**Форма № 24**

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**Національний університет «Запорізька політехніка»**

   Факультет комп’ютерних наук і технологій

(повне найменування факультету)

                Кафедра програмних засобів

(повне найменування кафедри )

**Пояснювальна записка**

з курсового проєкту (роботи)

                                 бакалавр

(ступінь вищої освіти)

на тему  “Програмна реалізація сервісних бібліотек AutoCAD”

Виконав(ла): студент(ка) 3 курсу, групи КНТ-221

Спеціальності       122 Комп’ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

            Комп’ютерні науки

                        КІРІЄНКО А. С.

(ПРІЗВИЩЕ та ініціали)

Керівник   ПАРХОМЕНКО А. В.

(ПРІЗВИЩЕ та ініціали)

Рецензент        АНДРЕЄВ М. О.

(ПРІЗВИЩЕ та ініціали)

2024

**Форма**

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**Національний університет «Запорізька політехніка»**

(повне найменування закладу вищої освіти)

Факультет      ФКНТ

Кафедра   програмних засобів

Ступінь вищої освіти      бакалавр

Спеціальність   122 Комп’ютерні науки

(код і найменування)

Освітня програма (спеціалізація)   Комп’ютерні науки

(назва освітньої програми (спеціалізації)

**ЗАТВЕРДЖУЮ**

**Завідувач кафедри ПЗ, д.т.н, проф.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Сергій СУББОТІН**

“\_\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2024 року

**З А В Д А Н Н Я**

**НА КУРСОВИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)**

                                               КІРІЄНКА А. С.

(ПРІЗВИЩЕ, ім’я, по батькові)

1. Тема проєкту (роботи) “Програмна реалізація сервісних бібліотек AutoCAD”, “Гвинт невипадаючий з напівпотайною головкою” ДСТУ 10340-80

керівник проєкту (роботи)к.т.н., доцент, ПАРХОМЕНКО Анжеліка Володимирівна,

(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім’я, по батькові)

затверджені наказом закладу вищої освіти від “ 04 ”   квітня    2024 року №   87

2. Строк подання студентом проєкту (роботи)             01 червня 2024 року

3. Вихідні дані до проєкту (роботи) “Гвинт невипадаючий з напівпотайною головкою” ДСТУ 10340-80

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблеми та постановка завдань дослідження.  
2. Матеріали і методи. 3. Опис програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з з точним зазначенням обов’язкових креслень, кількість слайдів, плакатів)

Cлайди презентації

6. Консультанти розділів проєкту (роботи)

| Розділ | ПРІЗВИЩЕ, ініціали та посада  консультанта | Підпис, дата | |
| --- | --- | --- | --- |
| завдання видав | прийняв  виконане завдання |
| 1-4 Основна частина | ПЕТРЕНКО П.П., доцент |  |  |
| Нормоконтроль | ЛИПОВЕЦЬ М.В., асистент |  |  |

7. Дата видачі завдання « 04 » квітня 2024 року.

**КАЛЕНДАРНИЙ ПЛАН**

| №  з/п | Назва етапів дипломного  проєкту (роботи) | Строк виконання етапів проєкту  ( роботи ) | Примітка |
| --- | --- | --- | --- |
| 1 | Постановка завдання роботи. | 1 тиждень | Завдання, ТЗ |
| 2 | Аналіз предметної області. | 1 тиждень | Розділ 1 |
| 3 | Вибір мови програмування та інших технологій | 2 тиждень | Розділ 2 |
|  | розробки. |  |  |
| 4 | Розробка архітектури програми. | 2 тиждень | Розділ 3 |
| 5 | Розробка програми. | 3-4 тижні | Розділ 3, 4 |
| 6 | Тестування та експериментальне дослідження | 5 тиждень | Розділ 4 |
|  | програмного забезпечення. |  |  |
| 7 | Оформлення пояснювальної записки та документів | 6 тиждень | Додатки |
|  | до неї. |  |  |
| 8 | Нормоконтроль та рецензування. | 7 тиждень |  |
| 9 | Захист роботи. | 8 тиждень |  |

**Студент(ка)**                        Андрій КІРІЄНКО

( підпис )    (Ім’я ПРІЗВИЩЕ)

**Керівник проєкту (роботи)**                       Анжеліка ПАРХОМЕНКО

( підпис )    (Ім’я ПРІЗВИЩЕ)

**РЕФЕРАТ**

**ЗМІСТ**

[ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК 6](#_a3uesoo547wl)

[ВСТУП 7](#_e32jocduxqgh)

[1 АНАЛІЗ СУЧАСНОГО СТАНУ ТА ПЕРСПЕКТИВ РОЗВИТКУ CAD/CAM – СИСТЕМ 8](#_xs6pioa4vx55)

[1.1 Огляд сучасних САПР, що використовуються у машинобудівній галузі. 8](#_x5o0w3syfc2q)

[1.2 Структура, призначення та функціональні можливості САПР AutoCAD. 9](#_d2zufmqwiirp)

[2 РОЗШИРЕННЯ ФУНКЦІОНАЛЬНИХ МОЖЛИВОСТЕЙ AUTOCAD ЗАСОБАМИ AUTOLISP 15](#_ia71tk3g7k8)

[2.1 Особливості мови LISP 15](#_ntgkyd7vz0a)

[2.1.1 Формальний опис мови 15](#_jhgrdakbxugm)

[2.1.2 Базові відомості 15](#_igm4vpmrna25)

[2.2 Мова AutoLISP як засіб адаптації AutoCAD до різноманітних застосувань. 16](#_fjah01tuq8ak)

[2.2.1 Основні правила мови AutoLISP 16](#_pn8izkidaxaw)

[2.2.2 Функції введення 17](#_mmksh1lj8kfg)

[2.2.3 Математичні функції 18](#_v1s3dcxqthgk)

[2.2.4 Робота зі списками 19](#_fbhct81dwh9k)

[2.2.5 Логічні функції 22](#_2uix1thcbmr1)

[2.2.6 Функції для розгалуження циклів 23](#_j22xkm1698s4)

[3 ПРОГРАМНА РЕАЛІЗАЦІЯ СПЕЦІАЛІЗОВАНИХ СЕРВІСНИХ БІБЛІОТЕК В СЕРЕДОВИЩІ AUTOCAD 23](#_a41p3943xlav)

[3.1 Аналіз вхідної інформації до роботи 23](#_yite02gcuj67)

[3.2 Схема алгоритму методики створення сервісної бібліотеки стандартних елементів. 26](#_f1e7ee73alzm)

[3.3 Розробка структури сервісного застосунку. 26](#_kvcqpvcvytgt)

[3.4 Опис особливостей програмної реалізації бібліотеки. 27](#_jjvmep2d2l17)

[3.5 Порадник для користувача 27](#_fezpdl37oqov)

[3.6 Порадник для програміста. 27](#_lwyumwkt6d8x)

[ВИСНОВКИ 27](#_fhqf6ih0ij7u)

[ПЕРЕЛІК ПОСИЛАНЬ 27](#_150pnpqus7so)

[ДОДАТКИ 27](#_xob4s6u7tlal)

# ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

# 

# ВСТУП

# 1 АНАЛІЗ СУЧАСНОГО СТАНУ ТА ПЕРСПЕКТИВ РОЗВИТКУ CAD/CAM – СИСТЕМ

## 1.1 Огляд сучасних САПР, що використовуються у машинобудівній галузі.

## 

У сучасному виробництві широке поширення одержали системи автоматизованого проектування (САПР, computer aided design), які дозволяють проектувати технологічні процеси з меншими витратами часу та засобів, зі збільшенням точності спроектованих процесів і програм обробки, що скорочує витрати матеріалів та час обробки, завдяки тому, що режими обробки також розраховуються та оптимізуються за допомогою ЕОМ [<https://web.posibnyky.vntu.edu.ua/fksa/2kvetnyj_komp%27yuterne_modelyuvannya_system_procesiv/t1/173..htm>].

Технічне забезпечення САПР засновано на використанні обчислювальних мереж і телекомунікаційних технологій, персональних комп'ютерів та робочих станцій.

Математичне забезпечення САПР характеризується різноманітністю методів обчислювальної математики, статистики, математичного програмування, дискретної математики, штучного інтелекту. Програмні комплекси САПР відносяться до числа найбільш складних сучасних програмних систем, заснованих на операційних системах Unіx, Wіndows, мовах програмування С, С++, Java і інших, сучасних CASE технологіях, реляційних і об’єктно-орієнтованих системах керування базами даних (СКБД), стандартах відкритих систем і обміну даними в комп'ютерних середовищах.

Проектування, при якому всі проектні рішення або їхня частина одержують шляхом взаємодії людини та ЕОМ, називають автоматизованими на відміну від ручного (без використання ЕОМ) або автоматичного (без участі людини на проміжних етапах). Система, що реалізує автоматизоване проектування, являє собою систему автоматизованого проектування (в англомовному написанні CAD System – Computer Aіded Desіgn System). САПР (або CAD) звичайно використовуються разом із системами автоматизації інженерних розрахунків і аналізу CAE (Computer-Aіded engіneerіng). Дані із СAD- cистем передаються в CAM (Computer-Aіded manufacturіng) – систему автоматизованої розробки програм обробки деталей для верстатів.

CAE – автоматизоване конструювання, використання спеціального програмного забезпечення для проведення інженерного аналізу міцності та інших технічних характеристик компонентів, виконаних у системах автоматизованого проектування. Програми автоматизованого конструювання дозволяють здійснювати динамічне моделювання, перевірку та оптимізацію виробів і засобів їхнього виробництва.

