МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний технічний університет України
„Київський політехнічний інститут”

В.Ю.Щербина, О.С.Сахаров, О.В.Гондлях, В.І.Сівецький

САПР. Програмування на функціональній мові AutoLISP при проектуванні технологічного обладнання

Київ
НТУУ «КПІ»
2014

УДК 76.004.9 + 66.041
ББК 32.973

Рекомендовано Вченою радою ІХФ НТУУ «КПІ»
(протокол №9 від 20.08.2014р.)

Рецензенти: *І.І.Назаренко*, д-р. техн. наук, проф. (КНУБА)
А.*Д.Петухов*, д-р. техн. наук, проф. (НТУУ «КПІ»)

Щ11 САПР. Програмування на функціональній мові AutoLISP при проектуванні технологічного обладнання /В.Ю.Щербина, О.С.Сахаров, О.В.Гондлях, В.І.Сівецький. – К.: НТУУ «КПІ», 2014. – 156с.: іл.
ISBN 966-8440-27-8

Розглянуті основи програмування на функціональній мові AutoLISP та використання функцій VisualLISP в графічній системі AutoCAD. Наводяться приклади виконання робіт по створенню програм для конструкторського та технологічного обладнання.

Розглянуті інструментальні засоби створення графічного інтерфейсу користувача, для розробки систем і підсистем автоматизованого проектування. Приведено приклади використання команд та функцій і завдання для самостійної роботи.

Для студентів технічних спеціальностей вищих закладів освіти.

УДК 76.004.9 + 66.041
ББК 32.973

Щербина	Валерій	Юрійович
Сахаров	Олександр	Сергійович
Гондлях	Олександр	Володимирович
Сівецький	Володимир	Іванович

САПР. Програмування на функціональній мові AutoLISP при проектуванні технологічного обладнання

В авторській редакції
ISBN 966-8440-27-8

ЗМІСТ

ВСТУП

Використання в системі AutoCAD мови AutoLISP не тільки значно прискорює процес розроблення проектної документації, але і дозволяє створювати нові команди графічного редактора, спеціалізовані меню в середовищі AutoCAD, здійснювати доступ до графічної бази даних і модернізувати її, розробляти функції для розв’язання найрізноманітніших задач і, крім того, створювати ефективні системи та підсистеми, пов'язані з обробкою інформації, поданої у вигляді символів і чисел. Форми подання програми й даних, отриманих нею у AutoLISP, однакові. Програми можуть обробляти, а також перетворювати інші програми і навіть самих себе, що дозволяє ефективно використовувати LISP для розв’язання широкого кола завдань.

AutoLISP – це створений спеціально для AutoCAD діалект LISP, отриманий у результаті зміни мови CommonLISP.

Мова LISP, розроблена 1961 року американським ученим Дж. Маккарті, є родоначальником функціональних мов. AutoLISP належить до так званих *функціональних мов*, тобто до мов, в основу яких покладено поняття функції (на відміну від звичайних операторних мов програмування – FORTRAN, PASCAL, C та ін.). До таких мов відносять також мову PROLOG. Усі обчислення, перетворення і керування програми у функціональних мовах здійснюються за допомогою елементарних (вбудованих) функцій, обумовлених програмістом під час написання програми. Програма в цілому є суперпозицією деяких функцій і, у свою чергу, може бути використана як функція іншими програмами.

Мова LISP ідеально підійшла для AutoCAD, “передавши” AutoLISP свої дуже зручні засоби роботи з глибоко структурованою інформацією. AutoLISP – це

засіб, що дозволяє програмно працювати з об'єктами AutoCAD, довідковими таблицями, зчитувати і записувати файли з AutoCAD. AutoLISP можна вважати прямим вікном усередину AutoCAD. Крім того, AutoLISP дуже простий у вивченні і дуже гнучкий.

Усе, чи майже все, що ми робимо в середовищі AutoCAD вручну, можна реалізувати програмно. Для цього в AutoCAD передбачено кілька механізмів і засобів. Це, по-перше, засіб створення сценаріїв і, по-друге, використання мов програмування. В AutoCAD мовою є AutoLISP. Оскільки він зарекомендував себе в попередніх версіях як зручний і практичний засіб програмування, то AutoLISP з мовою C++ збережеться й у подальших версіях. Більше того, перш ніж приступити до розроблення додатків мовою C++, рекомендується ознайомитися з принципами та методами створення програм мовою AutoLISP.

Як частина AutoCAD, AutoLISP дозволяє оперувати зі змінними різних типів і передавати значення командам AutoCAD під час уведення даних. Відповідаючи на запити команд AutoCAD, можна також використовувати функції AutoLISP, у яких можуть виконуватися різні арифметичні й умовні операції над числовими значеннями і значеннями визначених змінних.

Крім засобів виконання обчислень, AutoLISP містить функції, що надають програмам доступ до графічної бази даних поточного креслення. AutoLISP дозволяє також керувати графічним редактором AutoCAD і звертатися до його власних команд. За допомогою програм AutoLISP можна створювати доповнення, спрямовані на конкретну предметну область застосування функції, що поєднують у собі запити до користувача (діалог), можливість вибору за умовою декількох варіантів використання значень за замовчуванням. Хоча макровизначення, які створюють під час написання меню AutoCAD, можуть бути досить складними та потужними, без AutoLISP вони являють собою лише комбінації стандартних команд AutoCAD. Включенням у макровизначення меню функцій AutoLISP можна перетворити меню AutoCAD на інтелектуальний засіб автоматизації проектування.

Отже, AutoLISP дозволяє:

- використовувати змінні та вирази, відповідаючи на запити команд AutoCAD;
- читати і писати зовнішні файли, у такий спосіб здійснюючи обмін із зовнішніми програмами, які можна запускати з AutoCAD;
- створювати різні функції та нові команди AutoCAD, розширюючи в такий спосіб графічні можливості усієї системи;
- здійснювати програмний доступ (зчитування) до даних, які є об'єктами рисунка, а також до таблиць AutoCAD, у яких зберігається інформація про блоки, шари, види, стилі і типи ліній;
- здійснювати програмне керування графічним екраном AutoCAD і введенням/виведенням з різних пристроїв.

1. AUTOLISP – БАЗОВА МОВА ПРОГРАМУВАННЯ В AUTOCAD

1.1 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.

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 дво- 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 *X*-координатою, 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.

1.2 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 *A*.

(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.

(getdist [<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 dist (getdist '(1.0 3.5) "Як далеко ?"))

Функція дозволяє ввести з клавіатури ціле число.

(getkword [<підказка >]

За допомогою цієї функції користувач має можливість увести ключове слово. Список ключових слів слід попередньо скласти за допомогою функції *INITGET*.

GETKWORD повертає введений рядок, якщо він збігається з ключовим словом. Якщо ж рядок не збігається ні з одним із ключових слів, то AutoCAD повертає запит. У результаті введення порожнього рядка функція повертає *nil* (якщо таку можливість передбачено завчасно).

Приклад: (initget 1 "Так Ні")
(setq**Ошибка! Закладка не определена.** X (getkword "ви впевнені ? (Так Ні)"))

(getorient [<точка>] [<підказка>])

В AutoLISP кути задають у радіанах. Віссю відліку кутів є горизонталь (напрямок – зліва направо), а сам відлік ведуть проти годинникової стрілки. *GETORIENT* аналогічна функції *GETANGLE*, але на відміну від останньої, залежить від того, як ведуть відлік. Тому *GETANGLE* рекомендується використовувати для відносних, а *GETORIENT* – для отримання абсолютних значень кутів.

(getpoint [<точка>] [<підказка>])

Функція дозволяє ввести точку з клавіатури. Якщо перший аргумент задано, то на екрані з'являється гумова нитка, яка зв'язує цю точку з поточним положенням курсора.

(getreal [<підказка>])

Функція зчитує з клавіатури дійсне число і видає його. Якщо задано аргумент, то в нижньому рядку дисплея виводиться "підказка".

(getstring [<знак>] [<підказка>])

Функція приймає з клавіатури рядок. Якщо “знак” заданий і не *nil*, то всередині рядка, який вводять, можуть міститися пробіли (функція обов'язково має завершуватись <ENTER>).

Приклад: (setq s (getstring *T* "Ваше прізвище, ім'я та по батькові? "))

(print1 <вираз> [<дескриптор файлу>])

Функція виводить вираз “вираз” на екран і повертає цей самий “вираз”. Першим аргументом може бути будь-який вираз; не обов'язково, щоб це був рядок. Якщо “дескриптор файлу” наявний (і є дескриптором відкритого для запису файлу), то “вираз” записують у файл у тому самому вигляді, в якому б він з'явився на екрані. Друкується тільки заданий вираз; ніяких нових рядків чи пробілів не додається.

Приклад: якщо (setq a 123) ,

то: (print1 'a) друкує *A* і повертає *A*
(print1 a) друкує 123 і повертає 123

Кожен запис попередніх прикладів виводиться на екран, оскільки не задавався ніякий дескриптор файлу. Якщо *F* є допустимим дескриптором файлу, відкритого для запису, то можливий запис:

(print1 a f)

(princ <вираз> [<дескриптор файлу>]

Функція подібна до функції *PRINT*1, за винятком того, що символи керуванння у “виразі” друкують без редагування. Взагалі, *PRINT*1 призначена для того, щоб друкувати вирази зручними для користувача, а *PRINC* – для запису у файл, призначеного для зчитування такими функціями, як, наприклад, *READ-LINE* .

(print <вираз> [<дескриптор файлу>])

Ця функція аналогічна функції *PRINT1*, однак перед “виразом” виводиться новий рядок, після чого – пробіл.

(terpri)

Зазначена функція виводить на екран порожній рядок. Вона повертає код символу “переведення рядка”. Функцію *TERPRI* не використовують для файлів уведення/виведення. Щоб увести “переведення рядка” у файл, слід користуватися функціями *PRINT* або *PRINC*.

(prompt <повідомлення>)

Функція відображає “повідомлення” в екранній області підказок і повертає *nil*. Аргумент “повідомлення” є рядком. У системах з двома екранами *PROMPT* відображає повідомлення на обох екранах і тому використовується частіше, ніж *PRINC*.

Приклад: (prompt "New value: ") відображає "New value: " і повертає *nil*.

1.3 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

(sin <кут>)

Функція повертає синус “кута”.

Приклад: (sin 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 *e*.
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

1.4 Робота зі списками

(car <список>)
Функція повертає перший елемент списку. Якщо список порожній, то повертається *nil*.

Приклад: (car '(a b c)) результат *a*

(cdr <список>)
Функція повертає список, з якого спочатку вилучають перший елемент. Якщо список порожній, то повертається *nil*.

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

(list <вираз>....)

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

Приклад: (list '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 списочного 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 *nil*.

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

(assioc <елемент> <a_список>)

Функція шукає у списку “a_список” підсписок, який має “елемент”. Якщо такий підсписок не знайдено, то видається *nil*.

Приклад: нехай список *AL* визначено так:

((name box) (width 3) (size 4.72) (depth 5)) ,

тоді (assoc 'size al) результат (size 4.72)
(assoc 'weig al) результат *nil*

(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 *nil*.

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

(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 неперероблениm.

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 '((first join) (mid q) (last public))) ,
то (setq old
(assoc 'first who)
) повертає (FIRST JOHN)
(setq new '(first 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"

1.5 Логічні функції

(= <число> <число>...)

Функція виконує перевірку ”на дорівнює” і повертає *T*, якщо всі аргументи чисельно дорівнюють один одному, у противному разі – повертає *nil*. Функція порівнює і текстові рядки.

Приклад: (= 20 388) результат *nil*
(= 2.4 2.4 2.4) результат *T*

(/= <число 1> <число 2>)

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

Приклад: (/= 10 20) результат *T*
(/= 5.43 5.43) результат *nil*

(< <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 *T*.

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

(~ <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 *nil*, то функція повертає *nil*, у протилежному випадку – повертає *T*.

Приклад: (setq a 103)
(setq b nil)
(setq c "string") ,
тоді (and 1.4 a b) повертає *T*
(and 1.4 a b c) повертає *nil*

(not <елемент>)

Функція повертає *T*, якщо значенням “елемента” є *nil*, і *nil* – 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 c nil) ,
тоді (not a) результат *nil*
(not c) результат *T*

(or <вираз> ...)

Функція повертає результат логічного додавання списку виразів. Якщо всі вирази мають значення *nil*, то *OR* повертає *nil*, у протилежному випадку – повертає *T*.

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

(eq <вираз1> <вираз2>)

Функція з'ясовує еквівалентні вирази 1 і 2. *EQ* повертає *T*, якщо вирази дорівнюють один одному і *nil* – у протилежному випадку.

Приклад: (setq f1 '(a b c))
(setq f2 '(a b c))
(setq f3 f2)
тоді (eq f1 f3) результат *nil* (f1 і f3 не той самий список)
(eq f3 f2) результат T (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 T (f1 і f3 є однаковими списками)
(eq f2 f2) результат *T* (f1 і f3 один і той самий список)

1.6 Функції для розгалуження програм

(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 *nil*, 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 nil))

Функція перевіряє склад символу *s*. Якщо значення дорівнює *Y* або *y*, то виробляється 1, якщо *n* або *N*, – то 0, в інших випадках – *nil*.

(if <умова> <вираз1> [<вираз2>])

Функція визначає, виконується умова чи ні. Якщо "умова" не дорівнює *nil*, то обчислюється "вираз1", інакше – "вираз2". "Вираз2" не є обов'язковим. *IF* повертає значення вибраного виразу.

Приклад: (if (= 1 3) "ТАК" "НІ") результат "НІ"
(if (= 2 (+ 1 3) "ТАК") результат *nil*

(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: (if (= a b) (progn (setq a (+ a 10) b (- b 10))))

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

1.7 Функції організації циклів

(foreach <ім'я> <список> <вираз>...)

Функція здійснює послідовне підставлення елементів списку замість імені і обчислює вказані вирази, кількість яких може бути довільною. *FOREACH* повертає результат обчислення останнього виразу.

Приклад: (foreach n '(a b c) (print n))
еквівалентно: (print a) (print b) (print c)

(repeat <число> <вираз> ...)

Функція обчислює кожний аргумент “вираз” стільки разів, скільки вказується аргументом “число”, і повертає значення останнього виразу.

Приклад: (setq b 100) ,
тоді (repeat 4
(setq b (+b 10))) повертає 140

(while <перевірка> <вираз> ...)

Функція обчислює аргумент “перевірка” і, якщо значення його не дорівнює *nil*, обчислює інші “вирази”, а потім знову “перевірку”. Це триває доти, доки значення “перевірки” не буде дорівнювати *nil*. Функція *WHILE* повертає останнє значення останнього аргументу “вираз”.

Приклад, якщо (setq a 1) ,
то (while (<= a 10)
(some-func a)
(setq a (1+ a))) результат: 11

1.8 Виклик програм з AutoCAD

(command <аргументи>...)

Функція здійснює запуск команд AutoCAD з AutoLISP.

Приклад: (setq pt1 '(1.45 3.21))
(setq pt2 (getpoint "Enter a point"))
(command "_line" pt1 pt2 "")

Перші дві функції встановлюють дві точки pt1 і pt2, потім викликається команда *LINE* і ці точки використовуються як відповіді на її підказки. Символ "" еквівалентний введенню пробілу з клавіатури.

(vl-cmdf [<параметр1> [<параметр2> ... [<параметри>] ... J])

Аргументами можуть бути будь-які вирази, які потрібно передати в командний рядок AutoCAD (у тому числі команди, опції команд, дані, вирази LISP і т.д.).

На відміну від функції command, як параметри функції vl-cmdf можуть фігурувати вираження LISP, що містять функції інтерактивного введення (getint та інші). Значення, що функція повертає - T, якщо всі вирази, передані в командний рядок, виконалися без збоїв, і nil - при виявленні помилок.

Основна відмінність функції vl-cmdf від функції command полягає в тім, що попередньо виконується перевірка команд. І якщо виявляється якась помилка, те жоден з параметрів функції vi-cmdf не виконується системою AutoCAD, і генерується повідомлення про помилку додатка. Це виключає можливість часткової роботи будь-яких команд AutoCAD (тобто коли виконуються перші операції, а потім команда видає збій).

Приклад: (setq pt1 '(1.45 3.21))
(setq pt2 (getpoint "Enter a point"))
(vl-cmdf "_line" pt1 pt2 "")

1.9 Функції перетворення рядків і перевірки типів даних

(atof <рядок>)

Функція перетворює рядок на дійсне число.

Приклад: (atof "97.1") результат 97.10000

(atoi <рядок>)

Функція перетворює рядок на ціле число.

Приклад: (atoi "97") результат 97

(atom <елемент>)

Функція повертає *nil*, якщо елемент є списком, і *T* – у протилежному випадку.

Нехай *a* і *b* встановлено так:

(setq a '(x y z)) ,

тоді (atom 'a) результат *T*
(atom a) результат *nil*

(angtos <кут> [<режим> [<точність>]])

Функція редагує перший аргумент, який є кутом у радіанах. Другий аргумент (“режим”) є цілим числом і визначає тип редагування (табл. 1).

Таблиця 1

	Режим	Тип редагування
ANGTOS	0	Градуси
	1	Градуси/хвилини/секунди
	2	Відсотки
	3	Радіани

Аргумент “точність” є цілим числом, за допомогою якого задають кількість знаків після десяткової крапки. Аргументи “режим” і “точність” відповідають змінним *AUNITS* і *AUPREC*.

(ascii <рядок>)

Функція повертає код першого символу “рядка” за таблицею *ASCII*.

Приклад: (ascii "a") результат 65

(boundp <атом>)

Функція повертає *T*, якщо “атом” використовувався в операції присвоювання і його значенням не є *nil*.

Нехай (setq a 2) і (setq b nil) ,
тоді (boundp 'a) результат *T*
(boundp 'b) результат *nil*

(chr <число>)

Функція повертає результат перетворення цілого числа на рядок, який складається з одного *ASCII*-символу.

Приклад: (chr 65) результат "A"

(fix <число>)

Функція повертає результат перетворення "числа" до цілого типу.

Приклад: (fix 3.7) результат 3.000000

(float <число>)

Функція повертає результат перетворення "числа" до дійсного типу.

Приклад: (float 3.7) результат 3.700000

(itoa <ціле>)

Функція повертає результат перетворення “цілого” на рядковий формат.

Приклад: (itoa 33) результат "33"

(listp <елемент>)

Функція повертає *T*, якщо аргумент є списком і *nil* – у протилежному випадку.

Приклад: (listp '(a b c)) результат *T*
(listp '(a)) результат *nil*

(max <число> <число>...)

Функція повертає максимальне з "чисел"-аргументів.

Приклад: (max – 88 19 5 2) результат 19

(min <число> <число>...)

Функція повертає мінімальне із заданих "чисел".

