.org/get-started) [[Архівовано](https://web.archive.org/web/20211126180707/https:/isocpp.org/get-started) 26 листопада 2021 у [Wayback Machine](https://uk.wikipedia.org/wiki/Wayback_Machine).]
11. [https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%27%D1%94%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D1%96%D1%94%D0%BD%D1%82%D0%BE%D0%B2%D0%B0%D0%BD%D0%B5\_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F#:~:text=%D0%9E%D1%81%D0%BD%D0%BE%D0%B2%D1%83%20%D0%9E%D0%9E%D0%9F%20%D1%81%D0%BA%D0%BB%D0%B0%D0%B4%D0%B0%D1%8E%D1%82%D1%8C%20%D1%87%D0%BE%D1%82%D0%B8%D1%80%D0%B8%20%D0%BE%D1%81%D0%BD%D0%BE%D0%B2%D0%BD%D1%96,%D0%B4%D0%B5%D1%81%D1%8F%D1%82%D0%BA%D0%B0%D0%BC%D0%B8%20%D0%BA%D0%BB%D0%B0%D1%81%D1%96%D0%B2%20%D1%96%D0%B7%20%D1%81%D0%B2%D0%BE8](https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%27%D1%94%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D1%96%D1%94%D0%BD%D1%82%D0%BE%D0%B2%D0%B0%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F#:~:text=%D0%9E%D1%81%D0%BD%D0%BE%D0%B2%D1%83%20%D0%9E%D0%9E%D0%9F%20%D1%81%D0%BA%D0%BB%D0%B0%D0%B4%D0%B0%D1%8E%D1%82%D1%8C%20%D1%87%D0%BE%D1%82%D0%B8%D1%80%D0%B8%20%D0%BE%D1%81%D0%BD%D0%BE%D0%B2%D0%BD%D1%96,%D0%B4%D0%B5%D1%81%D1%8F%D1%82%D0%BA%D0%B0%D0%BC%D0%B8%20%D0%BA%D0%BB%D0%B0%D1%81%D1%96%D0%B2%20%D1%96%D0%B7%20%D1%81%D0%B2%D0%BE%D1%97%D0%BC%D0%B8%20%D0%BC%D0%B5%D1%82%D0%BE%D0%B4%D0%B0%D0%BC%D0%B8)).

**ДОДАТКИ**

*#include <iostream>*

*#include <math.h> #include <fstream> using namespace std; class Figure { public:*

*virtual float Area() const = 0;*

*~Figure() {}*

*}; class Tetraid\_ploshcha : public Figure { float a; public:*

*Tetraid\_ploshcha() : a(0) {}*

*Tetraid\_ploshcha(float a1) : a(a1) {} virtual float Area() const { return a\*a\*sqrt(3);*

*}*

*}; class Tetraid\_obiem : public Figure { float a; public:*

*Tetraid\_obiem() : a(0) {}*

*Tetraid\_obiem(float a1) : a(a1) {} virtual float Area() const { return (a\*a\*a\*sqrt(2))/12;*

*} }; int main() { float a; cout<<"VVedit gran tetraidra"<<endl; cin>>a; Figure \*mas[2]; mas[0] = new Tetraid\_ploshcha(a); mas[1] = new Tetraid\_obiem(a); cout << "Ploshcha Tetraida "<<" = "<<mas[0]->Area()<<" sm kv;"<<endl; cout << "Obiem Tetraida "<<" = "<<mas[1]->Area()<<" sm kyb."<<endl; ofstream ofs("output.txt"); ofs<<"Ploshcha Tetraida "<<" = "<<mas[0]->Area()<<" sm kv;"<<endl; ofs<<"Obiem Tetraida "<<" = "<<mas[1]->Area()<<" sm kyb."<<endl; ofs.close(); for (int i=0;i!=2;i++) { delete mas[i];*

*}*

*}*