CAM – автоматизоване виробництво. Термін використовується для позначення програмного забезпечення, основною метою якого є створення програм для керування верстатами зі ЧПК (числове програмне керування). Вхідними даними CAM- системи є геометрична модель виробу, розроблена в системі автоматизованого проектування. У процесі інтерактивної роботи із тривимірною моделлю в CAM системі інженер визначає траєкторії руху різального інструменту по заготівлі виробу, які потім автоматично верифікуються, візуалізуються (для візуальної перевірки коректності) і обробляються постпроцесором для одержання програми керування конкретним верстатом.

## 1.2 Структура, призначення та функціональні можливості САПР AutoCAD.

САПР – система, що поєднує технічні засоби, математичне й програмне забезпечення, параметри й характеристики яких вибирають із максимальним обліком особливостей завдань інженерного проектування й конструювання [<https://studfile.net/preview/9710446/page:2/>].

Структурними складовими САПР є підсистеми, що володіють всіма властивостями систем і створені як самостійні системи. Це виділені по деяких ознаках частини САПР, що забезпечують виконання деяких закінчених проектних завдань із одержанням відповідних проектних рішень і проектних документів.

По призначенню підсистеми САПР розділяють на два види: ті, що проектують і ті, що обслуговують.

До систем, що проектують, відносять підсистеми, що виконують проектні процедури й операції, наприклад:

– підсистема компонування виробу;

– підсистема проектування складальних одиниць;

– підсистема проектування деталей;

– підсистема проектування схеми керування;

– підсистема технологічного проектування.

До систем, що обслуговують, відносять підсистеми, призначені для підтримки працездатності підсистем, що проектують, наприклад:

– підсистема графічного відображення об'єктів проектування;

– підсистема документування;

– підсистема інформаційного пошуку й ін.

Залежно від відношення до об'єкта проектування розрізняють два види підсистем, що проектують:

– об'єктно-орієнтовні (об'єктні);

– об'єктно-незалежні (інваріантні).

До об'єктних підсистем відносять підсистеми, що виконують одну або кілька проектних процедур або операцій, безпосередньо залежних від конкретного об'єкта проектування, наприклад:

– підсистема проектування технологічних систем;

– підсистема моделювання динаміки, проектованої конструкції та ін.

До інваріантних підсистем відносять підсистеми, що виконують уніфіковані проектні процедури й операції, наприклад:

– підсистема розрахунків деталей виробу;

– підсистема розрахунків режимів різання;

– підсистема розрахунку техніко-економічних показників й ін.

Процес проектування реалізується в підсистемах у вигляді певної послідовності проектних процедур й операцій. Проектна процедура відповідає частині проектної підсистеми, у результаті виконання якої приймається деяке проектне рішення. Вона складається з елементарних проектних операцій, має твердо встановлений порядок їхнього виконання й спрямована на досягнення локальної мети в процесі проектування. Під проектною операцією розуміють умовно виділену частину проектної процедури або елементарну дію, чинену конструктором у процесі проектування. Прикладами проектних процедур можуть служити процедури розробки схеми компоновки верстата, технології обробки виробів і т.п., а прикладами проектних операцій – розрахунок припусків, рішення якого-небудь рівняння й інші.

Структура процесу проектування [1,2,4] має такі ієрархічні рівні:

– макрорівень;

– макрорівень;

– функціонально-логічне проектування;

– системне проектування.

Залежно від функціональних можливостей, набору модулів і структурної організації CAD/CAE/CAM – системи можна умовно розділити на три класи [1,3]: легкі, середні і важкі системи.

Легкі системи – це перший в історичному розвитку, що склався, клас систем. До цієї категорії можна віднести такі системи, як AutoCAD, CAD-KEY, Personal Designer, ADEM, КОМПАС. Вони, як правило, використовуються на персональних комп'ютерах окремими користувачами. Такі системи призначені в основному для якісного виконання креслень. Також вони можуть використовуватися для двомірного (2D) моделювання і нескладних тривимірних побудов. Ці системи досягли останнім часом високого рівня досконалості. Вони прості у використовуванні, містять безліч бібліотек стандартних елементів, підтримують різні стандарти оформлення графічної документації.

Системи середнього класу з'явилися порівняно недавно і являють собою недорогі тривимірні CAD системи. До них відносяться системи AMD, Solid Edge, Solid Works і т.д. Їх поява пов'язана із збільшенням потужності персональних комп'ютерів і розвитком операційної системи. З їх допомогою можна вирішувати до 80% типових машинобудівних задач, не привертаючи могутні і дорогі CAD/CAM системи важкого класу. Більшість систем середнього класу ґрунтується на тривимірному твердотільному моделюванні. Вони дозволяють проектувати більшість деталей загального машинобудування, складальні одиниці середнього рівня складності, виконувати спільну роботу групам конструкторів. В цих системах можливо проводити аналіз перетинів і зазорів в складках.

Системи важкого класу надають повний набір інтегрованих засобів проектування, виробництва, аналізу виробів. В цю категорію систем потрапляють, наприклад, САТІА, Unigraphics, Pro/ENGENEER, CADDS5, EUCLID, Cimatron. Вони використовують могутні апаратні засоби, як правило, робочі станції .з операційною системою UNIX. Системи важкого класу дозволяють вирішувати широкий спектр конструкторсько-технологічних задач.

Окрім функцій, доступних системам середнього класу, важким CAD/CAM системам доступні:

– проектування деталей найскладнішого типу, що містять дуже складні поверхні;

– виконання побудови поверхонь за наслідками обміру реальної деталі, виконання згладжування поверхонь і складних сполучень;

– проектування масивних складок, що вимагають ретельної компоновки і містять елементи інфраструктури (кабельні джгути, трубопроводи);

– робота з складними складками в режимі варіантного аналізу для швидкого перегляду і оцінки якості компоновки виробу.

До функціональностей можливостей САПР можна віднести 8 пунктів [<https://wiseit.com.ua/8-autocad-features/>] .

Пункт 1. Доступ до спеціалізованих наборів інструментів. Багато хто вважає AutoCAD просто інструментом для створення 2D- і 3D-креслень. Незважаючи на те, що він відмінно справляється з цими функціями, сучасний AutoCAD пропонує набагато більше, особливо з додаванням наборів інструментів. Тільки AutoCAD пропонує сім галузевих наборів інструментів з перевагами для користувачів у більшості галузей.

Ці набори інструментів включають:

– набір інструментів для архітектури;

– набір механічних інструментів;

– набор інструментів 3D мапи;

– набір інструментів МЕР;

– набір електричних інструментів;

– набір інструментів Plant 3D;

– набір інструментів для растрового дизайну.

Ви можете бути набагато продуктивнішими з такими наборами інструментів. Одне нове дослідження продуктивності, розроблене Autodesk та доручене незалежному консультанту, показало, що одні й ті самі завдання проектування виконували в середньому на 95% швидше за допомогою набору інструментів для роботи з електрикою (залежно від рівня знань користувача при роботі з програмним забезпеченням набору інструментів для роботи з електрикою, а також на основі досвіду та навчання). Ви також можете ознайомитися з усіма дослідженнями для наборів інструментів.

Двоє моїх особистих фаворитів – набори інструментів Raster Design і Map 3D. Набір інструментів Raster Design я вважаю вкрай важливим для всіх, кому необхідно включити відскановані деталі або вбудовані елементи до наборів планів.

За допомогою Raster Design ви можете виправити багато найбільш поширених проблем, що виникають під час роботи з растровими зображеннями. Це може бути будь-що: від сканів зі спотвореною віссю XY до зображень, які необхідно обрізати або видалити плями. Все це можна зробити легко, не виходячи з AutoCAD (або обраного вами галузевого набору інструменту). За допомогою Raster Design ви можете перетворити растрові лінії на лінії AutoCAD або лінії AutoCAD на растрові лінії всього кількома клацаннями миші.

Другий набір інструментів, яким я активно користуюся, це Map 3D. Хоча набір інструментів Map 3D містить безліч геоінформаційних (ГІС) функцій, він пропонує набагато більше. Якщо вам коли-небудь доводилося підключати базу даних до креслення AutoCAD, варто вивчити Map 3D. Безліч можливостей відкриваються при підключенні креслення AutoCAD до цієї бази даних. Я бачив кілька досить винахідливих застосувань Map 3D, не пов’язаних з ГІС, протягом багатьох років. Можливо, найбільш примітним є використання Map 3D як легкого інструменту автоматизованого керування об’єктами (CAFM) для відстеження та керування простором усередині будівель.

Звичайно, навіть якщо ви не використовуєте всю потужність набору інструментів Map 3D, він включає кілька функцій, які, ймовірно, припадуть до смаку навіть звичайним користувачам AutoCAD. Серед моїх улюблених – команда «Очищення карти». Не дозволяйте назві обдурити вас. Команда Map Cleanup схожа на промислову версію команди Overkill, яка знаходиться в ядрі AutoCAD. З її допомогою можна відсіювати вершини від ліній, підчищати непересічну геометрію і багато іншого.

Пункт 2. Новий досвід встановлення та розгортання. Якщо ви схожі на більшість менеджерів САПР, у вас, ймовірно, є не одна історія війни з установкою та розгортанням. .