Приклад: (min 638 – 10.0) результат –10.000000

(minusp <елемент>)

Функція повертає *T*, якщо "елемент" є від'ємним числом, у противному разі – *nil*.

(numberp <елемент>)

Функція повертає *T*, якщо “елемент” є дійсним або цілим числом, і *nil* – у протилежному випадку.

Приклад: якщо (setq a 123)
(setq b 'a) ,

то:

(numberp 'a)	повертає	*nil*
(numberp a)	повертає	*T*
(numberp b)	повертає	*nil*
(numberp (eval b))	повертає	*T*

(null <елемент)

Функція повертає *T*, якщо аргумент “елемент” має значення *nil*, і *nil* – у протилежному випадку.

Приклад: (setq a 123) ,
тоді (null a) повертає *nil*
(setq c nil)
(null c) повертає *T*

(rtos <число> [<режим> [<точність>]])

Функція повертає рядок, який складається з аргументу “число” відповідно до значень аргументів “режим”, “точність” і системної змінної *DIMZIN* системи AutoCAD.

Допустимі значення аргументу “режим” такі (табл. 2):

Таблиця 2

Режим	Формат редагування
1	Науковий
2	Десятковий
3	Технічний (фути і десяткові частки дюймів)
4	Архітектурний (фути і дробові дюйми)
5	Вільні дробові одиниці

Приклад: (rtos 17.5 1 4) повертає "1.7500E+01"
(rtos 17.5 2 2) повертає "17.50"
(rtos 17.5 3 2) повертає "1'-5.50""
(rtos 17.5 5 2) повертає "17 1/2"

(strcase <рядок> [<регістр>])

Функція *STRCASE* бере рядок, заданий аргументом “рядок”, і повертає копію з усіма літерними символами, перетвореними на верхній або нижній регістр, залежно від значення другого аргументу “регістр”.

Приклад: (strcase "Sample") повертає "SAMPLE".

(strlen <рядок>)

Функція повертає довжину рядка (кількість символів).

Приклад: (strlen "abcd") повертає 4

(trans <точка> <з> <в> [<вектор>]

Дана функція перетворить координати точки (або величину переміщення) з однієї системи координат в іншу. Аргументи

<точка> - список трьох дійсних чисел, які можна інтерпретувати або як тривимірну точку, або як тривимірне переміщення (вектор).

<з> - код системи координат у якій знаходиться зазначена <точка>,

<в> - код системи координат, у якій відбувається перетворення координат точки. Якщо є присутнім факультативний аргумент

<вектор> і його значення не дорівнює нулю, то аргумент <точка> буде трактуватися як тривимірне переміщення.

Аргументи <з> й <в> можуть мати значення зазначені в таблиці:

Код	Система координат
0	Світова (WCS)
1	Користувальницька (поточна UCS)
2	Екранна (DCS поточного видового екрана)

Приклад: Нехай UCS повернений на кут 90 градусів проти годинникової стрілки навколо осі Z відносно WCS.

(trans ' (1.0 2.0 3.0) 0 1) повертає (2.0 -1.0 3.0)
(trans ' (1.0 2.0 3.0) 1 0) повертає (-2.0 1.0 3.0)

(type <елемент>)

Функція повертає тип аргументу <item>, де тип має одне з таких значень:
REAL – числа з плаваючою крапкою;
FILE – дескриптори файлів;
STR – рядки;
INT – цілі числа;
SYM – символи;
LIST – списки (і користувацькі функції);
SUBR – внутрішні функції AutoLISP;
PICKSET – варіанти виборів AutoCAD;
ENAME – імена примітивів AutoCAD;
PAGETB – таблиця розподілу функцій на сторінки.

Приклад: (setq a 123 s "Hello!") ,
тоді: (type a) повертає *INT*
(type s) повертає *STR*

Наступний приклад показує, як використовувати функцію *TYPE*.

```
(defun  isint (a)
(if  (= (type  a) 'INT)   ; ціле число?
T                         ; так, повертає T
nil                       ; ні, повертає nil
)
```

(zerop <елемент>)

Функція повертає *T*, якщо аргумент “елемент” є цілим або дійсним числом, що дорівнює нулю, у протилежному випадку – повертає *nil*. Вона не визначається для інших типів “елементів”.

Приклад: (zerop 0.0) повертає *T*
(zerop 0.00011) повертає *nil*

1.10 Функції для роботи з геометричним описом об'єктів

(angle <pt1> <pt2>)

Функція обчислює кут між двома точками “pt1” і “pt2” двовимірного простору (точку подано у вигляді двохелементного списку).

Приклад: (angle '(1.0 1.0) '(1.0 4.0)) результат 1.570796

(distance <точка1> <точка2>)

Функція повертає відстань між двома точками.

Приклад: (distance '(1.0 2.5) '(7.7 2.5)) результат 6.7

(inters <точка1> <точка1> <точка3> <точка4> [<>])

Функція визначає точку перетину двох ліній. "Точка1" і "точка2" є кінцями відрізка першої лінії, а “точка3” і “точка4” – другої лінії.

Приклад: (setq a '(1.0 1.0) b '(9.0 9.0))
(setq c '(1.0 1.0) d '(9.0 9.0)) ,
тоді (insert a b c d) результат *nil*
(insert a b c d nil) результат (4.000000 4.000000)

(osnap <точка> <режим>)

Функція повертає точку, яка є результатом використання режимів фіксації об'єктів, що задаються аргументом “режим” для точки “точка”. Аргумент “режим” є рядком, що складається з одного або більше найменувань режимів фіксації об'єктів, таких як "midpoint" (середина), "center" (центр) тощо. Для розподілу назв режимів використовують кому.

Приклад: (setq pt2 (osnap pt1 "midp"))

(polar <точка> <кут> <відстань>)

Функція повертає точку, яка знаходиться на заданій відстані “відстань” від точки “точка”, і ту, яка лежить на промені, що складає кут “кут” з базовим напрямом. Точка є списком двох дійсних чисел, а кут виражається в радіанах.

Приклад: (polar '(1.0 1.0) 0.785398 1.414214) повертає (2.0 2.0)

(trace <функція> ...)

Функція є засобом налагодження. Вона встановлює прапорець трасування для заданих функцій і повертає останнє ім'я функції. Кожного разу, коли функція обчислює, на екрані відображається її ім'я і рівень входження (рівень глибини виклику), результат функції виводиться на друк.

Приклад: (trace my-func) повертає *MY-FUNC* і встановлює режим трасування для функції *MY-FUNC*. Режим скасовують під час використання функції (UNTRACE <функція> ...).

1.11 Функції керування зображенням

(graphscr)

Функція переключає екран AutoCAD з текстового режиму в графічний. *GRAPHSCR* завжди повертає *nil*.

(textscr)

Функцію використовують для перемикання з екрана графічного зображення на текстовий екран у разі одноекранних систем. Вона завжди повертає *nil* (див. також функцію *GRAPHSCR*).

(redraw [<примітив> [<режим>]])

Функція здійснює повний контроль над перекреслюванням “примітив”, де “примітив” – ім'я перекреслюваного примітиву, а “режим” – ціле число, яке має одне з таких значень (табл. 3).

Таблиця 3

Режим	Дія
1	Перекреслити примітив на екрані

2	Не перекреслювати (погасити)
3	Виділити примітив більш яскравим світлом
4	Відмінити завищену яскравість примітиву

Якщо аргумент “примітив” є заголовком складного примітиву (полілінії або блоку з атрибутами), і якщо аргумент “режим” позитивний, то оброблятися буде основний примітив і його підпримітиви. Якщо ж від'ємний, то функцією буде оброблятися тільки заголовний примітив. Функція *REDRAW* завжди повертає *nil*, якщо не виявлено ніяких помилок.

1.12 Завантаження, створення функцій та оброблення помилок

(load <ім'я_файлу>)

Функція завантажує файл, складений з виразів AutoLISP, та обчислює ці вирази. "Ім'я_файлу" – це ім'я файлу в системі. Розширення *LSP* друкувати не треба. Якщо файл знаходиться у другому каталозі, то замість імені можна задати весь шлях.

Приклад: "function/test1".

Якщо операція завантаження пройшла вдало, то обчислюється ім'я останньої функції з файлу. Якщо ні, то повертається ім'я файлу. Нехай, наприклад, у файлі "/fred/test1.lsp задано *DEFUN*, яка визначає функцію *MY-FUNK*, а файл test2 не існує.

Приклад: (load "/fred/test1") результат my-func
(load "test2") результат "test2"

(defun <символ> <список аргументів> <вираз>...)

DEFUN визначає функцію з іменем “символ”. Ім'я символу можна не брати в лапки, оскільки інтерпретатор робить це сам. За іменем функції йде список її аргументів (можливо, порожній). За цим списком можуть розміщуватися локальні параметри функції, які відокремлюють від списку похилою рискою (з обох боків від похилої риски має бути по одному пробілу). Якщо списку аргументів і параметрів немає, то дужки все одно слід ставити.

Приклад: (defun myfunc (x y)...) визначення функції з двома аргументами
(defun myfunc (/ x y)...) визначення функції з двома локальними символами.

За списком аргументів і локальних параметрів (символів) записуються вирази, які складають тіло функції.

DEFUN повертає значення функції, котру вона визначає.

Коли визначена таким чином функція починає виконуватись, спочатку обчислюються значення її аргументів. Указані локальні символи будуть використовуватися тільки в тілі цієї функції, не розповсюджуючись на інші. Функція повертає результат обчислень останнього виразу.

Приклад: (defun add10 (x) (+ 10 x))
результат add10
add10 5 результат 15
add10 -7.4 результат 2.600000

Якщо ім’я функції *C:XXX*, то можна розширити AutoCAD, додавши до нього нові функції за допомогою конструкції *DEFUN*. Такі функції задовольняють вимоги:

1. Ім’я функції повинно мати формат *C:XXX*. XXX – ім'я нової команди, яке не може збігатися з уже існуючим іменем.
2. Список аргументів команди має бути порожнім, але визначення локальних символів дозволяється.

Функцію, котру визначено як команду, можна викликати просто введенням *XXX* у відповідь на підказку AutoCAD Command:

(*error* <рядок>)

Цю, визначену користувачем функцію, використовують для роботи над помилками. Якщо значення її не дорівнює *nil*, то кожного разу, коли утворюється помилковий стан AutoLISP, вона виконується як функція. Функція передає один аргумент, рядок, який уміщує опис помилки.

Приклад: (defun *error* (msg) (princ "error: ") (princ msg) (terpri))

Функція виконує те саме, що і стандартний маніпулятор помилок AutoLISP; видає на екран дисплея "error:" і опис.

1.13 Функції файлових входів/виходів

(findfile <файл>)

Функція шукає файл по імені (короткому або повному). Якщо аргумент містить ім'я файлу без повного шляху, то пошук виконується у робочому каталозі і по стандартних для Autocad шляхам пошуку. Повертає повне ім'я файлу або nil, якщо файлу немає.

Приклад: (findfile "data.txt")

(getfiled <заголовок> <ім'я > <расширення> <прапорець>)

Функція Викликає діалогове вікно пошуку файлу.

Заголовок – назва діалогового вікна

Ім'я – ім'я файлу або теки, з якою починається пошук

Розширення – розширення файлу. (“” замінюється на “*”)

Прапорець – опція функції яка має значення

1 – при створенні нового файлу (не можна використовувати для вибору існуючого файлу)

4 – дозволяє вводити ім'я файлу з будь-яким розширенням (або без нього)

16 – аргумент ім'я трактується як ім'я каталогу, в якій треба шукати файл

Приклад: (setq a (open "file.ext" "r"))

(getfiled “Виберіть файл” “E:/stud” “doc” 16) вибирається каталог для пошуку файлу

(getfiled “Збережіть файл” “E:/stud” “doc” 17) вибирається каталог для збереження файлу без вказівки імені файлу в полі діалогу

(getfiled “Збережіть файл” “E:/stud/rez” “doc” 1) вибирається каталог для збереження файлу з вказівкою імені файлу в полі діалогу

(open <ім'я файлу> <режим>)

Функція відкриває файл для забезпечення доступу за допомогою функцій введення/виведенння AutoLISP. Вона повертає дескриптор файлу, який має використовуватись іншими функціями введення/виведення; тому результат виконання даної функції присвоюється символу.

Приклад: (setq a (open "file.ext" "r"))

Допустимі режими описано в табл. 4.

Таблиця 4

Режим	Опис
"*r*"	Відкритий для зчитування. Якщо “ім'я файлу” не існує, то повертається *nil*
"*w*"	Відкритий для запису. Якщо файл не існує, то створюється і відкривається новий. Якщо існує, то його дані будуть втрачені
"*a*"	Відкритий для доповнення. Якщо файл не існує, то створюється і відкривається новий файл. Якщо існує, то він відкривається і нові дані розміщують наступними за тими, які зберігаються у ньому. Отже, реалізується доповнення даних у файл

“Ім'я файлу” може містити префікс директорії.

Приклад: (setq f (open "/x/new.tst" "w")) повертає дескриптор файлу

(close <дескриптор>)

Функція закриває файл і виробляє *nil*. Нехай *X* є дескриптором відкритого файлу, тоді

(close X) виробляє *nil* і закриває файл .

(read-line [<дескриптор файлу>])

Функція зчитує рядок, який уведено з клавіатури або з відкритого файлу, заданого своїм дескриптором “дескриптор файлу”. Якщо трапляється кінець файлу, то *READ-LINE* повертає *nil*, у противному разі – повертає зчитаний рядок.

(read <рядок>)

Функція повертає перший список або “атом”, одержаний з рядка. У рядку не має бути пробілів.

Приклад: (read "hello") повертає *HELLO*

(read-char [<дескриптор файлу>])

Функція зчитує один символ із вводного буфера клавіатури або з відкритого файлу, на який вказує аргумент “дескриптор файлу”. Вона повертає ціле число в коді *ASCII*, яке є відображенням зчитаного символу.

(write-char <число> [<дескриптор файлу>])

Функція записує один символ на екран або у відкритий файл, заданий своїм дескриптором “дескриптор файлу”.

Приклад: (rite-char 67 f) повертає 67 і записує літеру *C* у файл.

(write-line <рядок> [<дескриптор файлу>])

Функція відображає аргумент “рядок” на екрані або записує у відкритий файл, який визначає аргумент “дескриптор файлу”. Він повертає взятий у лапки аргумент “рядок”, але опускає лапки, коли здійснює запис у файл.

Приклад: (write-line "Test" f) записує *Test* у файл з дескриптором *f* і повертає "Test"

1.14 Спеціальні функції

(eval <вираз>)

Функція розраховує «вираз» і повертає його значення. Вираз є будь-якою допустимою у *LISP* конструкцією.

Приклад: (eval (abs -10)) результат 10
(setq a '(+ 3 5)) (eval a) результат 8

(initget [<код>] [<рядок>])

Функція дозволяє задати режими, які надалі будуть використовуватись у функціях *GETxxx* (крім функцій *GETSTRING* і *GETVAR*). Результатом функції завжди є *nil*. Код є цілим числом, яке інтерпретується відповідно до табл. 5.

Таблиця 5

Код	Інтерпретація
1	Уведення порожнього рядка не дозволяється
2	Заборона введення нульових значень
4	Заборона введення негативних чисел
8	Вимкнення перевірки меж (навіть при ввімкненому *LIMCHECK HECK*)

Дозволяється складати табличні коди. У цьому разі новий код інтерпретується як сукупність відповідних режимів. Якщо введення користувача не задовольняє встановлені режими, то AutoCAD виводить відповідне оголошення і пропонує повторити введення.

Приклад: (initget (+ 1 2 4)
(setq age (getint "введіть рік народження: "))

Функція *GETINT* дозволяє ввести рік народження. Якщо вводиться порожній рядок, нуль чи негативне число, то запит на введення повторюється. Режими перевіряються тільки для тих функцій, для яких це доцільно.

Другий необов'язковий аргумент функції є списком ключових слів, які будуть перевірятися наступним запитом *GETxxx* у тому випадку, коли користувач уводить не те, на що очікує програма (наприклад, точку для *GETPOINT*). Якщо введення користувача збігається з одним із ключових слів, то функція *GETxxx* повертає це слово у форматі рядка. Як правило, функція *COND* визначає подальшу дію програм, якщо вводяться дані іншого типу або ключове слово не можна впізнати, тоді AutoCAD повторює запит на введення. Список ключових слів має формат *Line Circle*, слова відокремлено пробілами. При вказаному запису специфікувати слово можна за першою літерою. Область дії функції *INITGET* – тільки до першого виклику функції *GETxxx*. Потім всі прапорці *INITGET* автоматично скидаються.

(menucmd <рядок>)

Функція забезпечує візуалізацію вказаної сторінки меню. Сторінка викликається з поточного файлу меню. Всередині функції ім'я сторінки завдається її позначкою (без "**"). Крім того, знак $ не вказується.

Приклад: (menucmd "S=OSNAP ")

У результаті на екрані з'явиться меню команди *OSNAP* (за умови, що елемент із таким ім'ям є у поточному меню).

1.15 Системні змінні

Система AutoCAD має різні попередні змінні, багато з яких можуть використовуватись для зміни різноманітних режимів і меж креслення. Доступ до системних змінних здійснюється за допомогою функцій *GETVAR* і *SETVAR*.

(getvar <ім'я-змінної>)

Функція дозволяє одержати від AutoCAD значення системної змінної. Ім'я змінної необхідно взяти у подвійні лапки. Нехай, наприклад, радіус спряження дорівнює 0,25 (значення змінної *FILLETRAD* дорівнює 0,25).

Приклад: (getvar "FILLETRAD") результат 0.250000

(setvar <ім'я-змінної> значення)

Присвоює задане значення вказаній системній змінній. Ім'я змінної слід узяти у подвійні лапки (у дод. 1 наведено список системних змінних AutoCAD).

1.16 Доступ до примітивів і пристроїв

1.16.1. Спеціальні типи даних

(ssget [<режим>] [<точка1> [точка2>]])

Функцію використовується для організації набору та вибору. Аргументи “точка1” і “точка2” є списками, котрі задають точки вибору. Задання точки без аргументу “режим” рівносильне вибору примітиву вказуванням точки. Якщо всі аргументи пропускаються, то *SSGET* через AutoCAD видає користувачу загальний запит *Select objects*:, дозволяючи інтерактивну конструкцію наборів вибору. Додатковий аргумент “режим” є рядком, котрий задає тип набору примітиву, що виконується. Цим типом може бути "*_W*", "*_C*","*_L*","*_P*" або "*_X*", чи "*_I*".