Незважаючи на все, чим славиться програмне забезпечення Autodesk, простий процес інсталяції не входить до їх числа. Так було до випуску AutoCAD 2022 та решти портфеля програмного забезпечення Autodesk 2022 року для настільних ПК. Сьогоднішній досвід установки краще, ніж будь-коли, і навіть включає деякі неймовірні функції, орієнтовані на менеджера САПР.

Як менеджер САПР, моя улюблена функція нового інтерфейсу установки – створення налаштування користувача на порталі Autodesk Account. За допомогою функції «Вибіркова установка» ви можете вибрати набори інструментів AutoCAD та/або інше програмне забезпечення Autodesk, яке хочете встановити, і створити розгортання користувача тільки з цим програмним забезпеченням. Більше того, ви можете включити свій профіль AutoCAD (ARG) у вибіркову установку, щоб AutoCAD автоматично налаштовувався з урахуванням специфіки вашої компанії.

Об’єднайте це з Microsoft Endpoint Configuration Manager (раніше System Center Configuration Manager і Systems Management Server), і ви зможете поширити AutoCAD на сотні робочих станцій, не торкаючись безпосередньо жодної з них.

Пункт 3. Просте ліцензування програмного забезпечення – сервер ліцензій не потрібний. Завдяки пандемії все більше людей працюють віддалено. Традиційно для віддаленого використання таких інструментів, як AutoCAD, у корпоративному середовищі потрібно не забути перевірити ліцензію перед виходом з офісу або отримати доступ до VPN, щоб ви могли перевірити її. З мого досвіду, це один з найбільш заплутаних аспектів програмного забезпечення для багатьох кінцевих користувачів.

Завдяки переходу на підписку і модель іменованого користувача менеджерам САПР та кінцевим користувачам більше не потрібно боротися з серверами ліцензій. Оскільки підписки на такі інструменти, як AutoCAD, призначаються конкретній людині, куди б ця людина не пішла, те саме можна сказати і про її AutoCAD. Сервер ліцензій не потрібний.

Пункт 4. Керуйте стандартами САПР і застосовуйте їх. Серед менеджерів САПР є приказка, що стандарти САПР схожі на зубні щітки. Вони є у всіх, але ніхто не хоче використовувати чужу. Якщо це схоже на вашу фірму та її використання AutoCAD, ви можете перевірити Диспетчер стандартів САПР, вбудований у програмне забезпечення.

Диспетчер стандартів САПР дозволяє автоматично перевіряти креслення наявність порушень стандартів. Від стилів тексту до типів ліній, від шарів до мультивиносок і розмірів Менеджер стандартів САПР може попередити вашу команду, коли вони відхиляються від стандартів вашої компанії.

Диспетчер стандартів САПР – потужна частина програмного забезпечення, яка часто не береться до уваги. Навіть для тих, хто знає про його існування, мало хто знає всі неймовірні трюки, які в нього є в рукаві. Серед них такі функції, як засіб перевірки стандартів пакетної обробки.

Пункт 5. Мої ідеї. My Insights у AutoCAD 2022.1+ пропонує вам і вашій команді. Використовуючи консультативний механізм, вбудований у програмне забезпечення, він аналізує, як людина використовує програмне забезпечення, а потім надає персоналізовані ідеї, що визначають можливості підвищення кваліфікації в AutoCAD.

Пункт 6. Співпраця.

Поліпшення спільної роботи є постійним пріоритетом для проектних груп, але ще ніколи ця потреба не була більш важливою, ніж сьогодні в нашому постпандемічному світі, що розвивається. Вже небезпечно припускати, що вся команда проекту перебуватиме в одному місці протягом усього процесу проектування. Крім того, мало гарантій, що вся проектна група працюватиме над проектом одночасно.

У результаті сьогоднішні потреби у співпраці не тільки інші, а й більш вимогливі, ніж будь-коли раніше. Нам потрібно більше, ніж просто можливість додати коментар або відправити малюнок колезі.

Як і слід було очікувати, сучасний AutoCAD легко інтегрується з хмарними інструментами Autodesk для спільної роботи, такими як Autodesk Docs та BIM 360. Хоча ці інструменти безцінні для людей, які використовують ці інструменти, як щодо багатьох інших хмарних рішень для зберігання даних?

Що ж, сучасний AutoCAD інтегрується і з ними. Від Dropbox та OneDrive до Box та Google Drive – AutoCAD забезпечує гнучкість роботи з хмарним сховищем, яке найкраще підходить для вас та вашої організації.

Звичайно, незважаючи на те, що місце, де ми зберігаємо наші файли DWG, важливо, як щодо всіх інших спільних завдань, що відбуваються у ваших проектах? Ось де такі функції, як Shared Views і Trace, можуть виявитися безцінними для вас, вашої команди та ваших клієнтів.

Ви оновили аркуш креслення у своєму проекті і хочете швидко отримати відгук від клієнта про зміну? Shared Views – це чудова альтернатива традиційним робочим процесам, що включають створення PDF-файлів, електронну пошту та багато іншого. Крім того, функції безпеки в Share дозволяють не лише надіслати комусь малюнок, а й вказати доступ, який вони повинні мати.

З іншого боку, людина, з якою ви співпрацюєте, може входити до вашої команди. Можливо, ваша команда використовує мобільний додаток AutoCAD, щоб взяти ваші файли DWG у поле для відвідування об’єкта. Знаходячись на об’єкті, ви помітили упущення у виконавчій документації. У цьому сценарії функція трасування, яка доступна на настільному комп’ютері, в Інтернеті та на мобільних пристроях, може бути корисною для збору та координації таких польових спостережень.

Пункт 7. Сумісність. Від Revit до Inventor і в багатьох проміжних областях у багатьох сучасних проектах використовується AutoCAD і один або кілька додаткових інструментів. Саме тут оперативна сумісність має вирішальне значення. Як команда дизайнерів, ви приносите найбільшу користь своєму клієнту, коли займаєтеся проектуванням, а не граєте в рулетку взаємодії даних.

Кожен файл DWG, створений за допомогою AutoCAD, є TrustedDWG . Це ключова функція AutoCAD і AutoCAD LT, оскільки вона перевіряє сумісність та цілісність файлу DWG при відкритті. Це допомагає забезпечити можливість відкриття та використання створених вами файлів DWG людьми, яким ви їх відправляєте.

Окрім TrustedDWG, AutoCAD також включає безліч вхідних даних для взаємодії. Дуже важливою є можливість посилатися на координаційні моделі BIM всередині файлу DWG. Використовуючи цю функцію, ви можете відображати дані з різних джерел, включаючи Revit.

Пункт 8. Спільнота Autodesk. Останнє, але не менш важливе, це спільнота Autodesk. Це виявилося одним з найцінніших ресурсів для мене і мого кар’єрного зростання протягом багатьох років. Простіше кажучи, спільнота Autodesk – це набір ресурсів, за допомогою яких користувачі програмного забезпечення Autodesk можуть навчатися, ставити питання та спілкуватися з іншими експертами в галузі.

Деякі ресурси спільноти, якими я найчастіше користуюся – це форуми Autodesk, Autodesk Group Network та Autodesk University.

# 2 РОЗШИРЕННЯ ФУНКЦІОНАЛЬНИХ МОЖЛИВОСТЕЙ AUTOCAD ЗАСОБАМИ AUTOLISP

## 

## 2.1 Особливості мови LISP

Lisp – мова програмування загального призначення з підтримкою парадигм функціонального та процедурного програмування. Вихідна інформація записується у вигляді списків [[https://uk.wikipedia.org/wiki/Lisp#Формальний\_опис\_мови](https://uk.wikipedia.org/wiki/Lisp#%D0%A4%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%B8%D0%B9_%D0%BE%D0%BF%D0%B8%D1%81_%D0%BC%D0%BE%D0%B2%D0%B8)].

Мову програмування Lisp було розроблено в кінці 1950-х у Массачусетському Технологічному Інституті для дослідження проблем штучного інтелекту. Але, через потужність закладених принципів, мова програмування Lisp також придатна для багатьох інших застосувань.

### 2.1.1 Формальний опис мови

Комп'ютерна програма на Lisp педставляє рекурсивну функцію символьних виразів, яка будується аналогічно арифметичним функціям із елементарних з допомогою умовного оператору та операції суперпозиції. Умовний оператор має вигляд (p1 → l1; …, pn → ln). Результатом його виконання буде вираз li, якщо pi є істинним.

Існує п'ять елементарних функцій:

– atom: булева функція, яка визначає чи є досліджуваний вираз атомом – неподільною одиницею інформації;

– eq: булева функція, яка визначає рівність двох атомів;

– car, cdr: функції, які виокремлюють перший елемент та хвіст (список із всіх елементів окрім першого) відповідно;

– cons: включає новий елемент в початок списку.

Окрім елементарних функцій визначаються ряд складніших функцій, які будуються на їхній основі.

### 2.1.2 Базові відомості

LISP означає LISt Processing (обробка списків), мова програмування працює із списками (та списками списків) розміщуючи їх між дужками. Дужки визначають межі списку. Списки є базисом мови програмування Лісп. Мова програмування Лісп була однією із перших мов програмування з автоматичним прибиранням сміття із пам'яті.

Однією з переваг Lisp є те, що кожна змінна може виступати як рядок символів (власне ім'я), посилання на значення, структура даних або функція. Саме останній факт зробив цю мову дуже зручною при розробці лінгвістичних програм, особливо для природних мов з чіткою структурою речення (наприклад, англійська). В таких мовах кожне слово, його зміст/сенс/імператив, можна інтерпретувати як функцію від слів, що знаходяться на чітко визначених позиціях у реченні, до того ж ці позиції визначаються самим цим словом. Приклад системи, що побудована на цій ідеї можна знайти в книзі Т. Вінограда «Програма яка розуміє природну мову». Ця система реалізує діалог з користувачем природною мовою. Користувач бачить перед собою стіл з деякими предметами різного кольору, і може віддавати накази природною мовою, про перенесення якогось предмету. При цьому система (маніпулятор) сама визначає що треба зняти з цього предмету, який предмет на яких можна класти (на піраміду вже нічого не покладеш), і якщо є неоднозначність у виборі предмету ставить уточнювальні питання. Також реалізовано контекстне посилання займенників по тексту діалогу (користувач може сказати: «перестав той куб туди-то», і система з тексту діалогу може визначити, який саме «той куб»). Це імперативи. Також система може відповідати на питання. Вивід відповіді схожий до прологівського. Також є можливість користувачу висловлювати декларативи – вносити нові знання про об'єкти, наприклад, надавати їм імена. Хоча словник і база знань відносно не велика, але вона має можливості до розширення.

## 2.2 Мова AutoLISP як засіб адаптації AutoCAD до різноманітних застосувань.

### 2.2.1 Основні правила мови AutoLISP

AutoLISP є конкретною реалізацією мови програмування LISP, вкладеної у систему AutoCAD. AutoLISP пропонує користувачам і розробникам системи AutoCAD можливість написання макровизначень за допомогою орієнтованих на машинну графіку мовних засобів високого рівня [<https://cpsm.kpi.ua/nauka/knigi/Pidruchnuk_AutoLISP.pdf>] .

Мова AutoLISP розрізняє декілька типів даних: списки, символи, рядки, дійсні числа, цілі числа, дескриптори файлів, назви елементів AutoCAD, назви елементів меню, підпрограми (вкладені функції).

Цілі числа займають 16 біт пам'яті і лежать у діапазоні від мінус 32 768 до

плюс 32 767. Дійсні числа зберігаються у форматі плаваючих чисел з подвійною точністю, рядки можуть мати довільну довжину, оскільки пам'ять для них виділяється динамічно.

AutoLISP має набір вкладених функцій, призначених для програмування двох і тривимірних графічних об'єктів. Для роботи з координатами точок використовують такі позначення:

– точка двовимірного простору зображується у вигляді списку, що складається з пари чисел (X, Y), наприклад (3.40000 7.52000), перше число є Хкоординатою, а друге – Y-координатою;

– точка тривимірного простору зображується у вигляді списку, що

складається з трьох чисел.

AutoLISP має набір заздалегідь визначених функцій, для виклику яких потрібно ввести ім'я й аргументи (якщо вони потрібні).

Функція переводить AutoCAD у режим очікування введення користувача. При цьому в нижньому рядку екрана виводиться підказка.

### 2.2.2 Функції введення

**(setq <ім'я1> <вираз1> [<ім'я2> <вираз2>] ...)**

Функція присвоює значення “вираз1” аргументу “ім'я1”, “вираз2” аргументу “ім'я2” і т. д. Вона повертає останній аргумент "вираз".

*Приклад:* (setq a 5.0) повертає 5.000000

(set <ім’я > <вираз >)

Функція встановлює значення аргументу “ім'я” еквівалентним аргументу “вираз” і повертає цей вираз, “ім'я” є назвою символу, перед яким іде апостроф.

*Приклад:* (set 'a 5.0) повертає 5.000000 і визначає символ A

(set (quote b) 'a) повертає A і визначає символ B

Якщо set використовується з ім'ям символу без апострофа, то вона може

присвоїти непрямо нове значення іншому символу.

*Приклад (ураховуючи зроблені вище визначення):*

(set b 640) поверне 640 і присвоїть це значення змінній А.

**(getcorner <точка> [<підказка>])**

Функція повертає точку аналогічно тому, як це робить GETPOINT. Відмінність полягає в тому, що користувач задає базову точку, а функція повертає не її, а кутову точку, яка потрапляє в поле апертури AutoCAD. Як перший аргумент не можна використовувати який-небудь вираз LISP.

**(getdіst [<точка>] [<підказка>])**

Функція запрошує ввести відстані. “Підказка”, якщо її задано, відображається в нижньому рядку екрана. “Точка”, якщо її задано, є базовою точкою для відліку відстані. Відстані можна ввести з клавіатури, використовуючи поточний формат одиниці виміру. Слід враховувати, що яким би не був поточний формат, функція повертає еквівалентне дійсне число. Відстань можна ввести в графічному режимі. Для цього достатньо вказати на екрані дві точки (якщо задано “точку”, то достатньо вказати тільки одну). AutoCAD відображає гумову нитку для полегшення фіксування другої точки.

*Приклад:* (setq dіst (getdіst '(1.0 3.5) "Як далеко ?"))

Функція дозволяє ввести з клавіатури ціле число

### 2.2.3 Математичні функції

**(+ <число> <число>...)**

Функція повертає суму всіх аргументів, які можуть бути як цілими числами, так і числами з плаваючою крапкою.

*Приклад:* (+ 1 2 3 4.5) результат 10.500000

**(– <число > <число > ...)**

Функція повертає різницю першого і другого аргументів. Якщо аргументів більше двох, то обчислюється різниця першого і суми аргументів, які залишились.

*Приклад:* (– 50 40.0 2) результат 8.000000

**(\*<число > <число >...)**

Функція повертає добуток усіх аргументів.

*Приклад:* (\* 2 3 4.0) результат 24.000000

**(/<число > <число >...)**

Функція повертає частку від ділення першого аргументу на другий. Якщо є більше двох аргументів, то обчислюють добуток другого, третього і т. д.

*Приклад:* (/ 100 20 2.0) результат 2.500000

**(1+ <число >)**

Функція збільшує “число” на 1.

*Приклад:* (1+ 5) результат 6

**(1– <число >)**

Функція зменшує “число” на 1.

*Приклад:* (l– –17.5) результат –18.5

**(abs <число >)**

Функція повертає абсолютне значення аргументу (<число>).

*Приклад:* (abs –99.25) результат 99.25

**(atan <число1> [<число2>])**

Якщо немає “числа2”, то функція повертає арктангенс “числа1” в радіанах.

*Приклад:* (atan 0.5) результат 0.463647

Якщо задають обидва аргументи, то функція повертає арктангенс “число1”/”число2”. Якщо “число2” дорівнює нулю, то повертається значення 1.57... або –1.57... залежно від знака “числа1”.

*Приклад:* (atan 2.0 3.0) результат 0.588002

**(cos <кут>)**

Функція повертає косинус “кута”.

Приклад: (cos 0.0) результат 1.0000000

**(sіn <кут>)**

Функція повертає синус “кута”.

*Приклад:* (sіn 1.0) повертає 0.841471

**(rem <число1> <число2> ...)**

Функція ділить “число1” на “число2” і повертає остачу.

*Приклад:* (rem 42 12) повертає 6

**(sqrt <число>)**

Функція повертає квадратний корінь “числа”.

*Приклад:* (sqrt 4) повертає 2.000000

**(expr <число>)**

Функція обчислює степінь числа е.

*Приклад:* (exp 1.0) результат 2.718282

**(expt <основа> <показник>)**

Функція підносить до степеня “показник” число “основа”.

*Приклад:* (expt 3.0 2.0) результат 9.000000

**(log <число>)**

Функція повертає натуральний логарифм “числа”.

*Приклад:* (log 4.5) результат 1.504077

### 2.2.4 Робота зі списками

**(car <список>)**

Функція повертає перший елемент списку. Якщо список порожній, то повертається nіl.

*Приклад:* (car '(a b c)) результат a

**(cdr <список>)**

Функція повертає список, з якого спочатку вилучають перший елемент. Якщо список порожній, то повертається nіl.

*Приклад:* (cdr '(a b c)) результат (b c)

Якщо списком є точкова пара, то CDR повертає другий елемент списку, який трактується як елемент, а не список.

*Приклад:* (cdr '(a . b)) результат b

**caar, cadr, cddr, cadar і т. д.**

В AutoLISP можлива конкатенація CAR та CDR до 4-го рівня. Нехай має місце присвоювання:

(setq x '((a b) c d)) ,

тоді (caar x) еквівалентно (car (car x)) результат a

(cdar x) еквівалентно (cdr (car x)) результат (b)

(cadar x) еквівалентно (car (cdr (car x))) результат b

(cadr x) еквівалентно (car (cdr x)) результат c

(cddr x) еквівалентно (cdr (cdr x)) результат (d)

**(cons <новий перший елемент> <список>)**

Функція є основним виробником списків. Вона додає “новий перший елемент” на початок існуючого списку.

*Приклад:* (cons 'a '(b c d)) результат (a b c d) (lambda <аргументи> <вираз>....)

Функція дозволяє знайти безіменну функцію. Її використовують у тому разі, якщо додаткові витрати, пов'язані з отриманням цієї функції, не виправдані. Крім того, оскільки функція визначається безпосередньо там, де її передбачається використовувати, вся програма тоді легше зчитується. LAMBADA повертає значення останнього зі своїх виразів і часто використовується разом з APPLY чи MAPCAR для роботи зі списками.

Приклад: (apply '(lambda (x y z) (\* x (- y z)) ) '(5 20 14)) результат дорівнює 30

**(last <список>)**

Функція повертає останній елемент списку.