(ssget "*_P*") Вибирає останній набір вибору

(ssget "*_L*") Вибирає останній, доданий до бази даних, примітив

(ssget '(2 2)) Вибирає примітив, що проходить через точку 2.2

(ssget "*_W*" '(0 0) '(5 5)) Вибирає примітиви у вікні від точки 0.0 до точки 5.5

(ssget "*_C*" '(0 0) '(1 1)) Вибирає примітиви, що перерізають прямокутник від 0.0 до 1.1

(ssget "*_X*" '((0 . "*TEXT* ")(8 . "*RAZM*"))) Вибирає текстові примітиви, примітиви, виконані командою *TEXT* в шари *RAZM*

(ssget "_I" '((0 . "LINE ") (62 . 5))) Аналогічно "*_X*"

(sslength <нв>)

Функція повертає ціле число, що містить номери примітивів у наборі вибору “нв”.

(ssname <нв> <індекс>)

Функція повертає ім'я примітиву елемента “індекс” набору вибору “нв”. Якщо аргумент “індекс” від'ємний або більше від номера останнього примітиву в наборі вибору, то повертається *nil*. Перший елемент у наборі має нульовий номер. Імена примітивів у наборах вибору, одержані за допомогою функції *SSGET*, завжди будуть іменами основних примітивів. Підпримітиви (атрибути блоків і верхівки поліліній) повертатися не будуть (доступ до них забезпечує *ENTNEXT*).

(ssadd [<ім'я> [<нв>]])

Якщо функцію можна викликати без аргументів, *SSADD* створює порожній набір вибору; якщо з аргументом імені одного примітиву – *SSADD* створює новий набір вибору, що містить це ім'я примітиву. Якщо функцію викликають з іменем примітиву та набором вибору, то вона додає іменований примітив до набору вибору. *SSADD* завжди повертає новий або модифікований набір. Зверніть увагу на те, що при додаванні примітиву до набору, новий примітив фізично додається до існуючого та набір, що передається як “нв”, повертається як результат. Отже, якщо набір присвоєно іншим змінним, вони також будуть відображати це додавання. У разі, якщо іменований примітив уже знаходиться у наборі, функція *SSADD* буде ігноруватися; але повідомлень про помилки не буде.

(ssdel <ім'я> <нв>)

SSDEL стирає ім'я примітиву з набору вибору “нв” і повертає ім'я набору вибору “нв”. Зверніть увагу на те, що примітив фізично стирається з набору

вибору та повертається новий набір без указаного елемента. Якщо в наборі вибору немає примітиву, то повертається *nil*.

(ssmemb <ім'я> <нв>)

Функція перевіряє, чи є ім'я примітиву “ім'я” членом набору вибору “нв”. Якщо так, то функція *SSMEMB* повертає ім'я примітиву “ім'я”. В іншому разі, вона повертає *nil*.

1.16.2. Функції імені примітиву

(entnext [<ім'я>])

Якщо ця функція викликається без аргументів, то вона повертає ім'я першого невидаленого примітиву у базі даних. Якщо функцію *ENTNEXT* можна викликати з аргументом імені примітиву “ім'я”, то вона повертає ім'я першого невидаленого примітиву, котрий є наступним за примітивом “ім'я” у файлі креслення. Якщо у файлі немає такого примітиву, то повертається *nil*.

(entlast)

Функція повертає ім'я останнього невидаленого головного примітиву у файлі креслення. Її часто використовують для отримання імені нового примітиву, котрий додано за допомогою функції *COMMAND*. Примітив не обов'язково має бути на екрані чи на “розмерзлому” шарі.

(entsel [<підказка>])

Іноді, працюючи з примітивами, бажано одночасно вибирати примітив і задавати точку, за якою він вибирався. Прикладом цього може бути використання режимів об'єктної фіксації та команд *BREAK, TRIM* і *EXTEND* системи AutoCAD. Функція *ENTSEL* дає можливість програмам AutoLISP виконувати цю операцію.

1.16.3. Функції обробки атрибутів примітивів

Функції, наведені у цьому розділі, дозволяють шукати і модифікувати дані, що є примітивами AutoCAD. У всіх функціях використовуються імена примітивів для ідентифікації конкретних даних.

(entdel <ім'я>)

Примітив з іменем “ім'я” вимикають з креслення, якщо він там є. Якщо примітив вимкнули з креслення раніше у даному сеансі, то функція повертає його. Вимкнення примітивів з файлу креслення відбувається під час виходу з графічного редактора, тому їх повернення за допомогою *ENTDEL* можливо тільки у рамках того сеансу, у якому вони були вимкнені.

(entget <ім'я>)

Функція шукає примітив “ім'я” у файлі креслення та повертає список, котрий є описом цього примітиву. Зверніть увагу на те, що в підсписок, який містить координати точки, не входить символ «точка». Оскільки точка є двохелементним списком, то вся група має бути трьохелементним списком.

Функція *CDR* має завжди повертати трьохелементний список, а символ “точка” доповнює список до необхідної довжини.

(entmake [elist])

Функція може визначати як графічні, так і неграфічні об'єкти. “elist” – список об'єктів визначення даних, який задається у форматі, подібному до формату функції *ENTGET*. Аргумент *ELIST* має містити всю інформацію, необхідну для визначення об'єкта. Якщо будь-які необхідні визначення даних пропущені, функція *ENTMAKE* повертає *nil* і не виконується. Якщо пропущено необов'язкові дані (наприклад, шар – "*layer*"), *ENTMAKE* використовує значення за замовчуванням.

Тип об'єкта (наприклад, *CIRCLE* чи *LINE*) має займати перше чи друге поле у функції *ELIST*. Якщо тип об'єкта займає друге поле, то воно може бути тільки назвою об'єкта. Функція *ENTMAKE* ігнорує назву об'єкта під час створення нового об'єкта. Якщо *ELIST* містить об'єкт, створений вручну, то *ENTMAKE* його також ігнорує.

У разі успішного завершення функція повертає список об'єктів визначення даних. Інакше – повертає *nil*.

Команда при визначенні блоку (*ENTMAKE* з *ENDBLK*) повертає його ім'я, а не список об’єктів [19].

Приклад: наступний код створює червоне коло (колір 62), з центром у (4,4) і радіусом.

Необов'язковий шар і тип лінії поля були опущені і тому набувають значень за замовчуванням.

```
Command: (entmake  '((0 . "CIRCLE ") (62 . 1) (10 4.0 4.0 0.0)
(40 . 1.0))
((0 . "CIRCLE") (62 . 1) (10 4.0 4.0 0.0) (40.1 ))
```

Групу з кодом 66 використовують тільки для вставлення об'єктів (потім іде значення атрибутів). Для об'єктів полілінії групи з кодом 66 примусово присвоюють величину, що дорівнює 1, значення (*vertices* – вершини), і для всіх інших об'єктів присвоюють за замовчуванням 0. Єдиний об'єкт, що може випливати за об'єктом полілінії, – об'єкт вершини.

Група з кодом 2 (ім'я блоку) об'єкта вимірювання є необов'язковою для функції *ENTMAKE*. Якщо імені блоку немає у списку визначення об'єктів, AutoCAD створює новий список. Інакше кажучи, AutoCAD створює список, використовуючи вже підготовлене ім'я.

(entmod <список>)

Функції *ENTMOD* передається список у формі, котру генерує функція *ENTGET*. *ENTMOD* робить модифікацію цього списку та відповідне відновлення файлу креслення. Механізм оновлення файлу креслення за допомогою AutoLISP може бути таким. Спочатку за допомогою *ENTGET* проводиться пошук і вилучення списку примітиву, потім цей список модифікується за допомогою

AutoLISP (слід звернути увагу на доцільність використання функції *SUBST* у цьому процесі) та, нарешті, за допомогою *ENTMOD* цей список знов запам'ятовується у файлі креслення.

Існує ряд обмежень на можливості модифікації примітивів. По-перше, не змінюйте тип примітиву. Якщо треба це зробити, то примітив слід вилучити за допомогою *ENTDEL* та створити потрібний примітив за допомогою функції *COMMAND*. Усі елементи, на котрі посилаються фрагменти оброблюваного примітиву, мають бути визначені у програмі AutoCAD на момент виклику функції. Так, типи текстів та ліній, форми і блоки слід визначити під час використання їх у функції *ENTMOD*. Винятком є шари. Функція створює новий шар, якщо зустрічається посилання на шар, невідомий AutoCAD. Атрибути цього шару беруть за замовчуванням.

(entupd <ім'я>)

Під час модифікації вершин поліліній або атрибута за допомогою *ENMOD* на екрані не відбувається відображення виконуваних дій. Функція *ENTUPD* забезпечує повну регенерацію вказаного об'єкта.

Приклад використання функцій оброби атрибутів примітивів:

```
(defun  c:k ( )
; функція коректування полілінії
 (setq  $os (getvar "OSMODE")) (setvar  "OSMODE" 3)
(princ  (setq op (getpoint  "\n Old point ? : ")))
(setvar "OSMODE" 0)
(setq np (getpoint op "\n New point ? : ")) (print  np)
(setq gn (append  '(10) np) go (append '(10) op))
(setq ee (entnext ))
(while  ee
   (setq ee (entnext ee) ed (entget  ee))
   (if  (equal  (assoc 10 ed) go)
        (setq eg (subst  gn go ed) en ee ee nil)
    )
); while
(entmod  eg (entupd  en))
(setvar "OSMODE" $os)
)
```

1.16.4. Використання імен примітивів і наборів виборів

Ім’я примітиву або набір виборів є допустимою вхідною інформацією, що передається з AutoLISP у AutoCAD, у відповідь на запрошення вказати примітиви. Як результат, примітиви, вибрані у AutoLISP, можуть оброблятися AutoCAD. На запрошення *Select objects* AutoLISP може відповісти передачею імені конкретного примітиву або передачею набору виборів. Перший випадок рівносильний указанню цього примітиву з екрана, а у другому випадку в AutoCAD передаються усі примітиви набору. Передача в AutoCAD імені примітиву або набору виборів допустима тільки у тих випадках, коли можна

було б використати опцію *LAST* процесу казання на елементи. У всіх випадках примітиви не обов'язково мають бути видимими.

1.16.5. Доступ до елементів таблиць

Таблицями символів AutoCAD є таблиці шарів, типів ліній, видів, описів блоків і текстових стилів. Функції *TBLNEXT* та *TBLSEARCH* забезпечують читання інформації з цих таблиць.

(tblnext <ім'я таблиці> [<початок>]

Функцію використовують для повного перегляду таблиці символів. Допустимими іменами є *LAYER* – таблиця шарів, *LTYPE* – таблиця типів ліній, *VIEW* – таблиця видів, *STYLE* – таблиця текстових стилів і *BLOC* – таблиця описів блоків. Якщо задається другий аргумент і його значення не *nil*, то відбувається повернення до початку відповідної таблиці та вибір її першого елемента. У противному разі вибирається наступний елемент. Якщо всі елементи вичерпано, то функція повертає *nil*. Повернення вилучених елементів не відбувається. Якщо відповідний примітив виявлено, то функція повертає його у формі “точкових пар”, близькій до формату функції *ENTGET*.

Приклад: (tblnext "layer" t) повертає перший шар
((0 . "LAYER ") тип символу
(2 . "0") ім'я символу
(70 . 0) прапорці
(62 . 7) номер кольору (якщо від'ємний, то шар вимкнено)
(6 . "CONTINIOUS") ім'я типу лінії)

(tblsearch <ім'я таблиці> <символ>)

Функція шукає у таблиці символів символ, що має задане ім'я. Імена таблиці та символу перетворюються до верхнього регістра автоматично. Якщо пошук закінчується успішно, то символ видається у такому ж форматі, як у функції *TBLNEXT*. В іншому разі – видається *nil*. Наприклад: (tblsearch "style" "standart") шукає у таблиці текстових стилів. Функція *TBLSEARCH* не впливає на порядок виведення елементів за допомогою *TBLNEXT*.

1.16.6. Доступ до екрана та інтерактивних пристроїв

Існують деякі функції AutoLISP, що забезпечують прямий доступ до екрана дисплея та пристрою введення даних, що дає можливість користувачу спілкуватися з AutoLISP, використовуючи функції AutoCAD.

(grclear)

Функція проводить очищення графічного екрана AutoCAD. Рядки підказки та статусу не змінюються. Попередній стан екрана можна відновити за допомогою функції *REDRAW* .

(grdraw <з> <в> <колір> [<яскравість>])

Функція креслить вектор, що з'єднує дві точки; “з” та “в” є відповідно початковою та кінцевою точками цього вектора. Координати точок задають у математичному просторі та відсікають відповідно до загальних правил. Колір вектора обумовлює третій аргумент функції.

(grtext [<рамка><текст>[<яскравість>]])

Функція дозволяє заносити текстову інформацію до відповідної позиції графічного екрана. Якщо “рамка” має номер від нуля до максимального номера елемента меню без одиниці, то текст потрапляє у поле відповідного елемента меню. При цьому відбувається усікання текстового рядка, якщо він занадто довгий, і доповнення справа пробілами, якщо навпаки – занадто короткий.

(grread [<стеження>])

Функція дозволяє напряму опитувати пристрої введення. Якщо необов'язковий аргумент “стеження” задано і він не є *nil*, то відбувається автоматичне стеження за пристроєм уведення, наприклад, *GRREAD* повертає список, першим елементом якого є код типу пристрою. Другий аргумент – це ціле число або точковий список, залежно від типу пристрою введення.

Приклад: (defun c:dellayer(/ e L)
(setq L (strcase (getstring "\nLayer to delite?")))
(setq e (entnext)) ;розпочати перегляд з першого примітиву
(while e ;перевірити шар (групу 8)
(if (=L (cdr(assoc 8(entget e))))
(entdel e) ;якщо потрібний шар, то вилучити примітив
)
(setq e (entnext e)) ;Наступний примітив
) ;цикл по всіх примітивах
)

У цьому прикладі розглянуто команду, за допомогою якої видаляються усі примітиви з конкретного шару.

2. ФУНКЦІЇ *VISUAL LISP*

Visual LISP - технологія й система програмування, створена за допомогою об’єктно–орієнтованого середовища розробки додатків ObjectARX, що дає прості в застосуванні візуальні інструменти розробки прикладних програм для AutoCAD. Програмне забезпечення Visual LISP розширює можливості мови програмування AutoLISP, забезпечуючи просту розробку й застосування прикладних систем на базі AutoCAD, збільшуючи їхню продуктивність, інтеграцію з Microsoft Windows, програмну модульність і безпеку. Додатки, написані за допомогою Visual LISP, виконуються **в 2-5 рази швидше**, ніж програми, створені в AutoLISP, за рахунок використання переваг Active і можливості створення LISP-програм, керованих (Object Reactors)**.**

Visual LISP - це інструмент із серії засобів розроблювачів для AutoCAD**,** що включає також Microsoft Active, ObjectARX й Microsoft Visual Basic. Всі ці засоби є об’єктно–орієнтованими й дозволяють розробляти прикладні програми, які сумісні не тільки з AutoCAD, але й з іншим програмним забезпеченням, що підтримують Active. Розробники, які використають Visual LISP, можуть також скористатися перевагами убудованих можливостей Internet, удосконалених графікою, та загальною продуктивністю, що властива AutoCAD**.**

Основні відмінності середовища програмування Visual LISP від стандартного AutoLISP, що поставляє в складі AutoCAD, полягають у наступному:

- Простота використання**,** повністю візуальне середовище для розробки й налагодження програмного коду.
- Компільований LISP**,** що поліпшує продуктивність і запобігає небажаний доступ до вихідного програмного коду.
- Поліпшена продуктивність, з якої LISP-код виконується в інтерфейсі AutoCAD**,** заснований на технології ObjectARX.
- Удосконалений інтерфейс Microsoft Windows Active стосовно об'єктної моделі AutoCAD дає більшу гнучкість у крос-програмній інтеграції.

Середовище програмування Visual LISP рекомендуються для використання досвідченими користувачами AutoCAD, які займаються автоматизацією власної діяльності, а також для розроблювачів прикладних систем, що працюють у слабко формалізованих областях, що вимагають частої зміни програмного коду.

Завантаження розширених функції AutoLISP, які використовуються в Visual LISP здійснюється по команді **(vl-load-com).**

2.1 Функції для роботи з директоріями та файлами

(vl-mkdir <ім'я каталогу>**)**

Функція створює каталог з імям <ім'я каталогу>. Повертає T, якщо створення вдале. Для створення вкладених уаталогів потрібно створювати їх послідовно.

Приклад: (vl-mkdir "c:\\mydirectory")

результат "T"

(vl-directory-files [<**directory**>] [<**шаблон** >][<**число**>]**)**

Функція виконує пошук файлів в каталозі **directory** і повертає перелік файлів та каталогів. Якщо каталог відсутній або nil, використовується поточний каталог, якщо **шаблон** nil або відсутній, використається пошук по *.*; якщо **число** дорівнює -1 виконується пошук тільки каталогів; якщо 0 – файли й каталоги; якщо 1 – файли.

Приклад: (vl-directory-files "C:/AUTOCAD/VLISP" "*.ARX" 1)

результат ("Vlarts.arx" "Vlide.arx" "Vlrts.arx")

(vl-file-copy <"**файл1**"> <"**файл2**"> [<**число>**]**)**

Функція копіює **файл1** і додає до **файлу2**, якщо третій параметр є і не nil. Якщо функція повертає nil, то можлива помилка при копіюванні: **файл1** не читається, файл є каталогом, **файл2** існує.

Приклад: (vl-file-copy "C:/AUTOEXEC.BAT" "C:/NEWAUTO.BAT")

результат 1417

(vl-file-delete <"**ім'я_файлу**">**)**

Функція видаляє файл **ім'я_файлу**. Повертає T, якщо видалення вдале, і nil, якщо видалення не відбулося.

Приклад: (vl-file-delete " NEWAUTO.BAT ")

результат nil

(vl-file-directory-p <"**ім'я**">**)**

Функція визначає, чи є **ім'я** ім'ям каталогу. Повертає T, якщо так, і nil, якщо ні.

Приклад: (vl-file-directory-p "SAMPLE")

результат nil

(vl-file-rename <"**нове_ім'я**"> < "**старе_ім'я**" >**)**

Функція перейменовує файл зі **старе_ім'я** в **нове_ім'я**. Повертає T, якщо перейменовано, і nil, якщо операція виконана.

Приклад: (vl-file-rename "C:/PRW.LSP" "C:/PR1W.LSP")

результат T

(vl-file-size <"**ім'я_файлу**">**)**

Функція визначає розмір файлу **ім'я_файлу** в байтах. Повертає nil, якщо ім'я файлу не прочитано, і 0, якщо ім'я файлу – каталог або порожній файл.

Приклад: (vl-file-size "C:/AUTOEXEC.BAT")

результат 1895

(vl-file-systime <"**ім'я_файлу**">**)**

Функція визначає час останньої модифікації файлу. Повертає список, що містить дату і час останньої зміни, або nil, якщо файл не знайдено.

Приклад: (vl-file-systime "C:/ALISP/HANOIW.LSP")

результат (2009 1 0 31 20 37 38 0)

Останній раз файл був змінений в 2009 році, в 1-му місяці року (січень), 0-й день тижня (неділя), 31 січня о 20:37:38

(vl-filename-base <"**ім'я_файлу**">**)**

Функція повертає **ім'я_файлу** після вилучення його шляху до каталогу **directory** і розширення.

Приклад: (vl-filename-base "C:/ALISP/HANOIW.LSP")

результат "HANOIW"

(vl-filename-directory <"**ім'я_файлу**">**)**

Функція повертає шлях до каталогу файлу **ім'я_файлу**, без імені файлу.

Приклад: (vl-filename-directory "C:/ALISP/HANOIW.LSP")

результат "C:/ALISP"

(vl-filename-extension <"**ім'я_файлу**">**)**

Функція повертає тільки розширення від імені файлу **ім'я_файлу**, якщо його немає – nil.

Приклад: (vl-filename-extension "C:/PRIVETW.LSP")

результат ".LSP"

(vl-filename-mktemp [<**шаблон**>][<**directory**>][<**розширення**>]**)**

Функція визначає унікальне ім'я файлу, яке потрібно використати для тимчасового файлу. Основа імені – до 5 символів, прийнятих з імені файлу, та 3 унікальні комбінації із символів. Усі імена файлу, що були згенеровані функцією **(vl-filename-mktemp...)** протягом сеансу Visual LISP видаляються, коли виходять із Visual LISP.

Приклад: (vl-filename-mktemp "MYAPP.DEL")

результат "C:\\TMP\\MYAPP005.DEL"

2.2 Функції для визначення змінних в AutoLISP

(vl-acad-defun <символ>)

Функція визначає ім'я функції Visual LISP в AutoLISP.

Приклад: (vl-acad-defun c)

(vl-acad-undefun <символ>)

Функція скасовує ім'я функції Visual LISP в AutoLISP. Повертає T, якщо скасувала, і nil, якщо не скасувала (наприклад, функція не була визначена в AutoLISP).

Приклад: (vl-acad-undefun r)

результат nil

(vl-autocad-defun <'ім'я функції>)

Функція експортує **ім'я_функції** в AutoCAD.

За замовчуванням будь-які функції Visual LISP, чиї імена починаються із C:, експортуються в AutoCAD автоматично, якщо Visual LISP змінна системи *C-COLON-EXPORT* встановлена як T.

(vl-init)

Функція наслідує ініціалізації AutoLISP на відкритій або новій команді і повертає T, якщо ініціалізація пройшла успішно.

2.3 Функції для роботи з реєстром

(vl-registry-write <"розділ_реєстру"> [<"ім'я"> <"значення">])

Функція створює в **реєстрі** Windows новий **розділ_реєстру**, та параметри що складаються з **ім'я** і **значення**.

Приклад: (vl-registry-write "HKEY_CURRENT_USER\\Test" "test name" "test data")

результат "test data"

(vl-registry-read <розділ_реєстру> [<"ім'я"> <"значення">])

Функція читає дані, збережені у **розділ_реєстру** Windows.

Приклад: (vl-registry-read "HKEY_CURRENT_USER\\ Test" "test name")

результат "test name"

(vl-registry-delete <"розділ_реєстру"> [<"ім'я"> <"значення">])

Функція видаляє з **розділ_реєстру** Windows **ім'я** або **значення**.

Якщо регістр існує і не є nil, задана величина буде очищена від запису. Якщо **значення імені** відсутнє або є nil, функція видаляє визначену клавішу й всі її **значення**.

Приклад:(vl-registry-delete "HKEY_CURRENT_USER\\ Test")

результат T

(vl-registry-descendents <"розділ_реєстру"> [<список>])

Функція повертає **список** імен підрозділів обраного **розділ_реєстру**.

Приклад: (vl-registry-descendents "HKEY_LOCAL_MACHINE\\Software")

результат ("Description" "Program Groups" "Netscape" "Microsoft")

2.4 Функції для роботи зі списками

(vl-consp <символ>)

Функція визначає чи є змінна **символ** списком. Повертає T, якщо змінна є списком, і nil, якщо не є списком.

Приклад: (setq sp ‘(a b c)) (vl-consp sp)
результат T

(vl-list* <об'єкт>...)

Функція створює й повертає список. Коли останній параметр – атом, результат – точковий список. Якщо останній параметр – список, то функція приєднує до всіх попередніх параметрів.

Приклад: (vl-list* 1)
результат 1

(vl-list*->string <список ASCII-кодів>)

Функція перетворює **список ASCII-кодів** в елементи рядка.

Приклад: (vl-list->string '(49 50 51 52))
результат "1234"

(vl-list-length <список>)

Функція обчислює довжину **списку**.

Приклад: (vl-list-length ' (a b c d e))
результат 5

(vl-position <символ> <список>)

Функція повертає номер позиції визначеного елемента **списку**, де перший елемент **списку** під номером 0, другий − під номером 1 і так далі.

Приклад: (setq stuff (list "a" "b" "c" "d" "e"))
(vl-position "c" stuff)
результат 2

(vl-remove '<елемент> <список>)

Функція видаляє **елемент** зі **списку**.

Приклад: (**vl-remove** p1 (list p1 t 0 "abc"))
результат **(**t 0 "abc"**)**

(vl-prin1-to-string <"об'єкт">)

Функція повертає рядкове подання будь-якого **об'єкту** LISP, ніби вивід здійснювався функцією **(prin1...)**

Приклад: (vl-prin1-to-string "bcde")
результат "\"bcde\" "

(vl-princ-to-string <"об'єкт">)

Функція повертає рядкове подання будь-якого **об'єкту** LISP, ніби вивід здійснювався функцією **(princ)**.

Приклад: (vl-princ-to-string "abc")
результат "abc"

(vl-member-if '<умова> <список>)

Функція визначає, чи є **умова** вірною для одного з елементів **списку**. Умова – це функція з одним аргументом, що приймає значення T для будь-якої **умови**, призначеної користувачем.

Приклад: (vl-list-length ' (a b c d e))
результат 5

(vl-member-if <список>))

Функція передає кожен елемент у списку до функції, яка визначена у функції **умови**. Якщо функціональні повернення не є nil, то повертається частина **списку**, що залишилася, тим же самим способом, як і у функції **(member ...)**.

Приклад: (vl-member-if 'listp '(1 "str" (0. "string") nil T))
результат ((0. " string ") nil T)

(vl-member-if-not '<умова> <список>)

Функція визначає, чи є **умова** nil для одного з елементів **списку**.
Приклад: (vl-member-if-not 'atom '(1 "str" (0. "string") nil T))
результат ((0. " string ") nil T)

(vl-remove-if '<умова> <список>)

Функція видаляє всі елементи **списку**, які відповідають **умові**.
Приклад: (vl-remove-if 'listp (list ' (p1 t) 0 " abc "))
результат (0 " abc ")

(vl-remove-if-not '<умова> <список>)

Функція видаляє всі елементи **списку**, які не відповідають **умові**.
Приклад: (vl-remove-if-not 'listp (list ' (p1 t) 0 " abc "))
результат ((p1 t))

(vl-some '<умова> <список>...)

Функція перевіряє виконання **умови** для перших елементів кожного **списку**, потім для других і так далі. Якщо умова виконується – повертає T, якщо ні – nil.

Приклад: (vl-some '= Ll L2) ; порівнює елементи списків
результат nil

(vl-every '<умова> <список>...))

Функція перевіряє, чи виконується умова для кожної комбінації елементів списків (спочатку перших елементів, потім других і т.д.). Якщо умова виконується – повертає T, якщо ні – nil.

Приклад: (setq L1 '(1 2 3 4))
(setq L2 '(1 3 2 4))
(vl-every '<= L1 L2)
результат (1 2 3 4) (1 3 2 4) nil

(vl-sort <список> '<функція порівняння>)

Функція сортує елементи зі **списку** відповідно до **функції порівняння**. Функцією порівняння може бути будь-яка функція, що приймає два параметри і повертає T. Елементи, що повторюються, можуть бути вилучені зі списку.

Приклад: (vl-sort ' (3 2 1 3) '<) ; сортує за зростанням
результат (1 2 3)

(vl-sort-i <список> '<функція порівняння>)

Функція сортує елементи зі **списку** відповідно до **функції порівняння** і повертає номера (індекси) елементів. Елементи, що повторюються, будуть зберігатися.

Приклад: (vl-sort-i '("a" "d" "f" "c") '>) ; сортує за спаданням
результат (2 1 3 0)

2.5 Функції для роботи з рядками

(vl-string->list <рядок>)

Функція перетворює елементи **рядка** в список кодів ASCII.

Приклад: (vl-string->list "132")
результат (49 51 50)

(vl-string-elt <рядок > <позиція>)

Функція повертає зображення символу **рядка**, що знаходиться у визначеній **позиції** в **рядку**, в коді ASCII. Нульовій **позиції** відповідає перший символ у **рядку**.

Приклад: (vl-string-elt "Успіху Вам, користувачі!!!" 5)
результат 224 ; пояснення – на 5 позиції літера а, код ASCII ее – 224

(vl-string-left-trim <рядок1> <рядок2>)

Функція видаляє визначені символи з початку рядка символів. **Рядок1** – символи, що видаляються, **рядок2** – вихідні символи.

Приклад: (vl-string-left-trim "Пра" "Правнук")

результат "внук"

(vl-string-mismatch <рядок1> <рядок2> [<позиція1> < позиція2> <T>])

Функція повертає довжину співпадаючого фрагмента символів для двох **рядків**. Можна вказати **позиції** початку фрагмента. Наявність літери **T** знімає розходження в написанні рядкових і прописних літер.

Приклад: (vl-string-mismatch "Правнук" "Внук" 3 0 T)

результат 4

(vl-string-position <код символу> <рядок> [<поч.поз.1> <T>])

Функція визначає **позицію символу** з визначеним ASCII-**кодом** у **рядку**. **T**– пошук з кінця **рядка**.

Приклад: (vl-string-position 224 "Читач") ; ASCII код літери а – 224

результат 3

(vl-string-right-trim <"символи"> <рядок>)

Функція видаляє визначені **символи** з кінця **рядка**.

Приклад: (vl-string-right-trim "це" "сонце")

результат "сон"

(vl-string-search <"фрагмент"> <рядок> [позиція початку пошуку])

Функція визначає **початкову позицію** першого входження заданого **фрагмента** в **рядку**. При пушуку враховується значення регістра.

Приклад: (vl-string-search "Бар" "Санта-Барбара")

результат 6

(vl-string-subst <"нов_фраг"> <"стар_фраг"> <вих_рядок> [початок])

Функція виконує заміну **стар_фраг** у **вих_рядок** на **нов_фраг**. Пошук може починатися з **початок.** При пушуку враховується значення регістра.

Приклад: (vl-string-subst "98" "95" "Windows 95")

результат "Windows 98"

(vl-string-translate <"вихідний_набір"> <"новий_набір"> <рядок>)

Функція виконує заміну **вихідного_набору** символів у **рядку** на **новий_набір** символів. При цьому виконується заміна усіх символів.

Приклад: (vl-string-translate "246" "357" "str2 str4 str6")

результат "str3 str5 str7"

(vl-string-trim <"набір_символів"> <рядок>)
Функція видаляє **набір_символів** на початку і в кінці **рядка**.
Приклад: (vl-string-trim "—" "—***—")
результат **"***"**

2.6 Функції для роботи з символами

(vl-symbol-name <символ>)
Функція повертає рядок, який містить ім'я **символу**.
Приклад: (vl-symbol-name 's::startup)
результат "s::startup"

(vl-symbol-value <символ>)
Функція повертає поточне значення, пов'язане із **символом**. Ця функція еквівалентна функції (EVAL...) AutoLISP, але не викликає обчислювач LISP.
Приклад: (vl-symbol-value 't)
результат t

(vl-symbolp <об'єкт>)
Функція ідентифікує, чи дійсно визначений **об'єкт** є символом.
Приклад: (vl-symbolp t)
результат t

2.7 Функції для роботи з VLISP APLICATION

2.7.1. Функції для роботи з геометричним описанням об’єктів

(vlax-3d-point <список>)
Функція створює ACTIVEX-сумісну тривимірну структуру точки. **Список** – 2 або 3 числа, що являють точку.
Приклад: (vlax-3d-point 5 2 0)
результат (5.0 20.0 0.0)

(vlax-curve-getarea <крива>)
Функція визначає площу кривої (передбачається що крива замкнута), та повертає визначене значення.
Приклад: (vlax-curve-getarea ellipseobj)
результат 4.712393

(vlax-curve-getdistatparam <крива> <параметр>)
Функція повертає довжину сегмента **кривої** від початку **кривої** до відмітки, яка визначається **параметром**.
Приклад: (vlax-curve-getdistatparam splineobj (/(- endspline startspline) 2))
результат 8.99417

(vlax-curve-getdistatpoint <крива> <відмітка>)

Функція повертає довжину сегмента **кривої** між точкою початку **кривої** і визначеною **відміткою** (у координатах WCS).

Приклад: (vlax-curve-getdistatpoint splineobj selpt)
результат 5.17769

(vlax-curve-getendparam <крива>)

Функція повертає кінцеву точку **кривої**.

Приклад: (vlax-curve-getendparam ellipseobj)
результат 6.28319

(vlax-curve-getendpoint <крива>)

Функція повертає кінцеву точку **кривої** (у координатах WCS).

Приклад: (vlax-curve-getendpoint ellipseobj)
результат (2.0 2.0 0.0)

(vlax-curve-getpointatparam <крива> <параметр>)

Функція визначає точку на **кривій**, яка відповідає **параметру**, і повертає цю точку.

Приклад: (vlax-curve-getpointatparam splineobj (/(- endspline startspline) 2))
результат (6.71386 2.82748 0.0)

(vlax-curve-getstartparam <крива>)

Функція повертає параметр початку на **кривій**.

Приклад: (vlax-curve-getstartparam ellipseobj)
результат 0.0

(vlax-curve-getstartpoint <крива>)

Функція повертає відмітку початку **кривої** (у координатах WCS).

Приклад: (vlax-curve-getstartpoint splineobj)
результат (1.73962 2.12561 0.0)

(vlax-curve-isclosed <крива>)

Функція визначає, чи є визначена **крива** закритою.

Приклад: (vlax-curve-isclosed splineobj)
результат NIL

(vlax-curve-getclosestpointto <крива> <точка> [розширення])

Функція повертає точку (у координатах WCS) на **кривій**, яка є найближчою до визначеної **точки**. Якщо значення **розширення** не nil, то функція розширює криву при пошуку найближчої точки.

Приклад: (vlax-curve-getclosestpointto arcobj '(6.0 0.5 0.0))

результат (6.0 1.5 0.0)

(vlax-curve-getclosestpointtoprojection <крива> <точка> <нормаль> [розширення])

Функція повертає точку (у координатах WCS) на **кривій**, що є найближчою до визначеної **точки**. Якщо **розширення** не nil, то функція розширює криву при пошуку найближчої точки.

(vlax-curve-getfirstderiv <крива> <параметр>)

Функція повертає першу похідну (у координатах WCS) **кривої**.

Приклад: (vlax-curve-getfirstderiv splineobj (/ (- endspline startspline) 2))

результат (0.422631 -1.0951 0.0)

(vlax-curve-getsecondderiv <крива> <параметр>)

Функція повертає другу похідну (у координатах WCS) **кривої**.

Приклад: (vlax-curve-getsecondderiv splineobj (/ (- endspline startspline) 2))

результат (0.0165967 0.150848 0.0)

2.7.2. Функції для роботи з об’єктами

(vlax-dump-object <об'єкт VLA>)

Функція перераховує методи об'єкта і реквізити.

Приклад: (vlax-dump-object aa)

результат T

(vlax-ename -> VLA-object <вхідне ім'я об'єкта>)

Функція трансформує об'єкт в об'єкт VLA (**VLISP APLICATION**).

Приклад: (setq e (car (entsel)))

< entity name 27e0540 >

(vlax-ename-> vla-object e)

результат #<vla-object iautocadlwpolyline 03f713ao >

(vlax-erased-p <об'єкт VLA>)

Функція визначає, чи був об'єкт видалений. Повертає T, якщо об'єкт був видалений, в іншому випадку – nil.

(vlax-for <символ> <набір об'єктів> [вираз1 [вираз2...]])

Функція виконує ітерації через набір об'єктів, що оцінюють кожен вираз.

(vlax-get-autocad-object)

Функція відновлює верхній рівень об'єкта AutoCAD для поточного сеансу.

Приклад: (setq aa (vlax-get-autocad-object))

результат #<vla-object iautocadapplication 00b3b91c>

(vlax-invoke <об'єкт VLA> <метод> < список параметрів >)

Функція вказує визначений метод об'єкта. Функція **(vlax-invoke...)** може використовуватися для вказаного об'єкта ACTIVEX.

Приклад: (vlax-invoke vla-object "update" nil)

; це еквівалентно звертанню до стандарту vla:

(vla-update vla-object)

(vlax-ldata-delete <об'єкт VLA або AutoCad> <клавіша>)

Функція стирає дані LISP зі словника рисунка. Повертає T, якщо стерто успішно, і nil, якщо не стерто (наприклад, дані не існували).

Приклад: (vlax-ldata-delete "dict" "key")

результат T

(vlax-ldata-get <об'єкт VLA або AutoCAD> <клавіша> [задані за замовчуванням дані])

Функція відновлює дані LISP зі словника рисунка.

Приклад: (vlax-ldata-get "dict" "key")

результат (1)

(vlax-ldata-list <об'єкт VLA або AutoCAD>)

Функція перераховує дані LISP у словнику – асоціативний список, що складається з пар (клавіша-значення).

Приклад: (vlax-ldata-list "dict")

результат (("say" . "Floobar ") ("cay" . "Mumbo Jumbo "))

(vlax-ldata-put <об'єкт VLA або AutoCAD> <клавіша> <дані, що зберігаються>)

Функція зберігає дані LISP у словнику рисунка.

Приклад: (vlax-ldata-put "dict" "key" '(1))

результат (1)

(vlax-ldata-test <дані LISP >)

Функція визначає, чи можуть дані бути збережені за межами сеансу. Повертає T, якщо дані можуть бути збережені й відновлені за межами сеансу, і nil, якщо не можуть.

Приклад: (vlax-ldata-test "Gumbo Jumbo")

результат T

(vlax-map-collection <об'єкт VLA> <функція>)

Функція застосовує функцію до всіх об'єктів у наборі. Повертає T, якщо застосувала, і nil, якщо ні.

Приклад: (vlax-map-collection (vla-get-modelspace acaddocument)
'vlax-dump-object)
результат T

(vlax-method-applicable-p <об'єкт VLA> <метод>)

Функція визначає, чи підтримує об'єкт специфічний метод. Повертає T, якщо підтримує, і nil, якщо ні.