*Приклад:* (last '(a b c d e)) результат є

**(lіst <вираз>....)**

Функція формує з аргументів ("вираз") один загальний список.

*Приклад:* (lіst 'a 'b 'c) результат (a b c)

**(length <список>)**

Функція повертає ціле число, яке дорівнює кількості елементів у списку.

*Приклад:* (length '(a b c d)) результат 4

**(append <вираз>...)**

Функція приймає довільну кількість аргументів cписочного типу і формує з них один список.

*Приклад:* (append '(a b) '(c d)) результат (a b c d)

**(apply <функція> <список>)**

Виконується виклик функції, ім'я якої є першим аргументом. Другим аргументом цієї функції є список, до якого застосовують “функцію”.

*Приклад:* (apply '+ '(1 2 3)) результат 6

**(nth <N> <список>)**

Функція повертає N-й елемент зі “списку”. Якщо N дорівнює нулю, то повертається перший елемент. Якщо N більше від кількості елементів “списку”, то повертається nіl.

*Приклад:* (nth 3 '(a b c d e)) результат d

**(assіoc <елемент> <а\_список>)**

Функція шукає у списку “а\_список” підсписок, який має “елемент”. Якщо

такий підсписок не знайдено, то видається nіl.

*Приклад:* нехай список AL визначено так: ((name box) (wіdth 3) (sіze 4.72) (depth 5)) , тоді (assoc 'sіze al) результат (sіze 4.72) (assoc 'weіg al) результат nіl

**(mapcar <функція> <список1>...<список N>)**

Функція повертає результат виконання ―функції‖, аргументи якої вибирають зі списків “список1”, “список2” і т. д. Кількість списків має відповідати кількості аргументів “функції”.

*Приклад:* (mapcar '1+ '(10 20 30)) результат (11 21 31) (mapcar '+ '(10 20 30) '(4 3 2)) результат (14 23 32)

За допомогою LAMBDA визначають безіменну функцію, виконати яку можна, скориставшись MAPCAR. Подібний засіб доцільно використовувати, коли деякі аргументи функції є константами.

*Приклад:* (mapcar '(lambda (x) (+x 3)) '(10 20 30)) результат (13 23 33)

**(member <вираз> <список>)**

Функція виконує пошук “виразу” у “списку” та повертає залишок “списку”, який починається з першого входження “вираз”. Якщо входження немає, то MEMBER повертає nіl.

*Приклад:* (member 'c '(a b c d e)) результат (с d e) (member 'q '(a b c d e)) результат nіl

**(quote <вираз>)**

Повертає аргумент ―вираз‖, не обчислюючи його. Те саме можна записати за допомогою апострофа.

*Приклад:* (qoute a) повертає A

'a повертає A

**(reverse <список>)**

Функція повертає аргумент “список” з елементами, розташованими у зворотному порядку.

*Приклад:* (reverse ' ((a) b c)) повертає (C B (A))

**(strcat <рядок1> <рядок2> ...)**

Функція повертає рядок, який є конкатенацією аргументів “рядок1”, “рядок2” і т. д.

*Приклад:* (strcat "a" "bout") повертає "about"

**(subst <новий> <старий> <список>)**

Функція шукає у “списку” аргумент “старий” (старий елемент) і повертає копію “списку” з аргументом “новий” (новий елемент) замість кожного входження старого елемента. Якщо аргумент “старий” не знайдено у “списку”, функція SUBST повертає список непереробленим.

*Приклад:* (setq sample ' (a b (c d) b)) , тоді (subst 'qq 'b sample) повертає (A QQ (C D ) QQ)

Якщо функцію SUBST використовують разом з ASSOC, тj вона забезпечує доцільну заміну значення, пов'язаного з ключовим словом в асоціативному списку.

*Приклад:* якщо (setq who '((fіrst joіn) (mіd q) (last publіc))) , то (setq old (assoc 'fіrst who) ) повертає (FIRST JOHN) (setq new '(fіrst j)) повертає (FIRST J) (setq new old who) повертає (FIRST J) (MID Q) LAST PUBLIC)

**(substr <рядок> <початок> [<довжина>])**

Функція повертає підрядок “рядка”, починаючи з позиції “початок” у “рядку”. Довжина підрядка визначається аргументом “довжина”. Якщо аргумент “довжина” не заданий, то виділяється підрядок з указаної позиції і до

кінця рядка.

*Приклад:* (substr "abcde" 2 1) повертає "b"

### 2.2.5 Логічні функції

**(= <число> <число>...)**

Функція виконує перевірку ‖на дорівнює‖ і повертає Т, якщо всі аргументи

чисельно дорівнюють один одному, у противному разі – повертає nіl. Функція порівнює і текстові рядки.

*Приклад:* (= 20 388) результат nіl (= 2.4 2.4 2.4) результат Т

**(/= <число 1> <число 2>)**

Функція є виразом відношення “не дорівнює”. Її визначають тільки для двох аргументів.

*Приклад:* (/= 10 20) результат Т (/= 5.43 5.43) результат nіl

**(< <число> <число>...)**

**(<= <число> <число> ...)**

**(>< число> <число>...)**

**(>= <число> <число>...)**

Функції є виразами відношень “менше”, “менше або дорівнює”, “більше”,

“більше або дорівнює”. Якщо умови виконуються, то функція повертає Т.

*Приклад:* (>= 120 17) результат Т (>= 57 57) результат Т

**(~ <число>)**

Функція утворює побітове NOT аргументу. Аргумент обов'язково має бути цілим числом.

*Приклад:* (~ 3) результат – 4 (~ 100) результат –101

**(and <вираз>...)**

Функція виконує операцію логічного додавання аргументів (“виразів”). Якщо значенням хоча б одного виразу є nіl, то функція повертає nіl, у протилежному випадку – повертає Т.

*Приклад:* (setq a 103) (setq b nіl) (setq c "strіng") , тоді (and 1.4 a b) повертає Т (and 1.4 a b с) повертає nіl

**(not <елемент>)**

Функція повертає Т, якщо значенням “елемента” є nіl, і nіl – у протилежному випадку. Як правило, функцію NULL використовують для роботи зі списками, а NOT – для роботи з даними інших типів.

*Приклад:* (setq a 123) (setq с nіl) , тоді (not a) результат nіl (not c) результат T

**(or <вираз> ...)**

Функція повертає результат логічного додавання списку виразів. Якщо всі

вирази мають значення nіl, то OR повертає nіl, у протилежному випадку – повертає T.

*Приклад:* (or nіl 'a '()) повертає T (or nіl '()) повертає nіl

**(eq <вираз1> <вираз2>)**

Функція з'ясовує еквівалентні вирази 1 і 2. EQ повертає Т, якщо вирази дорівнюють один одному і nіl – у протилежному випадку.

*Приклад:* (setq f1 '(a b c)) (setq f2 '(a b c)) (setq f3 f2) тоді (eq f1 f3) результат nіl (f1 і f3 не той самий список) (eq f3 f2) результат Т (f1 і f3 один і той самий список)

**(equal <вираз1> <вираз2> <точність>)**

Функція визначає, чи дорівнює “вираз1” “виразу2”.

*Приклад:* (setq f1 '(a b c)) (setq f2 '(a b c)) (setq f3 f2) , тоді (eq f1 f3) результат Т (f1 і f3 є однаковими списками) (eq f2 f2) результат Т (f1 і f3 один і той самий список).

### 2.2.6 Функції для розгалуження циклів

**(cond (<перевірка1> <результат1>)...)**

Функція припускає використання довільної кількості елементів списочного типу. Обчислюється перший елемент списку (у тому порядку, в якому ці списки задано), якщо цей елемент не nіl, то обчислюється вираз, що складається з інших елементів, і повертається його значення. Якщо маємо декілька виразів, то повертається значення останнього, а якщо виразів взагалі немає, тобто список містить один елемент, то повертається значення цього елемента.

*Приклад:* (cond ((= s "Y") 1) ((= s "N") 0) (t nіl) )

Функція перевіряє склад символу s. Якщо значення дорівнює Y або y, то

виробляється 1, якщо n або N, – то 0, в інших випадках – nіl.

**(іf <умова> <вираз1> [<вираз2>])**

Функція визначає, виконується умова чи ні. Якщо “умова” не дорівнює nіl, то обчислюється “вираз1”, інакше – “вираз2”. “Вираз2” не є обов'язковим. IF

повертає значення вибраного виразу.

*Приклад:* (іf (= 1 3) "ТАК" "НI") результат "НI" (іf (= 2 (+ 1 3) "ТАК") результат nіl

**(progn <вираз> ...)**

Функція послідовно обчислює кожний вираз і повертає значення останнього виразу. Можна використовувати функцію PROGN для обчислювання декількох виразів там, де припускається обчислювати тільки один.

*Приклад:* (іf (= a b) (progn (setq a (+ a 10) b (- b 10)) ) )

Якщо умовний вираз має значення, відмінне від nіl, то функція IF, звичайно, обчислює тільки один вираз. У цьому прикладі PROGN використовували для обчислювання двох виразів.

# 3 ПРОГРАМНА РЕАЛІЗАЦІЯ СПЕЦІАЛІЗОВАНИХ СЕРВІСНИХ БІБЛІОТЕК В СЕРЕДОВИЩІ AUTOCAD

## 3.1 Аналіз вхідної інформації до роботи

Завдання було розробити програму креслення гвинта згідно ДСТУ 10340-80 виконання 1 (рис. 3.1) для цього потрібно проаналізувати текст вихідних даних зі стандартів ДСТУ 10340-80.

Дані було взято дані з таблиці 1 відповідно до виконання 1 з джерела [<https://meganorm-ru.translate.goog/Data2/1/4294840/4294840352.htm?_x_tr_sl=ru&_x_tr_tl=uk&_x_tr_hl=ru&_x_tr_pto=wapp> ]. Дані відображено в таблиці 3.1. Інші дані цієї таблиці не потрібні.

Виходячи з креслення гвинта згідно ДСТУ 10340-80 виконання 1 було взято додаткові дані про шліц з таблиці 2 зі стандартів ДСТУ 10340-80 [<http://vsegost.com/Catalog/46/4607.shtml>] (рис. 3.2). Дані відображено в таблиці 3.1. Інші дані цієї таблиці не потрібні.

Дані з джерел [<https://meganorm-ru.translate.goog/Data2/1/4294840/4294840352.htm?_x_tr_sl=ru&_x_tr_tl=uk&_x_tr_hl=ru&_x_tr_pto=wapp> ] та [<http://vsegost.com/Catalog/46/4607.shtml>] об’єдно в одній таблиці (табл. 3.1).

Було взято дані щодо відношення діаметра різьблення до довжини гвинта дані з таблиці 2 зі стандартів ДСТУ 10340-80 [<https://meganorm-ru.translate.goog/Data2/1/4294840/4294840352.htm?_x_tr_sl=ru&_x_tr_tl=uk&_x_tr_hl=ru&_x_tr_pto=wapp> ] (табл. 3.2).

Проаналізувавши гвинт ДСТУ 10340-80 виконання 1 виявилося, що розміри фаски гвинта не вказано в стандарті ДСТУ. Дані про цю фаску було взято з джерела [<http://vsegost.com/Catalog/43/4379.shtml> ] (рис. 3.2) та дописано в таблицю 3.1.

В процесі аналізу геометричних даних зображення конфліктують між собою значення f та R1. Тому в пріоритеті було обрано значення f а R1 буде нехтуватися від стандартів ДСТУ.

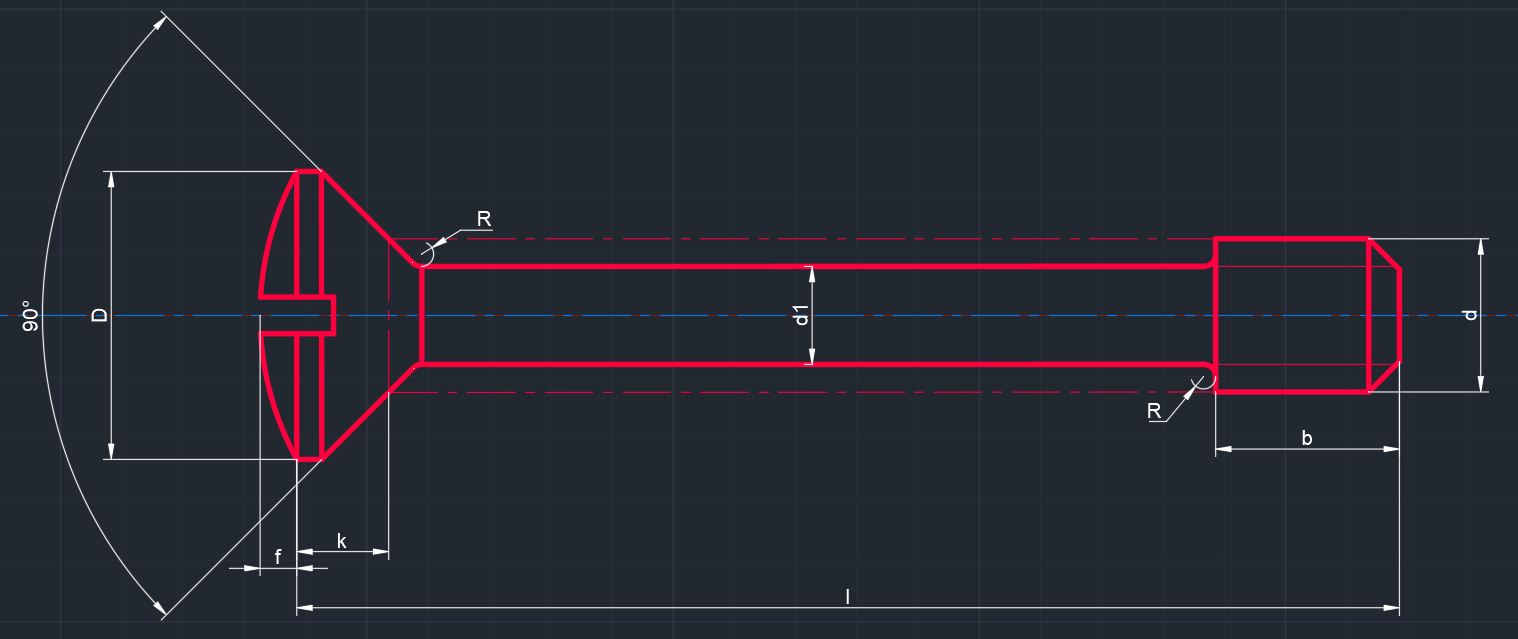


Рисунок 3.1 – Виконання 1 гвинта ДСТУ 10340-80

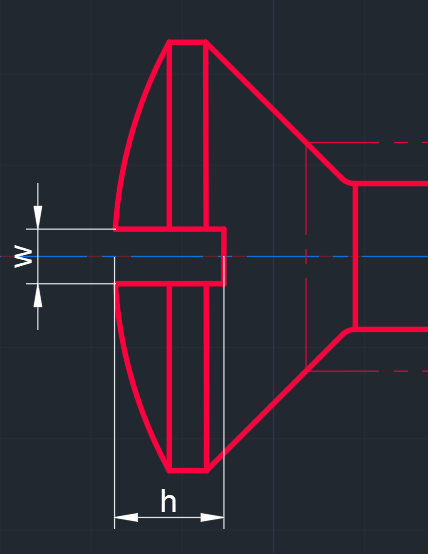


Рисунок 3.2 – Шліц гвинта ДСТУ 10340-80

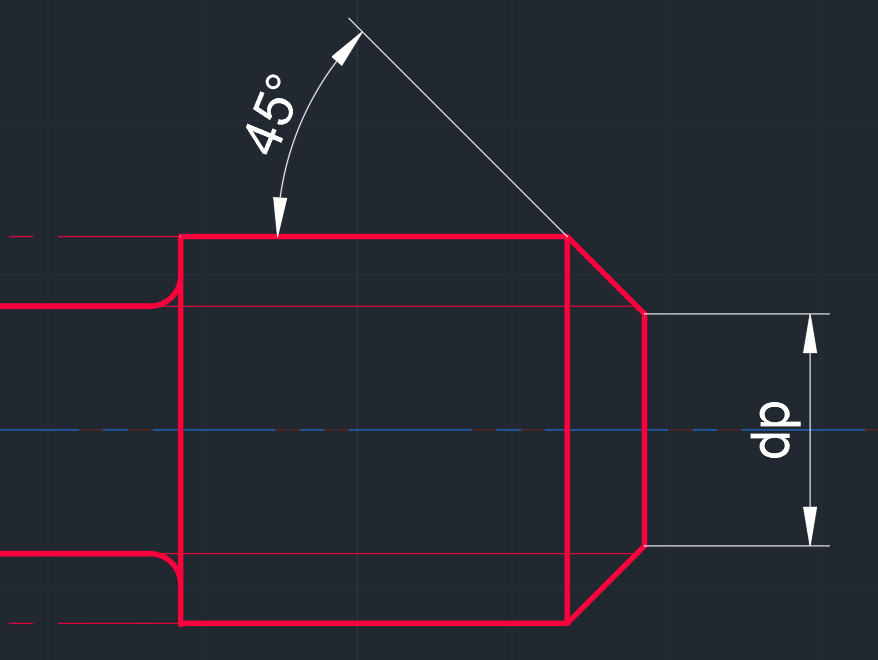


Рисунок 3.3 – Плоский кінець гвинта ДСТУ 10340-80

Таблиця 3.1 – Розміри гвинта, мм

| Діаметр різьблення *d* | | 2,5 | 3 | 4 | 5 | 6 | 8 | 10 | 12 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Ширина шліцу *w*\*1 | | 0,60 | 0,80 | 1,00 | 1,20 | 1,60 | 2,00 | 2,50 | 3,00 |
| Глибина шліцу *h*\*1 напівпотайної головки | | 1,20 | 1,45 | 1,90 | 2,30 | 2,80 | 3,70 | 4,50 | 5,40 |
| Діаметр стрижня *d* 1 | | 1,6 | 2,0 | 2,8 | 3,5 | 4,0 | 5,5 | 7,0 | 9,0 |
| Довжина різьблення *b* | | 3 | 4 | 5 | 6 | 8 | 10 | 12 | 16 |
| Діаметр головки *D* | | 4,7 | 5,6 | 7,4 | 9,2 | 11,0 | 14,5 | 18,0 | 21,5 |
| Висота головки (без сфери) *k* | | 1,50 | 1,65 | 2,20 | 2,50 | 3,00 | 4,00 | 5,00 | 6,00 |
| Висота сфери *f* | | 0,60 | 0,75 | 1,00 | 1,25 | 1,50 | 2,00 | 2,50 | 3,00 |
| Радіус під головкою *R* | | 0,2 | | | 0,4 | | 0,5 | | 0,6 |
| Радіус сфери *R* 1 | | 5,4 | 6,0 | 8,0 | 9,4 | 12,0 | 15,0 | 19,0 | 22,5 |
| Діаметр плоского кінця *dp*\*2 | | 1,5 | 2,0 | 2,5 | 3,5 | 4,0 | 5,5 | 7,0 | 8,5 |

Примітка 1. \*1 – Додаткові дані про шліц з джерела [<http://vsegost.com/Catalog/46/4607.shtml>].