Приклад: (vlax-method-applicable-p WhatsMyLine 'copy)
результат T

(vlax-object-released-p <об'єкт VLA>)

Функція визначає, чи був **об'єкт** реалізований. Повертає T, якщо об’єкт реалізований, і nil, якщо ні.

Приклад: (vlax-object-released-p excelobj)
результат T

(vlax-product-key)

Функція повертає **шлях** запису AutoCAD. Цей шлях може бути використаний з метою реєстрації додатка для завантаження запиту.

Приклад: (vlax-product-key)
результат "Software\\Autodesk\\AutoCad\\Rl4.0\\ACAD-2451118:40800680"

(vlax-property-available-p <об'єкт VLA> <властивість> [T])

Функція визначає, чи має **об'єкт** визначену **властивість**. Повертає **T**, якщо об’єкт має визначену властивість, і nil, якщо ні.

Приклад: (vlax-property-available-p mycircle "area")
результат T

Зверніть увагу, як забезпечення факультативного третього параметра змінює результат. Функція повертає nil, хоча коло має властивість "area" («область»), що не може змінюватися.

Приклад: (vlax-property-available-p mycircle "area" T)
результат nil

(vlax-typeinfo-available-p <об'єкт VLA>)

Функція визначає доступність **typelib** інформації. Повертає T, якщо **typelib** інформація доступна, і NIL, якщо не доступна.

(vlax-put <об'єкт VLA> <властивість> <значення>)

Функція встановлює **значення властивості** функції нижнього рівня. Повертає nil, якщо встановлено значення властивості.

Приклад: (vlax-put vlaobj "color" 1)
результат nil

(vlax-read-enabled-p <об'єкт VLA>)

Функція визначає, чи може **об'єкт VLA** читатися. Повертає T, якщо об'єкт може бути прочитаний, і nil, якщо ні.

(vlax-write-enabled-p <об'єкт VLA або **вхідний об'єкт AutoLISP>)**

Функція визначає, чи може об'єкт, рисунок AutoCAD, змінюватися. Повертає T, якщо може змінюватися, і nil, якщо ні.

(vlax-reg-app <ім'я реєстрації додатка> <список опису команди> <int> [ім'я додатка [булевий прапорець]])

Функція реєструє додаток **(vlisp function).**

Ім'я_реєстрації_додатка – рядок у формі "програмне забезпечення\\<назва компанії>\\<назва програми>\\<версія програми>".

Список_описань_команди, кожна з яких повинна бути однією з: (<GLOBAL-CMD-NAME>.<LOCAL-CMD-NAME>) або <CMD-NAME>. Кожне <CMD-ім'я> – рядок або символ.

Булевий_прапорець запобігає збою на операціях запису, якщо функція не повертає nil, і якщо повертає nil. Значення за замовчуванням nil.

Повернення – рядок, що містить шлях до розділу реєстру, в якому було зареєстровано додаток.

(vlax-release-object <об'єкт VLA>)

Функція звільняє об'єкт. Повернення невизначене.

(vlax-tmatrix <список>)

Функція повертає відповідне відображення для матриць перетворення 4x4, які потрібно використовувати у VLA методах. **Список –** список із чотирьох списків, кожен з яких складається з чотирьох чисел, і представляє елементи матриці перетворення.

(vlax-vla-object->ename <об'єкт VLA>)

Функція трансформує **об'єкт VLA** в об'єкт AutoLISP. Повертає ім'я об'єкта AutoLISP.

Приклад: (vlax-vla-object->ename vlaobj)

результат <entity name: 27e0540>

2.7.3. Функції для роботи з групами команд

(vlax-add-cmd <глобальне ім'я> <ім'я_функції> [<локальне ім'я> <cmd-прапорці>])

Функція додає команди до групи. **Ім'я_функції** – функції AutoLISP із нульовими параметрами.

CMD-прапорці – ціле число (значення за замовчуванням до acrx_cmd_modal | acrx_cmd_redraw).

Приклад: (vlax-add-cmd "hello-autocad" 'hello-autocad)

результат "hello-autocad"

(vlax-remove-cmd <глобальне ім'я>)

Функція видаляє одиночну команду або цілу групу команд для поточного сеансу AutoCAD. Повертає T, якщо видалила, і nil, якщо ні.

Приклад: (vlax-remove-cmd "hello-autocad")

результат T

2.8 Функції зв’язку між Visual LISP і AutoLISP

(vlisp-export-symbol 'symbol-name)

Функція експортує змінну AutoLISP до значення, яке вона має в Visual LISP.

Приклад: (setq dd 1) (vlisp-export-symbol 'dd)

результат 1

(vlisp-import-exsubrs '("ім'я додатку" "назва функцій" ...))

Функція реєструє точку входу або додатка ARX до середовища Visual LISP.

Приклад: (vlisp-import-exsubrs '("AUTOCADAPP.EXE" "APPLOAD"))

результат ("AUTOCADAPP" "APPload" "C:PSDRAG" "MTPROP" "MTEDIT" "C:PSFILL" "BHATCH" "BPOLY" "C:PSIN" "AUTOCAD_COLORDLG" "STARTAPP" "ISMNUGRPLOADED" "INITDIA")

(vlisp-import-symbol <'символ AutoLISP або список символів>)

Функція призначає символ Visual LISP тому ж самому значенню, яке він має в AutoLISP. Повертає значення символу, що імпортується, або значення останнього символу, що імпортується, зі списку символів.

Приклад: (vlisp-import-symbol 'dd)

результат 1

2.9 Функції для роботи з об’єктними REACTOR

Реактори - це зв'язок між об'єктами *AutoCAD* і функціями, які автоматично викликаються при певних діях або ситуаціях роботи над кресленням. Реактор і функція реакції (*callback function*) визначають сценарій, як повинен поводитися об'єкт.

Є кілька видів реакторів:

- управління документами (*Document Reactors*)
- бази даних креслення (*Database Reactors*)
- командного редактора (*Editor Reactors*)
- зв'язок додатків (*Linker Reactors*)
- об'єктні (*Object Reactors*).

В Visual LISP використовуються наступні функції для роботи з реактором.

(vlr-acdb-reactor <дані AutoLISP> <список пар>)

Функція створює базу даних об'єкту REACTOR (глобальною змінною). Список пар включає **(<ім'я події>.<функцію повторного виклику>)**, де ім'я події – один з символів, перерахованих в таблиці **ACDB**. Кожна функція повторного виклику приймає два параметри: REACTOR_OBJECT – об’єкт VLR, що називається функцією повторного виклику, і OBJ – об'єкт бази даних (передається як об'єкт AutoLISP), пов'язаний з подією.

(vlr-add <об'єкт VLR>)

Функція забезпечує доступ до заблокованого об'єкту REACTOR.

(vlr-added-p <об'єкт VLR>)

Функція забезпечує перевірку допуску до об'єкту REACTOR. Повертає T, якщо допуск можливий, і nil, якщо REACTOR заблокований.

(vlr-beep-reaction [параметри])

Функція видає звуковий сигнал. Ця функція працює тільки в Visual LISP IDE.

(vlr-current-reaction-name)

Функція повертає ім'я поточної події, якщо викликається з середини повторного виклику об'єкту REACTOR.

(vlr-data-set <об'єкт VLR>)

Функція повертає специфічні для додатку дані, пов'язані з об'єктом REACTOR.

(vlr-data-set <об'єкт VLR> <дані AutoLISP>)

Функція записує поверх специфічні для застосування дані, пов'язані з об'єктом REACTOR.

Приклад: (vlr-data-set circleReactor "Circle Area Reactor")

результат "Circle Area Reactor"

(vlr-editor-reactor <дані AutoLISP> <список пар>)

Функція створює редактор об'єкту REACTOR. Об'єкт REACTOR буде доданий до бази даних рисунку, але не стане постійним.

Список_пар включає **(<ім'я події>. <функція повторного виклику>)**, де **ім'я_події** – один з символів, перерахованих в **«таблиці редактора»**. Кожна функція повторного виклику приймає два параметри: REACTOR_OBJECT – об'єкт VLR, що називається функцією повторного виклику, і OBJ – об'єкт бази даних (передається як об'єкт AutoLISP), пов'язаний з подією.

Список_пар(<ім'я події> <список додаткових елементів даних, пов'язаних зі специфічною подією>)

(vlr-linker-reactor <дані AutoLISP> <список пар>)

Функція створює компоновник об'єкта REACTOR. **Список_пар** – (**<ім'я події>.<функція повторного виклику>**), де ім'я події – один із символів, перерахованих в «Таблиці подій компоновника». Кожна функція повторного виклику приймає два параметри: об'єкт VLR і список, що містить рядок імені програми ARX.

Приклад: (vlr-linker-reactor nil '((:vlr-rxAppLoaded . my-vlr-trace-reaction)))

результат #<VLR-Linker-Reactor>

(vlr-object-reactor <список AutoLISP об'єктів VLA> <дані AutoLISP> <список пар>)

Функція створює об'єктний об'єкт REACTOR. **Список_пар (<ім'я події> <функція повторного повернення>)**, де ім'я події – один із символів, перерахованих в «Таблиці подій обєкта». Кожна функція повторного виклику приймає три параметри: власник об'єкта VLA, об'єкт VLR і список додаткових елементів даних.

(vlr-owner-add <об'єкт VLR> <объект що додається>)

Функція додає об'єкт до списку NOTIFIERS об'єктів VLR об'єктного REACTOR нове джерело подій.

Приклад: (vlr-owner-add circleReactor archie)

результат #<vla-object iacadarc 03ad0bcc>

(vlr-owner-remove <об'єкт VLR> <объект VLR, що видаляється>)

Функція видаляє об'єкт зі списку NOTIFIERS об'єктного REACTOR.

Приклад: (vlr-owner-remove circleReactor archie)

результат #<vla-object iacadarc 03ad0bcc>

(vlr-owners <об'єкт VLR>)

Функція повертає список **об'єктів**, які повідомляють визначеному об'єкту REACTOR.

Приклад: (vlr-owners circleReactor)

результат (#<vla-object iacadcircle 01db98f4>
#<vla-object iacadcircle 01db9724>
#<vla-object iacadcircle 01db93d4>

#<vla-object iacadcircle 01db9084>)

(vlr-pers <об'єкт VLR>)
Функція робить **об'єкт** REACTOR постійним.
Приклад: (vlr-pers circleReactor)
результат #<VLR-Object-Reactor>

(vlr-pers-p <об'єкт VLR>)
Функція визначає, чи дійсно **об'єкт** REACTOR постійний.
Приклад: (vlr-pers-p circleReactor)
результат #<VLR-Object-Reactor>

(vlr-pers-release <об'єкт VLR>)
Функція робить об'єкт REACTOR нерезидентним.

(vlr-reaction-names <reactor-type>)
Функція повертає список всіх умов повторного виклику для цього типу об'єкта REACTOR (VLISP FUNCTION)
Приклад: (vlr-reaction-names :VLR-Editor-Reactor)
результат (:vlr-unknownCommand :vlr-commandWillStart…

(vlr-reaction-set <об'єкт VLR> <тип події> <функція>)
Функція додає або заміняє функцію AutoLISP в об'єкті REACTOR.
Щоб використати функції, створені в Visual LISP в AutoCAD, необхідно експортувати імена функції в AutoCAD або ввести їх неявно.
При явному експортуванні використають функцію (VL-AUTOCAD-DEFUN...).
Приклад: (vlr-reaction-set circleReactor :vlr-modified 'print-area)
результат print-area

ІНДИВІДУАЛЬНІ ЗАВДАННЯ НА AUTOLISP

Рекомендації відносно складання програм

Відзначимо, що кожне креслення є унікальним і програма, яка його створює, може бути орієнтованою тільки на цей тип креслення, однак є декілька корисних прийомів, які необхідно враховувати під час складання програм, за допомогою яких виконують креслення.

1. Детальний аналіз креслення з метою його спрощення. Тут необхідно умовно виділити однотипні елементи, які будуються певною кількістю однакових примітивів (ліній, полілініій, дуг, кіл тощо), ділянки симетрії, паралельних переносів і т. ін.

2. Дослідження виділеного типового елемента креслення. Потрібно визначити точки, між якими слід провести примітиви, що їх з'єднують.

3. Визначення тих вибраних точок або розмірів, які потрібно задавати за допомогою команд уведення, або ж тих, як будуть визначатися за відомими залежностями у процесі виконання креслення.

Переважно першу точку, яку задають, називають базовою і умовно позначають bp. Визначати її в кресленні потрібно так, щоб від неї було зручно проводити обчислення для усієї конструкції (наприклад, на перетині ліній симетрії).

Програму потрібно скласти так, щоб запитань до користувача було якнайменше – це зменшує ймовірність помилок при введенні. З цією метою слід передбачити запити таких значень на кресленні, за допомогою яких можна однозначно визначити розміри всієї конструкції.

4. Визначення однотипних ділянок креслення (див. п. 1) з однаковим алгоритмом креслення, за винятком, можливо, деяких формальних параметрів (розміри, написи, позначення і т. д.). Після цього можна зупинитися на двох способах розв'язання завдання:

4.1. Створити функцію, яка креслить блок із заданням змінюваних параметрів, які вводять у функцію як формальні. При цьому з функції вищого рівня викликають функцію креслення потрібну кількість разів (наприклад, за допомогою циклу).

4.2. Організувати блок, який можна підставляти в конструкцію, вказуючи точку вставлення, масштаби по осях X і Y та кути повороту.

5. Якщо якась область має бути заштрихованою, то її межі зручно креслити однією полілінією, тоді для операції штрихування достатньо вказати тільки ім'я примітивів і координати однієї точки цієї полілінії.

6. Для проставлення декількох розмірів (горизонтальних або вертикальних) зручно використовувати відповідну функцію з визначеними формальними параметрами. Вигляд цієї функції може бути таким:

<ф-я для проставлення гор. розмірів>

<ф-я для проставлення верт. розмірів>

7. Визначення системних змінних, які мають відрізнятись від стандартних, і введення їх нових значень у головну функцію, при цьому використовують оператори *GETVAR* і *SETVAR*. Після закінчення роботи програми слід відновити початкові значення системних змінних.

8. Під час виконання креслення зручно користуватися шарами різного кольору для виконання різних операцій, це спрощує знаходження помилок, поліпшує наочність рисунка. Шари можуть бути такими:

OSN – чорного (білого) кольору – для креслення основних ліній;

OSI – мажентного кольору – для нанесення осьових ліній;

SHTR – червоного кольору – для штрихування;

RAZ – зеленого кольору – для нанесення розмірних ліній і тексту;

Через обмження, накладені на існуючий нульовий шар, його краще не використовувати в кресленнях.

Наприклад, розглянемо побудову прямокутника з нанесенням штрихування і розмірних ліній.

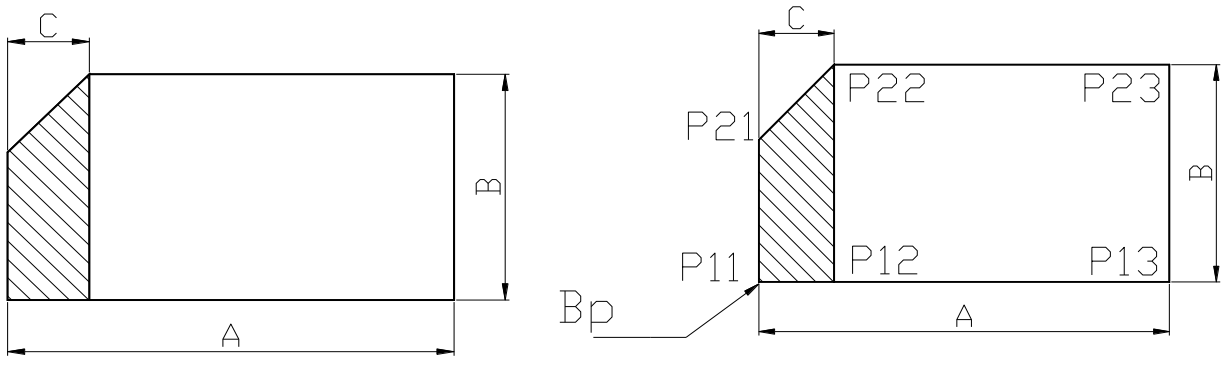
Базовою точкою *bp* вибрано нижній лівий кут прямокутника, *a* – розмір по осі *X*, *b* – по осі *Y*.

У тексті програми використовують координати точок *P1, P2, P3, P4*. *P1* – лівий нижній кут, *P2* – правий нижній кут, *P3* – правий верхній кут, *P4* – лівий верхній кут.

Зверніть увагу, що точки треба знаходити, визначивши їхні координати за допомогою функції *LIST* (визначається *P2*), або використовуючи функцію *POLAR* (визначають *P3, P4*). Повідомлення про помилки наведено в дод. 2. У разі використання графічної бази даних слід скористатися дод. 1,3.

Приклад виконання роботи

Розробити програму, з можливістю введенням формальних параметрів, для виконання креслення приведеного на Рис. 1 а.