Примітка 2. \*2 – Додаткові дані про плоский кінець з джерела [<http://vsegost.com/Catalog/43/4379.shtml> ].

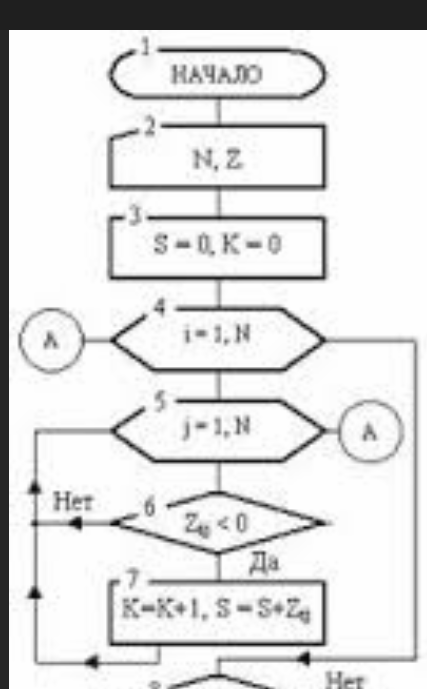
Таблиця 3.2 – Відношення діаметра різьблення до довжини гвинта

| Довжина гвинта l, мм | Діаметр різблення d, мм | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2,5 | 3 | 4 | 5 | 6 | 8 | 10 | 12 |
| 6 |  | # | - | - | - | - | - | - |
| 8 |  |  |  | - | - | - | - | - |
| 10 |  |  |  | - | - | - | - | - |
| 12 |  |  |  |  | # | - | - | - |
| (14) |  |  |  |  |  | - | - | - |
| 16 |  |  |  |  |  | - | - | - |
| (18) |  |  |  |  |  | - | - | - |
| 20 | - |  |  |  |  | - | - | - |
| (22) | - |  |  |  |  |  |  | - |
| 25 | - |  | Стандартні довжини | | |  |  | - |
| (28) | - |  |  |  |  |
| 32 | - |  |  |  |  |
| (36) | - |  |  |  |  |  |  |  |
| 40 | - |  |  |  |  |  |  |  |
| (45) | - |  |  |  |  |  |  |  |
| 50 | - |  |  |  |  |  |  |  |
| (55) | - |  |  |  |  |  |  |  |
| 60 | - |  |  |  |  |  |  |  |
| (70) | - | - | - |  |  |  |  |  |
| 80 | - | - | - |  |  |  |  |  |

Під час креслення виявилося, що деякі розміри діаметрів різьблення не відповідають довжині гвинта. А саме шліц гвинта відображається поверх ділянки з різьбленням. Ці розміри відображені символом “#” в таблиці 3.2. Це означає, що в ході роботи програми потрібно виключити комбінацію вхідних даних.

## 3.2 Схема алгоритму методики створення сервісної бібліотеки стандартних елементів. (блок-схема)

етапи для аналіз даних, аналіз розмірів гвинта (точки), реалізація коду, тестування коду, підключення в інтерфейс, оформлення записки, блок схема коду



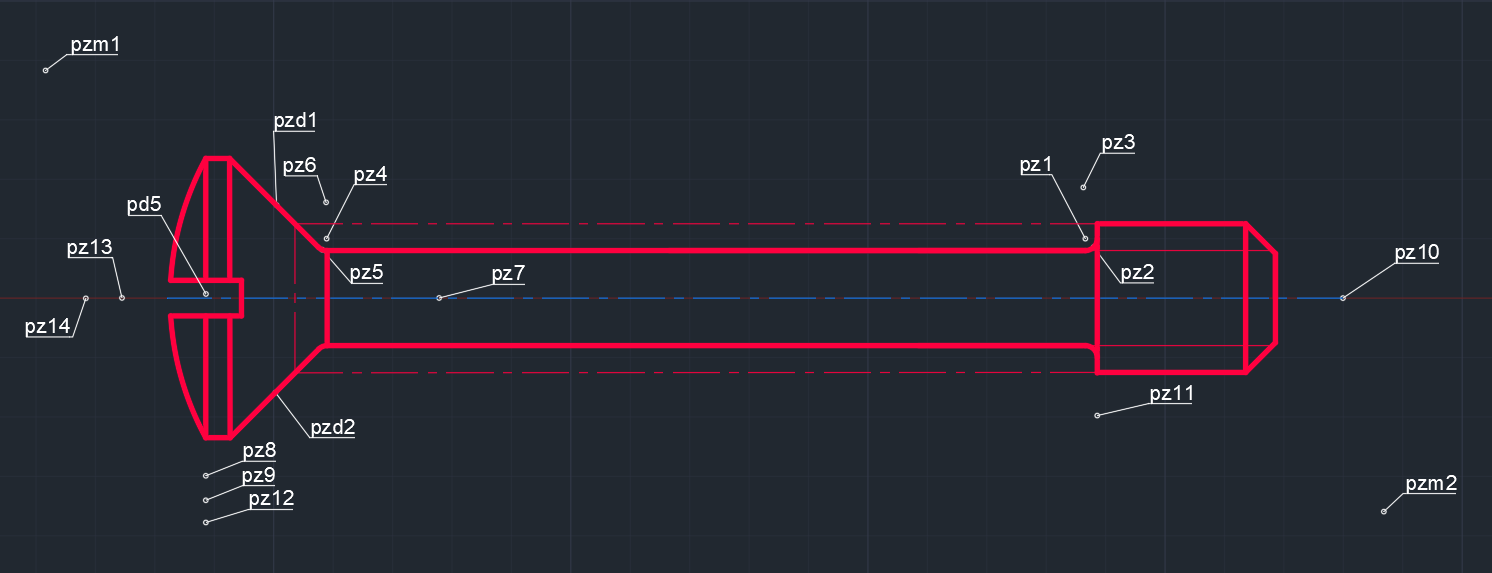
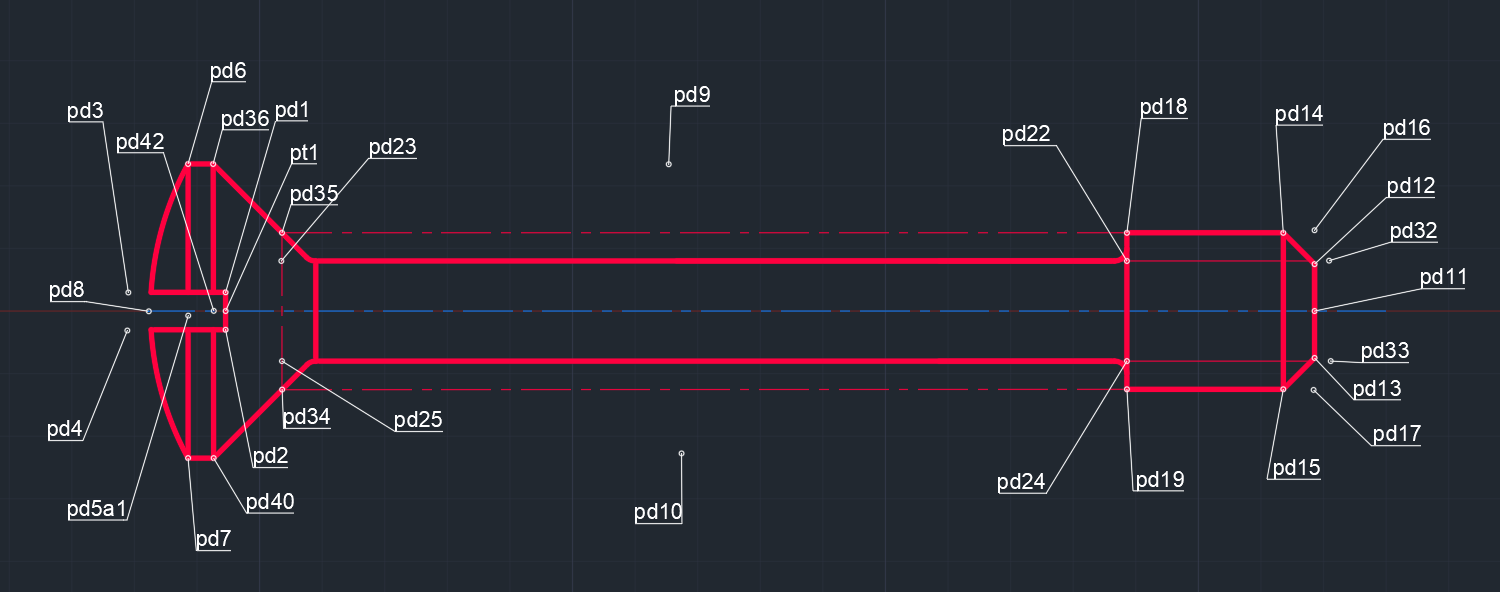
## 3.3 Розробка структури сервісного застосунку.