а) б)

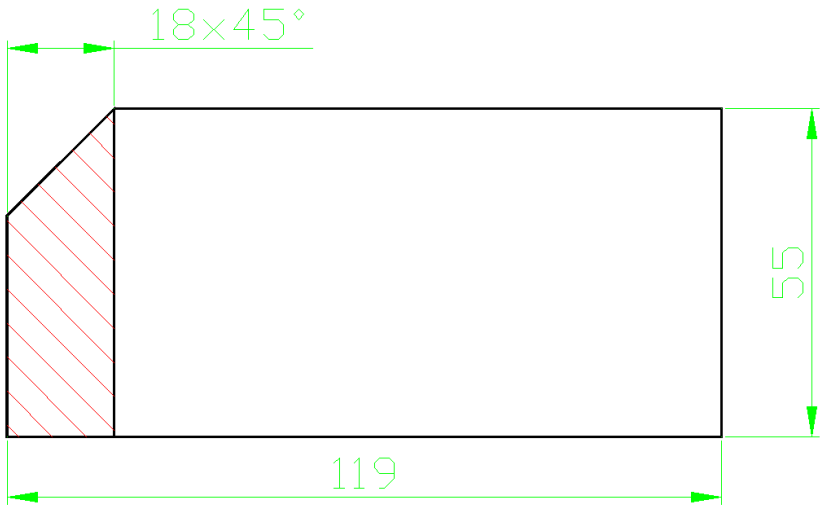
Рис. 1 – Приклад завдання
а) креслення деталі; б) визначення базової та опірних точок

Лістинг програми:

```
(defun c:prm ( )
; Установлення системних змінних для проставлення розмірів
 (setq txt (getvar "DIMTXT")) (setvar "DIMTXT" 5) ; висота тексту
 (setq asz (getvar "DIMASZ")) (setvar "DIMASZ" 5) ; розмір стрілки
 (setq tad (getvar "DIMTAD")) (setvar "DIMTAD" 1) ; текст над розм. лінією
 (setq tih (getvar "DIMTIH")) (setvar "DIMTIH") 0); неперервність розм. лінії
 (setq gap (getvar "DIMGAP"))(setvar "DIMGAP") 1.5); відстань від розм. лінії до тексту

; Уведення вихідних даних
  (setq bp (getpoint "\n Ввести Bp: "))
  (setq a  (getdist bp "\n Ввести a: "))
  (setq b  (getdist bp "\n Ввести b: "))
  (setq c  (getdist bp "\n Ввести c: "))

; Визначення точок
  (setq p11 bp
     p12 (polar p11 0 c)
     p13 (polar p11 0 a)
     p23 (polar p13 (/ pi 2) b)
     p21 (polar p11 (/ pi 2) (- b c))
     p22 (list (car p12)(cadr p23))
  )

  ; Побудова контура
; створення шару для креслення контуру
  (command "_LAYER" "_M" "OSN" "")
  (command "_PLINE" p11 p21 p22 p12 "_C")
 ; Визначення імені примітиву en для вказання його під час штрихування
 (setq en (entlast))
  (command "_PLINE" p22 p23 p13 p12 "")

; штрихування
  (command "_LAYER" "_M" "HATCH" "_C" 1 "" "")
  (command "_HATCH" "_u" 135 4  "_n" en "")

 ; установлення розмірів
 (command "_LAYER" "_M" "TEXT" "_C" 3 "" "")
  (setvar "dimasz" 5)(setvar "dimtxt" 5)(setvar "dimgap" 1.5)
  (command "_DIM")
   (command "_hor" p11 p13 (polar p11 (* pi 1.5) 10) (rtos a 2 0))
   (command "_ver" p13 p23 (polar p13 0 15) (rtos b 2 0))
   (command "_hor" p21 p22 (polar p22 (/ pi 2) 10) (strcat (rtos c 2 0) "x45%%d"))
```

```
 (command "_EXIT")

; Відновлення змінених системних змінних
 (setvar "DIMTXT" txt) (setvar "DIMASZ" asz)
 (setvar "DIMTAD" tad) (setvar "DIMTIH" tih)
 (setvar   "DIMGAP" gap )
 (princ)
)
```

Результат виконання



Індивідуальні завдання вибирають з відповідних рисунків "Завдання №..." згідно з номером бригади.

Завдання 1. Функція введення і обчислення арифметичних виразів. Побудова елементарних фігур

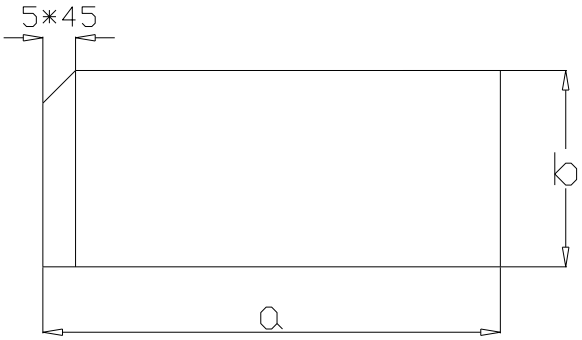
Мета: навчитись, використовуючи функції AutoLISP, уводити в програму значення змінних, застосовувати математичні функції та команди AutoCAD.

Порядок виконання

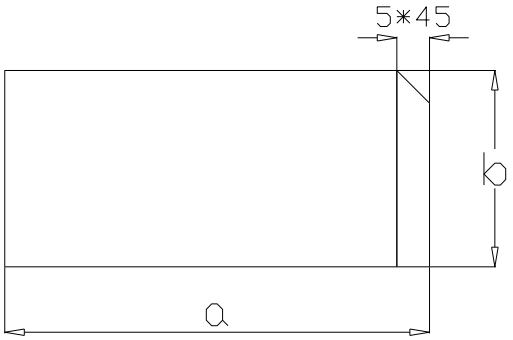
1. Індивідуальні завдання вибираються з **Рис. 2** згідно номеру бригади.
2. Аналізуючи індивідуальне завдання, визначити базову точку рисунка і змінні для введення.
3. Увести за допомогою редактора програму виконання рисунка, створивши файл *LRn.LSP*, де *n* – номер бригади.
4. Виконати програму.
5. Вивести результат на друк.

Контрольні запитання

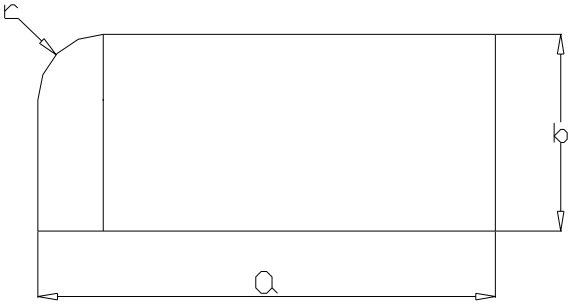
1. Функції введення-виведення, використані в програмі, їх структура і застосування.
2. Математичні функції, використані у програмі, їх застосування.
3. Виклик програм з AutoCAD.



Вр.1

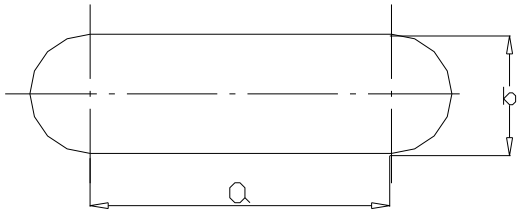


Вр.2

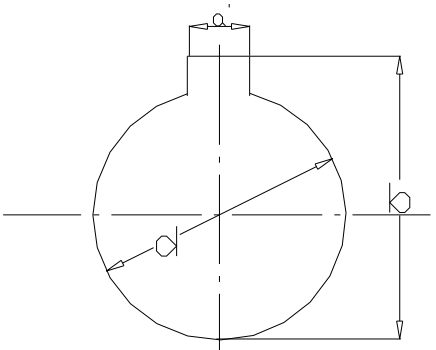


Вр.3

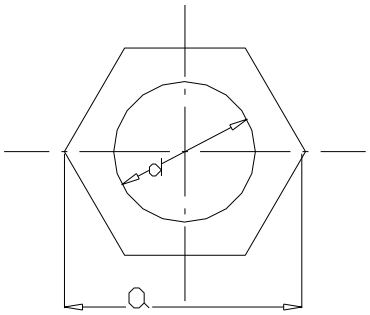
Вр.4



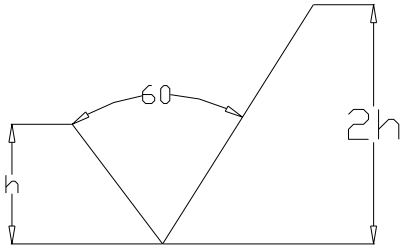
Вр.5



Вр.6



Вр.7



Вр.8

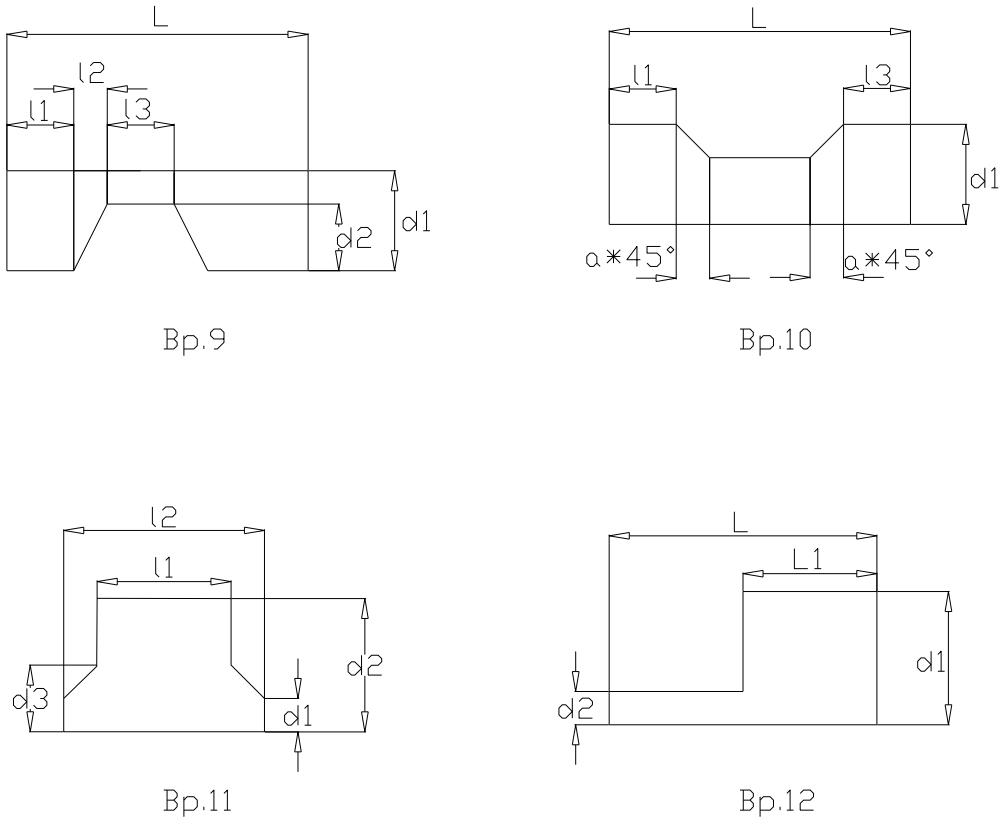


Рис. 2 – Завдання 1

Завдання 2. Функції введення-виведення, обчислення арифметичних виразів

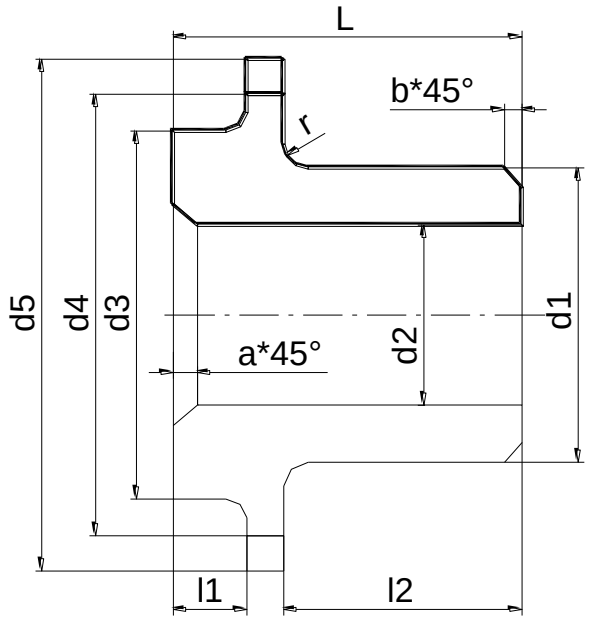
Мета: навчитися використовувати функції введення-виведення, математичних обчислень у програмі мовою AutoLISP.

Порядок виконання

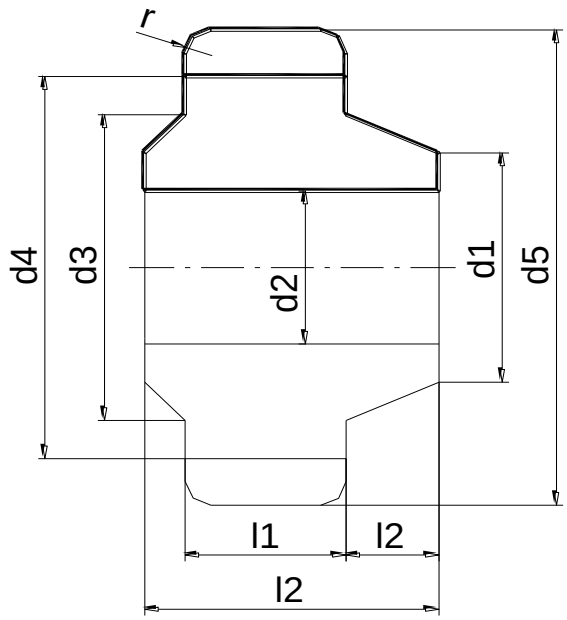
1. Індивідуальні завдання вибираються з **Рис. 3** згідно номеру бригади.
2. Аналізуючи індивідуальне завдання, визначити базову точку рисунка і змінні для введення.
3. Увести за допомогою редактора програму виконання рисунка, створивши файл *LRn.LSP*, де *n* – номер бригади.
4. Виконати програму.
5. Вивести на друк результат.

Контрольні запитання

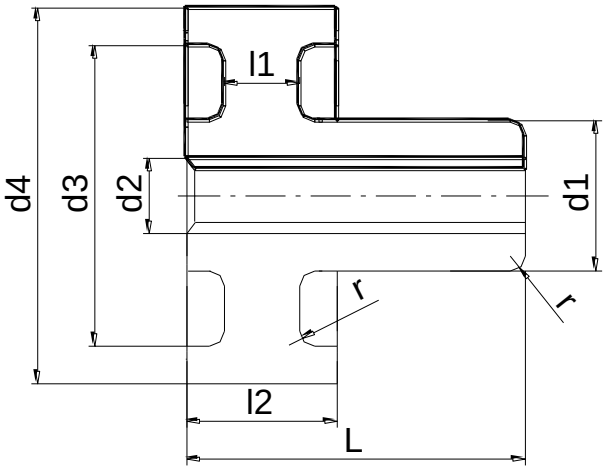
1. Функції введення-виведення в програмах, написаних мовою AutoLISP.
2. Математичні функції.
3. Функції для роботи з геометричним описом об'єкта.



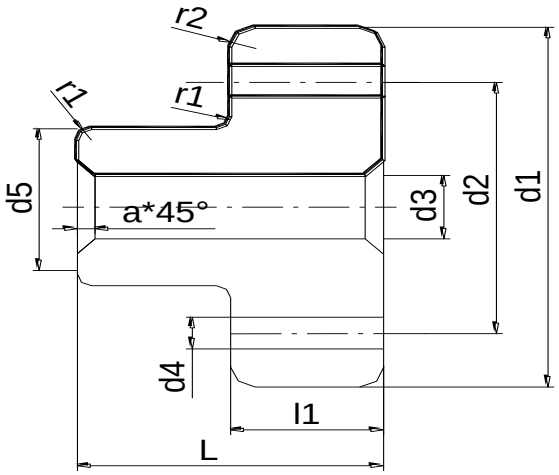
Бригада 1



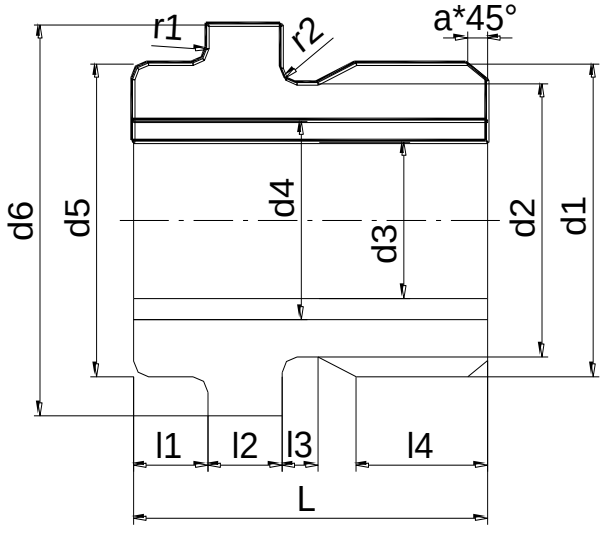
Бригада 2



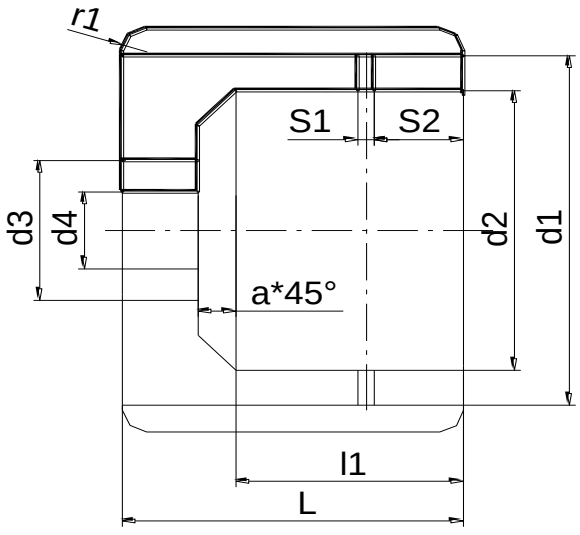
Бригада 3



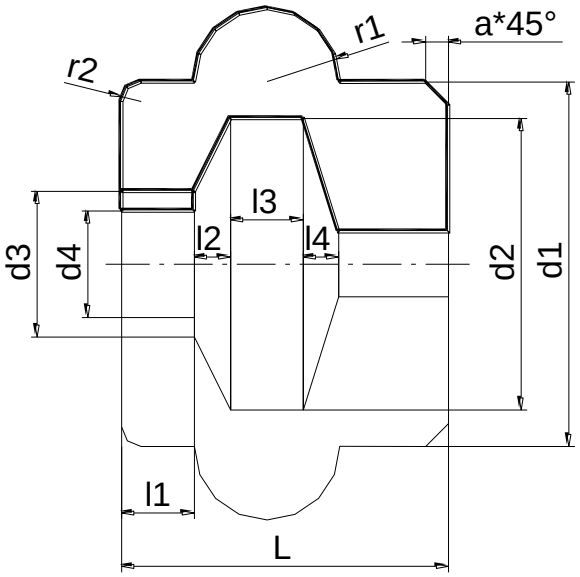
Бригада 4



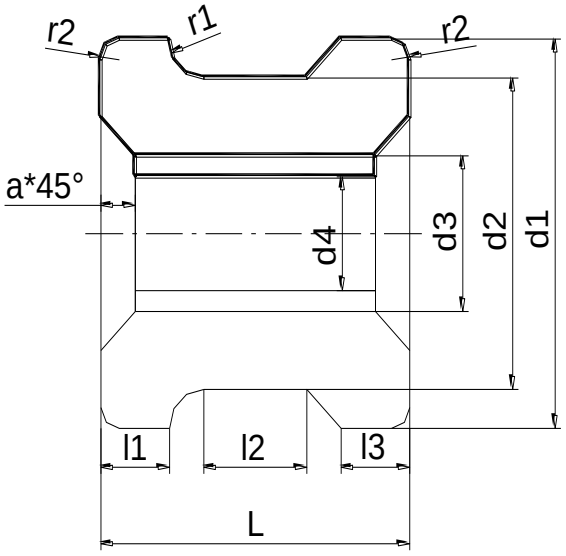
Бригада 5



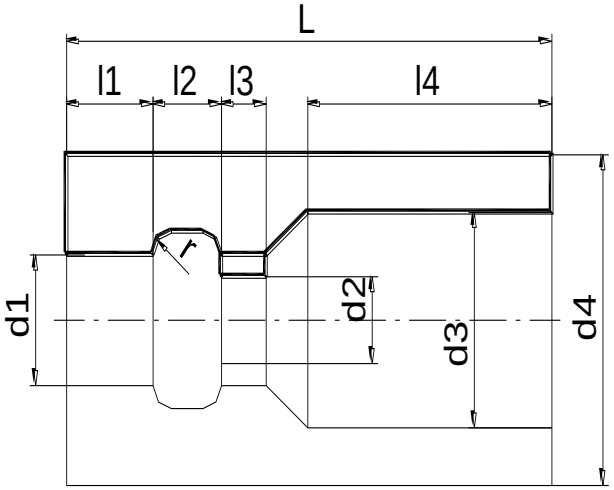
Бригада 6



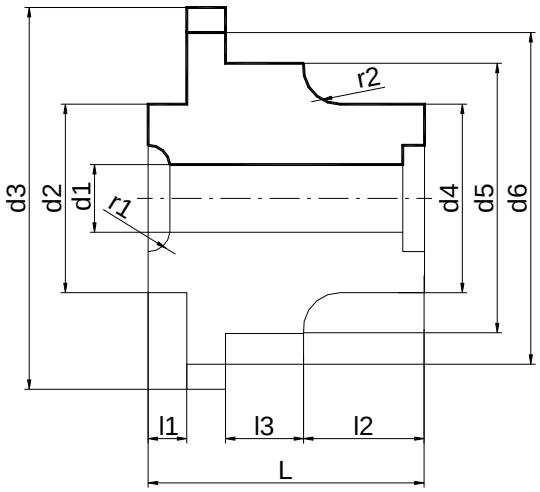
Бригада 7, де r1=(l2+l3+l4)/2



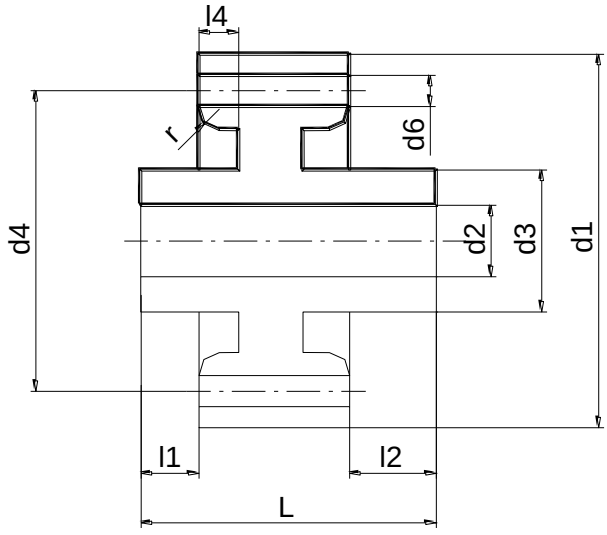
Бригада 8



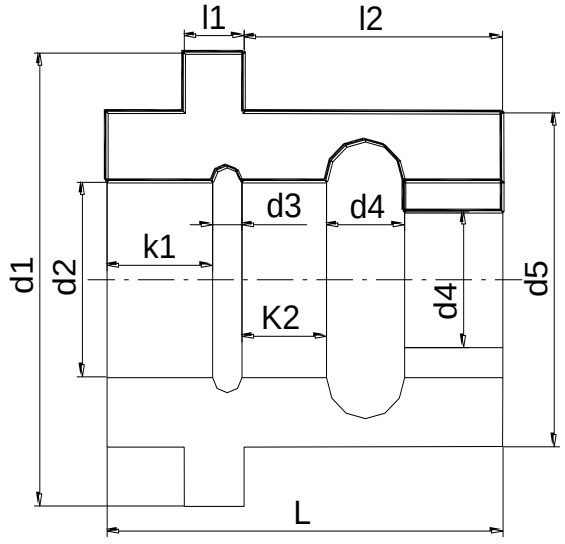
Бригада 9



Бригада 10



Бригада 11



Бригада 12

Рис. 3 – Завдання 2

Завдання 3. Програми, які передбачають симетричневідображення примітивів і нанесення штрихування

Мета: навчитися, використовуючи можливості AutoLISP, симетрично відображати або переносити ділянки креслення і наносити штрихування.

Порядок виконання

1. Індивідуальні завдання вибираються з **Рис. 4** згідно номеру бригади.
2. Аналізуючи індивідуальне завдання, визначити базову точку рисунка і змінні для введення.
3. Увести за допомогою редактора програму виконання рисунка, створивши файл *LRn.LSP*, де *n* – номер бригади.
4. Виконати програму.
5. Вивести результат на друк.

Контрольні запитання

1. Масиви (набори) даних в AutoLISP. Їх структура.
2. Функції для вибору змінних із набору даних.

Завдання 4. Програми, які передбачають проставлення розмірів

Мета: навчитися, використовуючи можливості AutoLISP, наносити на креслення розмірні лінії і проставляти розміри.

Порядок виконання:

1. Індивідуальні завдання вибираються з **Рис. 4** згідно номеру бригади.
2. Аналізуючи індивідуальне завдання, визначити базову точку рисунка і змінні для введення.
3. Увести за допомогою редактора програму виконання рисунка, створивши файл *LRn.LSP*, де *n* – номер бригади.
4. Виконати програму.
5. Вивести результат на друк.

Рекомендації щодо виконання роботи

Під час проставлення розмірів необхідно враховувати, що:
– відстань між паралельними розмірними лініями має бути 6…10 мм, а між першою лінією і контуром деталі – 10 мм;
– довжина стрілки розмірної лінії 4…6 мм;
– текст наносять над розмірною лінією паралельно;
– висота тексту – 4…5 мм.

Контрольні запитання

1. Функції введення і їх використання.
2. Функції для роботи з набором даних.
3. Функції для організування циклів.

Завдання 5. Програми для побудови об’ємної моделі

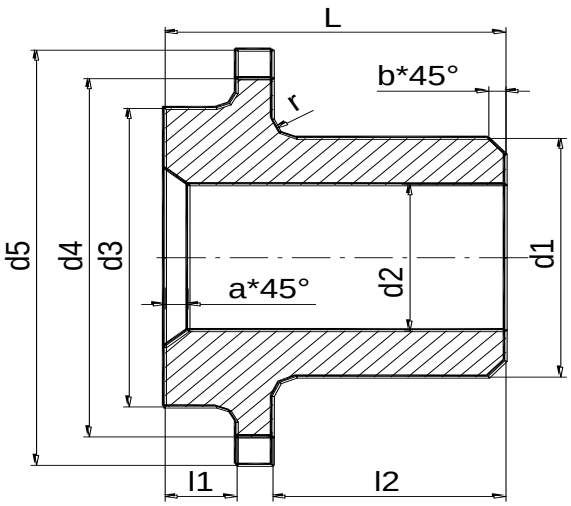
Мета: навчитися, використовуючи можливості AutoLISP та функції AutoCAD, виконувати об’ємної моделі.

Порядок виконання:

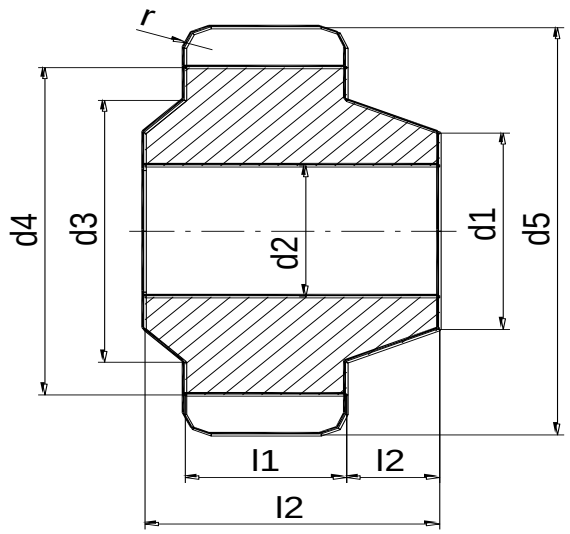
1. Індивідуальні завдання вибираються з **Рис. 4** згідно номеру бригади.
2. Аналізуючи індивідуальне завдання, визначити базову точку рисунка і змінні для введення.
3. Увести за допомогою редактора програму виконання рисунка, створивши файл *LRn.LSP*, де *n* – номер бригади.
4. Виконати програму для побудови об’ємної моделі.
5. Вивести результат на друк.

Контрольні запитання

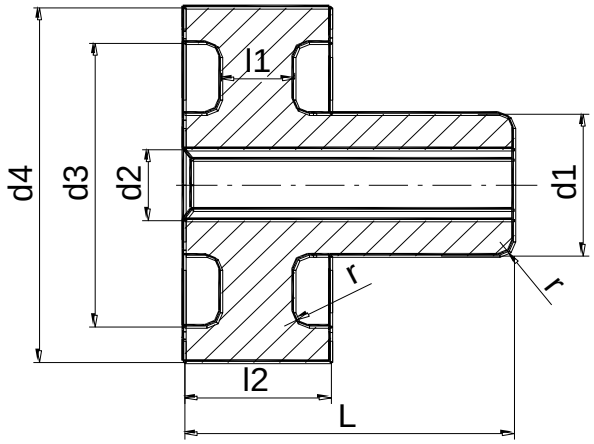
1. Команди для виконання 3D конструкції в AutoCad.
2. Виклик команд AutoCad з AutoLISP.
3. Визначення характеристик 3D конструкції .



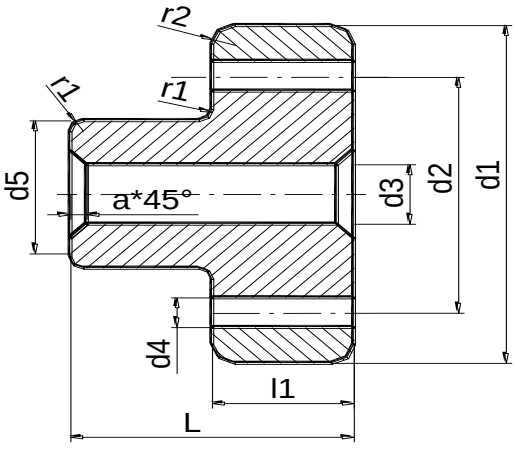
Бригада 1



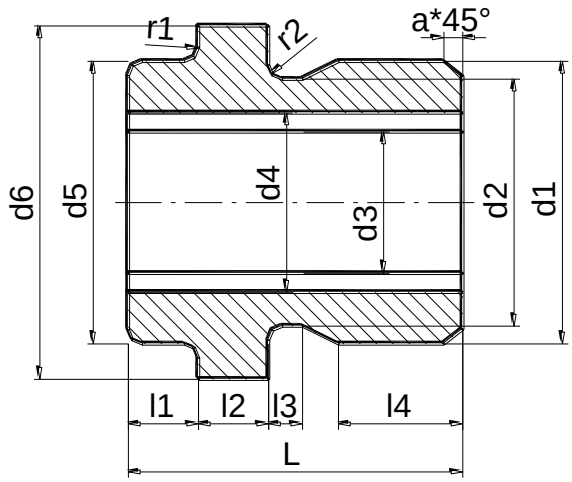
Бригада 2



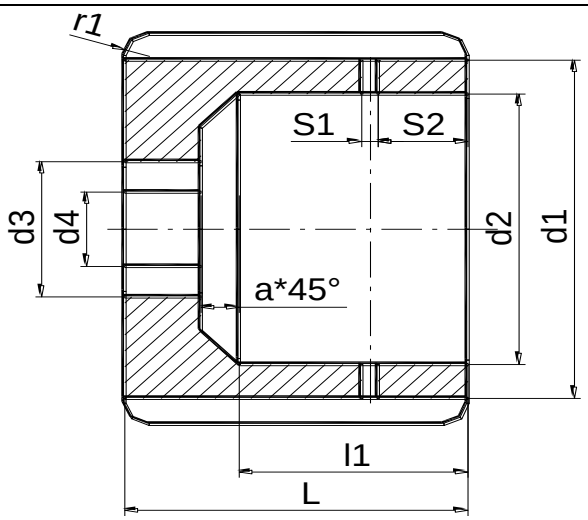
Бригада 3



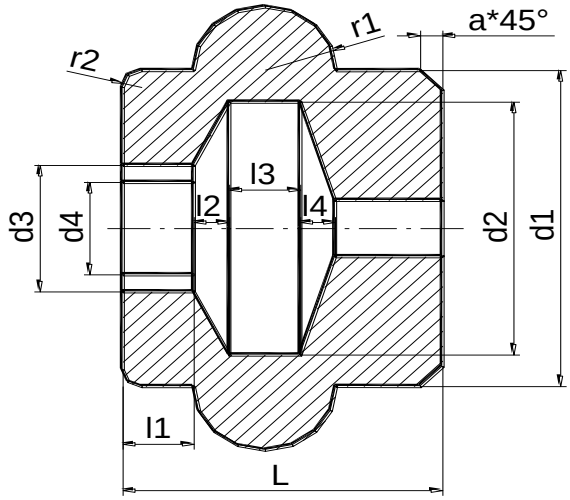
Бригада 4



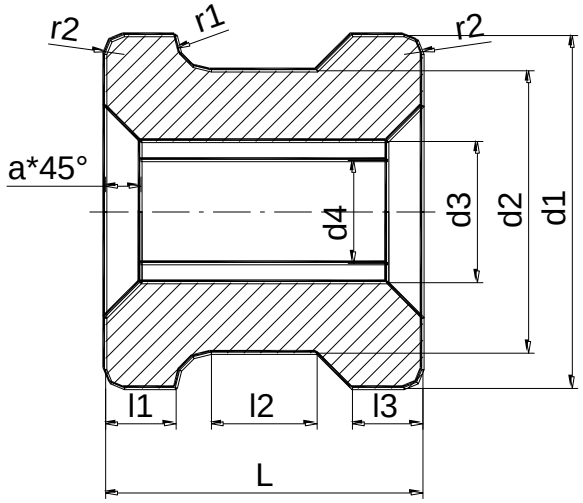
Бригада 5



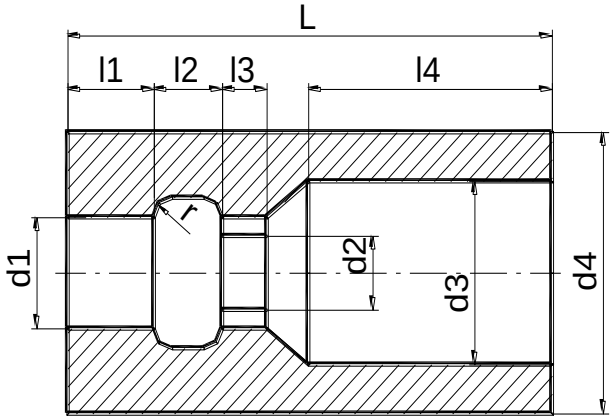
Бригада 6



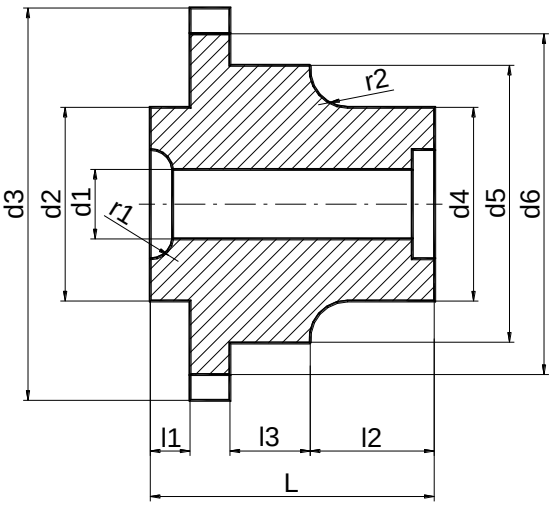
Бригада 7, де r1=(l2+l3+l4)/2



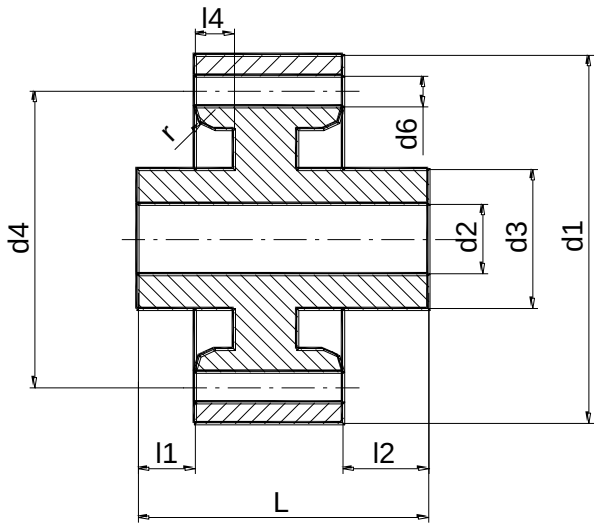
Бригада 8



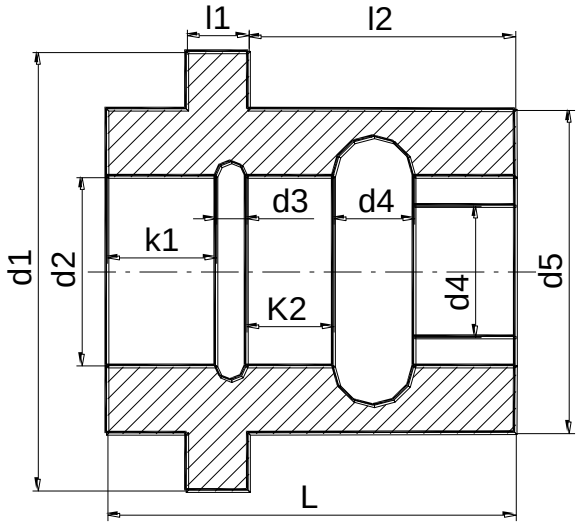
Бригада 9



Бригада 10



Бригада 11



Бригада 12

Рис. 4 – Завдання 3, 4, 5

Завдання 6. Розробка програм для побудови об’ємної моделі конструкції

Мета: навчитися складати програми з використанням розроблених функцій і команд AutoCAD.

Виконання

Скласти програму (функцію) для виконання твердотільної деталі (типу SOLD) згідно завданню групі та номеру бригади, приведеному на **Рис. 5** – **Рис. 25**.

Контрольні запитання

1. Функції перевірки типів даних.
2. Функції роботи зі списками.
3. Функції керування зображенням.