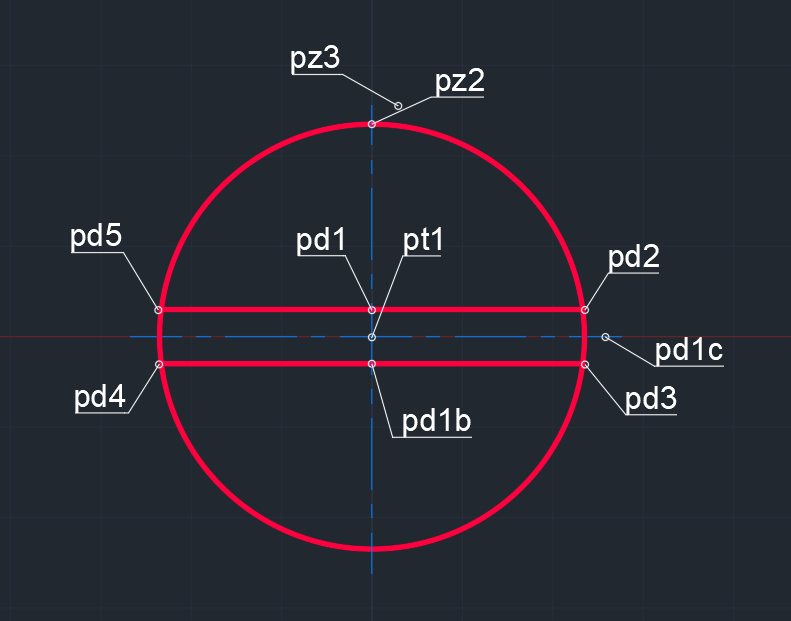
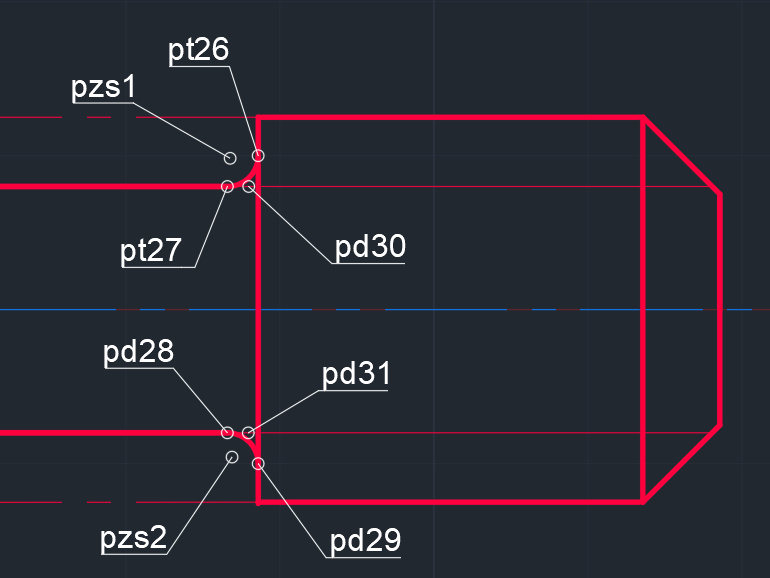
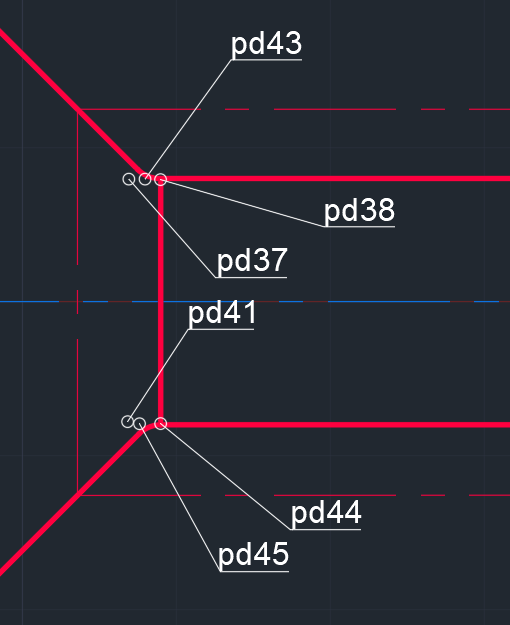
взаємодія між функцією (схемою) для середовища автокада , зробити кнопку



## 3.4 Опис особливостей програмної реалізації бібліотеки.

Для більш зручних та точних розрахунків потрібно проаналізувати початкові ескізи та виявити потрібну множину прив’язок, накреслити ескіз допоміжних точок (рис. 3.4 - ) та обрати базову точку для кожного виду ().





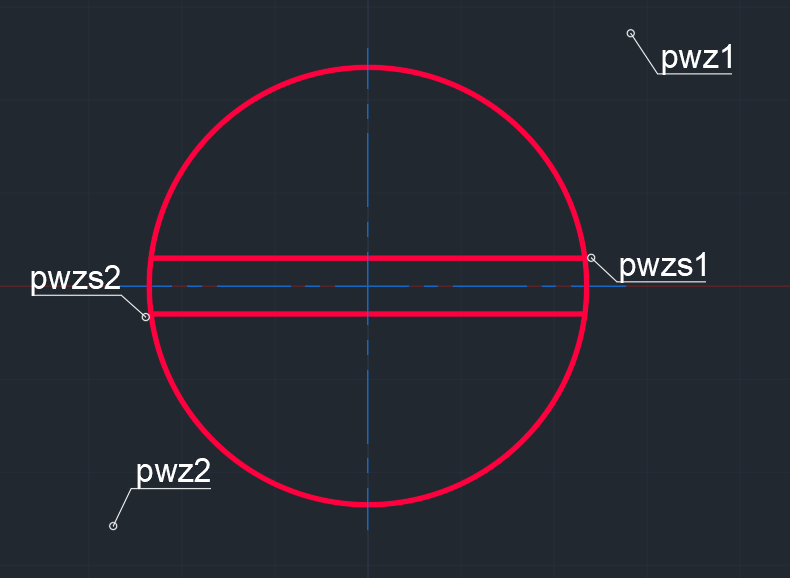
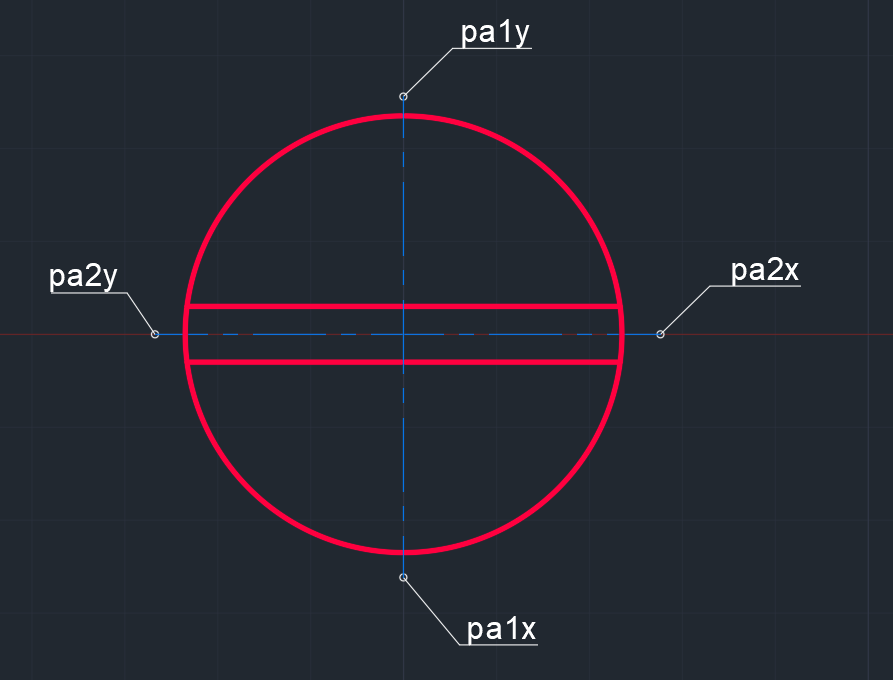


Рисунок 3.9 – Основні та допоміжні точки для збільшення кресленика бокового виду

Рисунок 3.10 – Точки для креслення вісей бокового виду

## 3.5 Порадник для користувача

юз кейси в табл

## 3.6 Порадник для програміста.

Для зміни та поправки програмної реалізації сервісної бібліотеки AutoCAD гвинт невипадаючий з напівпотайною головкою ДСТУ 10340-80 потрібно звернути увагу в програмному коді на списки під назвою *d\_other* – оголошення табличних даних для розміра гвинта, L\_all – оголошення всіх розмірів довжини гвинта, L\_unrec – оголошення не рекомендованих розмірів довжини гвинта, d\_Lb – оголошення табличних даних відношення діаметра різьблення до довжини гвинта, d\_all – оголошення всіх розмірів для діаметра гвинта. Для зміни налаштування програмного середовища та для змінення параметрів шарів потрібно звернутися до функції з назвою tune\_env. Для виправлення або дописування опорних точок, змінення креслення точок, проставлення розмірів слід звернутися до функції plot\_main – реалізація головного виду, plot\_side – реалізація виду з боку. (переробити в табл)

# ВИСНОВКИ

# ПЕРЕЛІК ПОСИЛАНЬ

# ДОДАТКИ