Завдання для групи ЛС

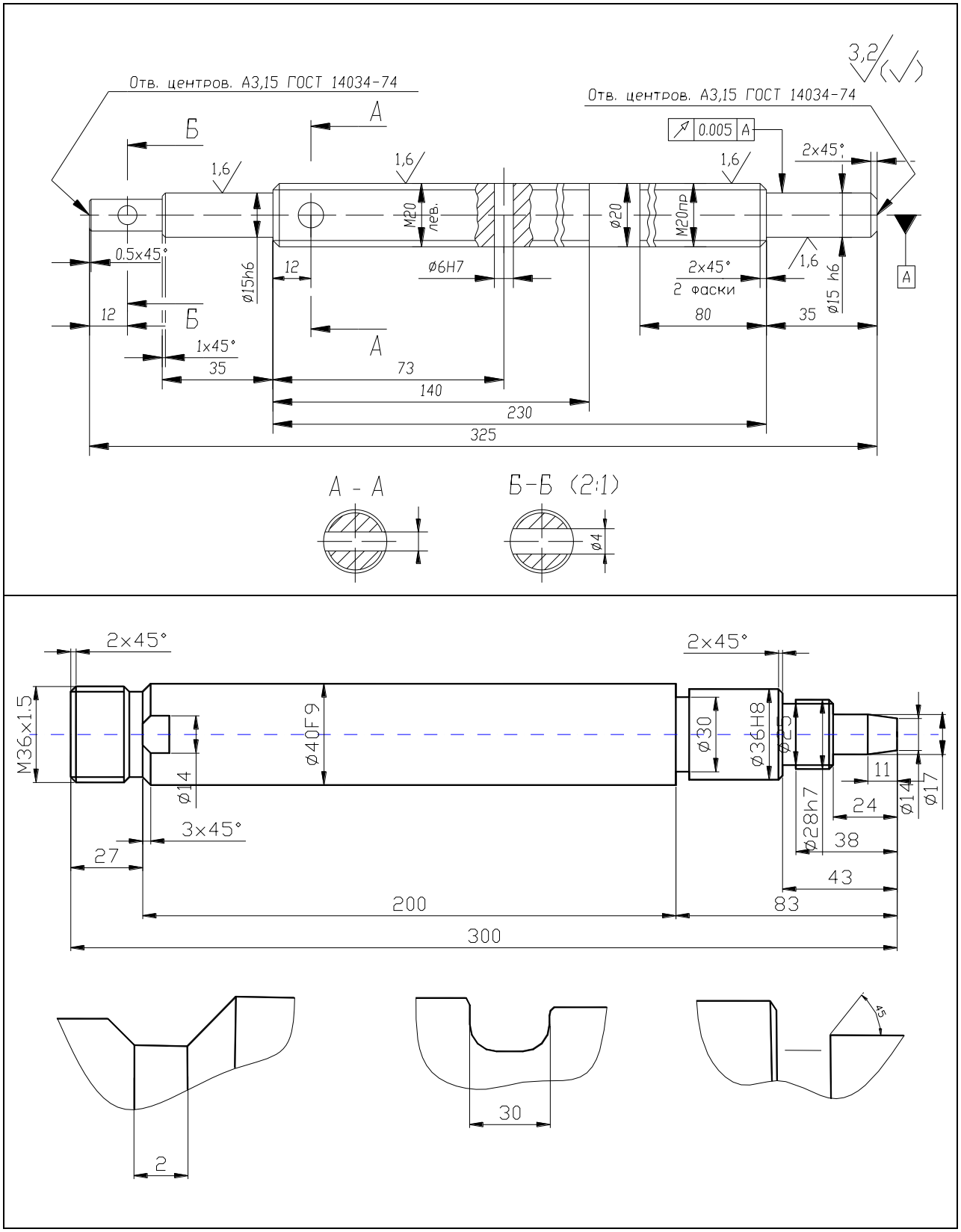


Рис. 5 – Завдання группа ЛС, бригада 1, бригада 2.

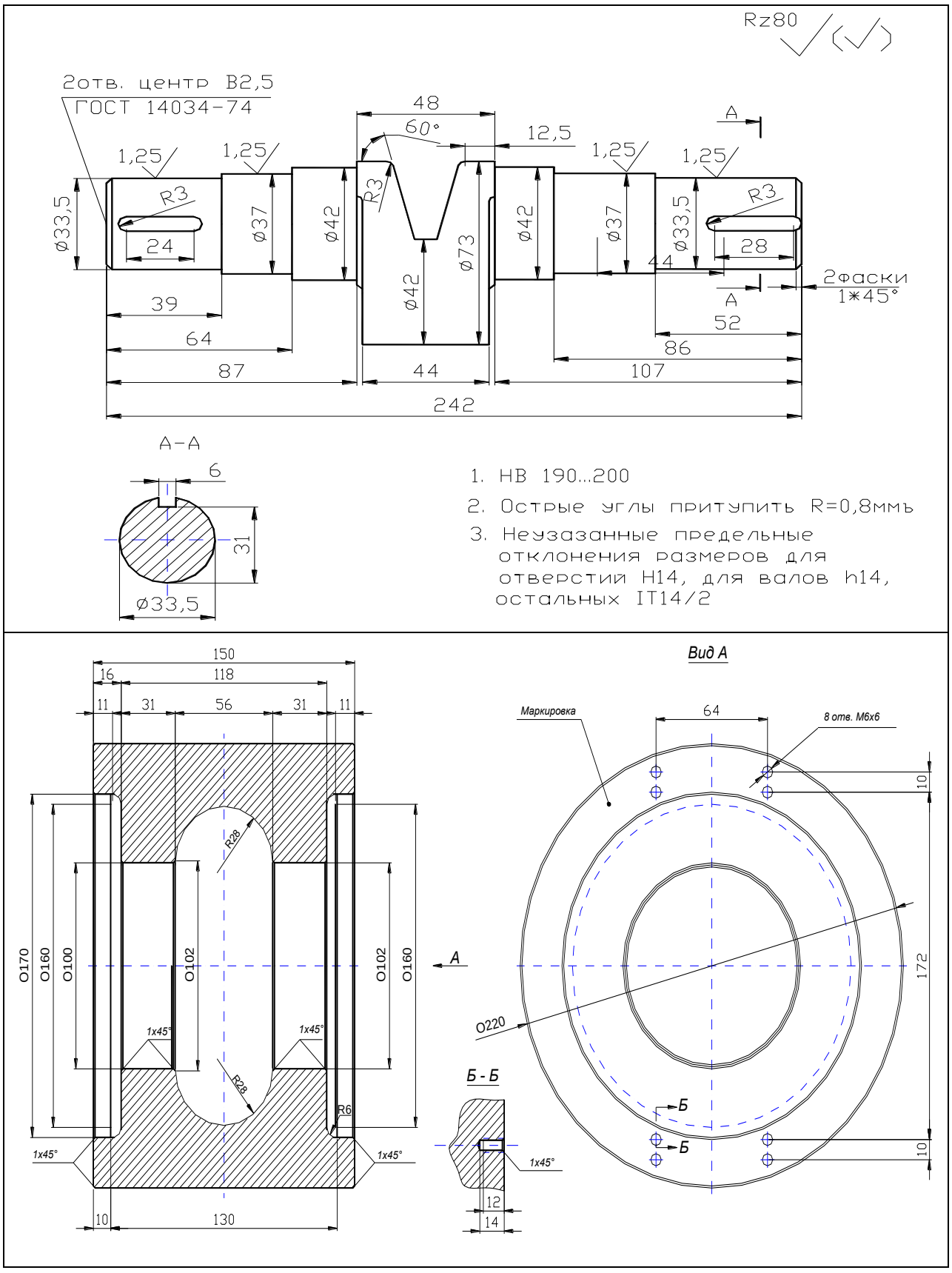


Рис. 6 – Завдання группа ЛС, бригада 3, бригада 4.

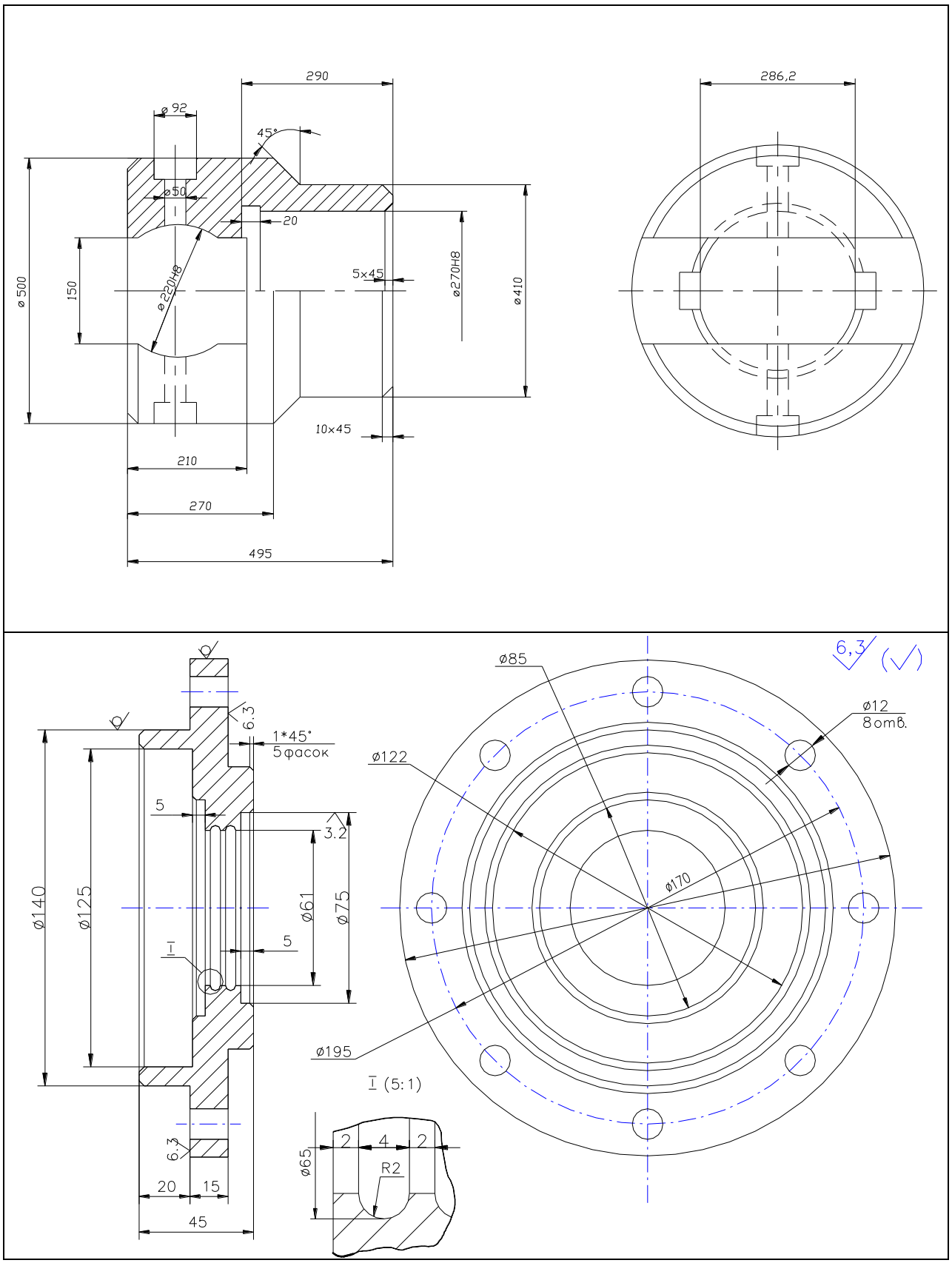


Рис. 7 – Завдання группа ЛС, бригада 5, бригада 6.

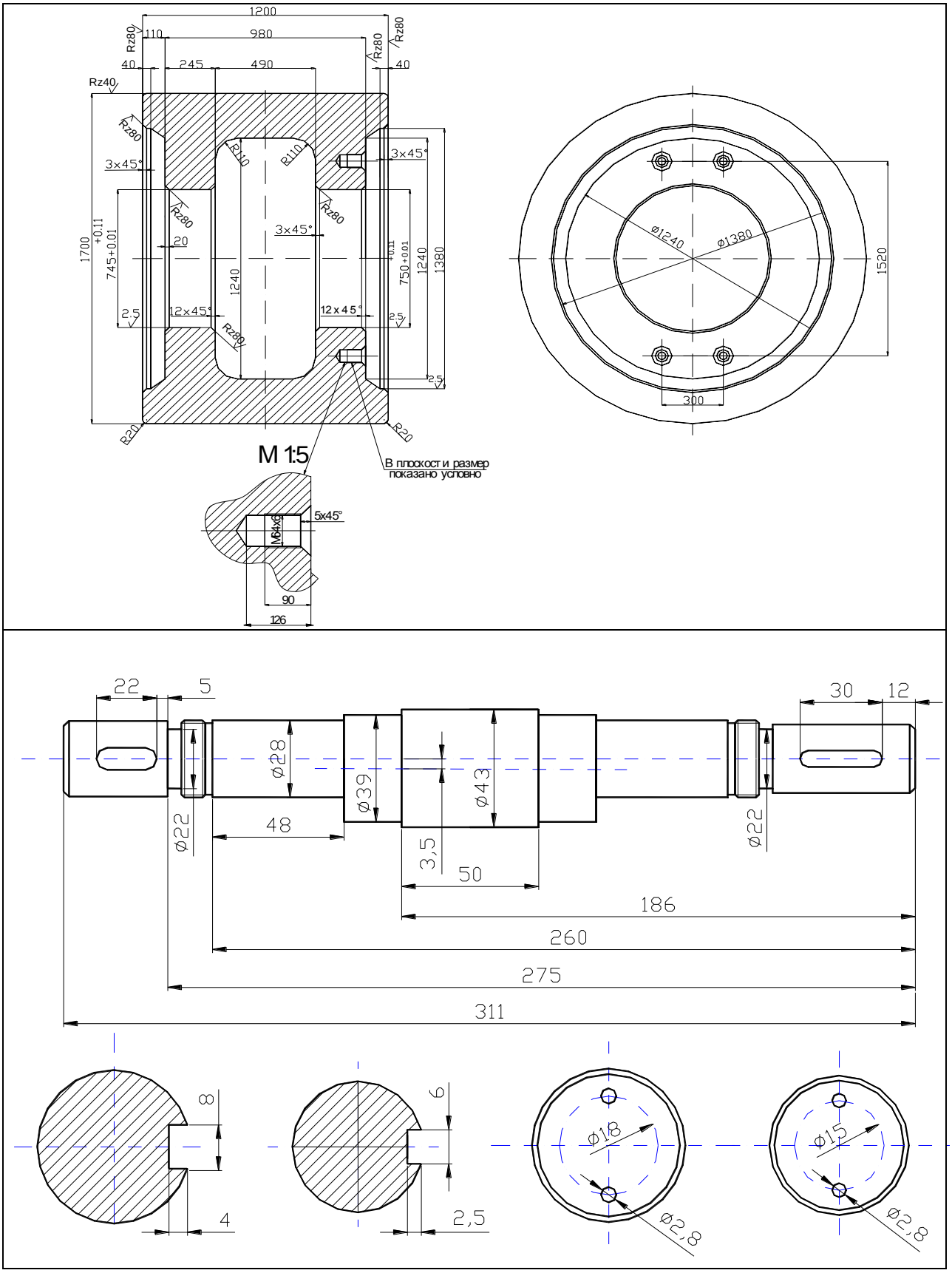


Рис. 8 – Завдання группа ЛС, бригада 7, бригада 8.

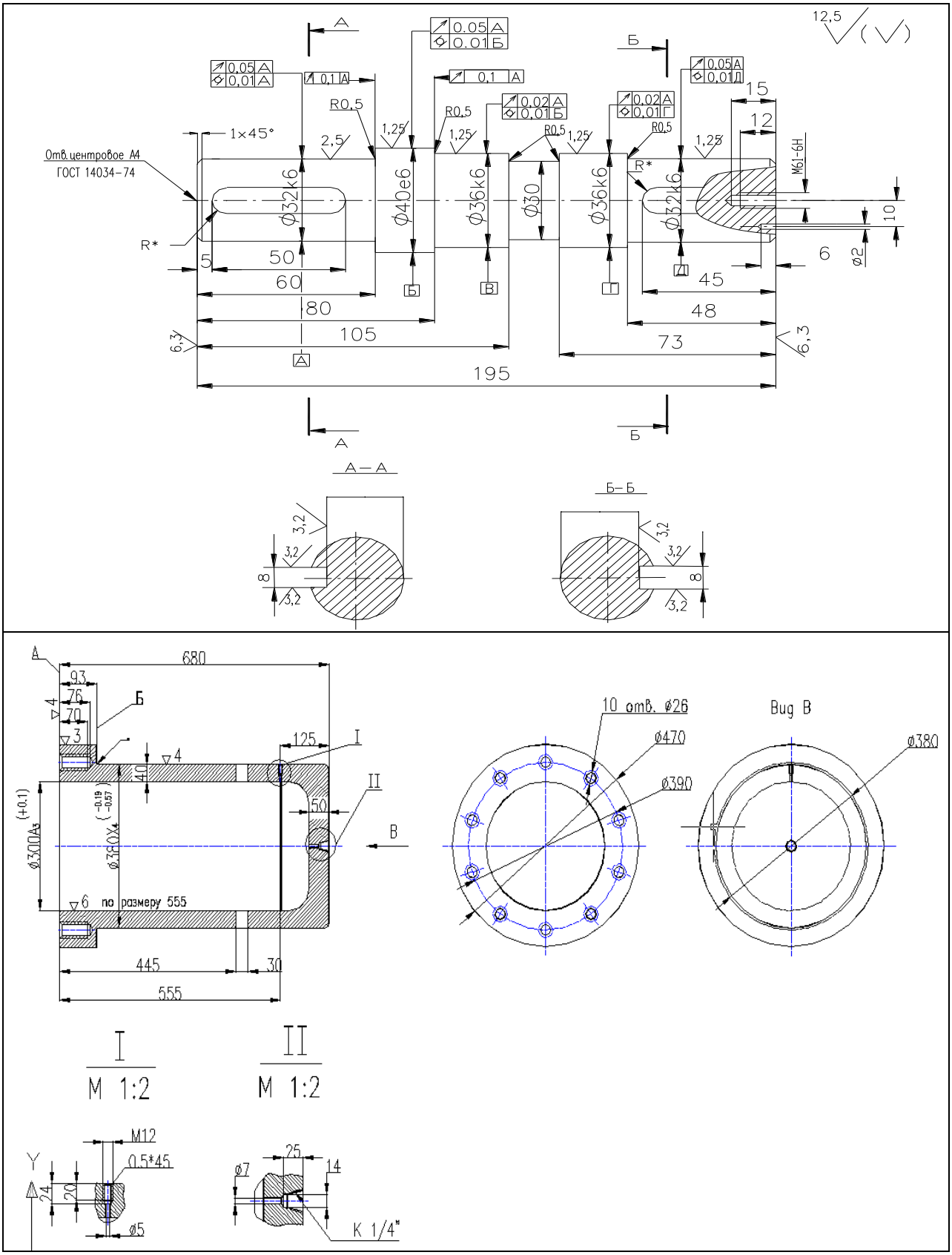


Рис. 9 – Завдання группа ЛС, бригада 9, бригада 10.

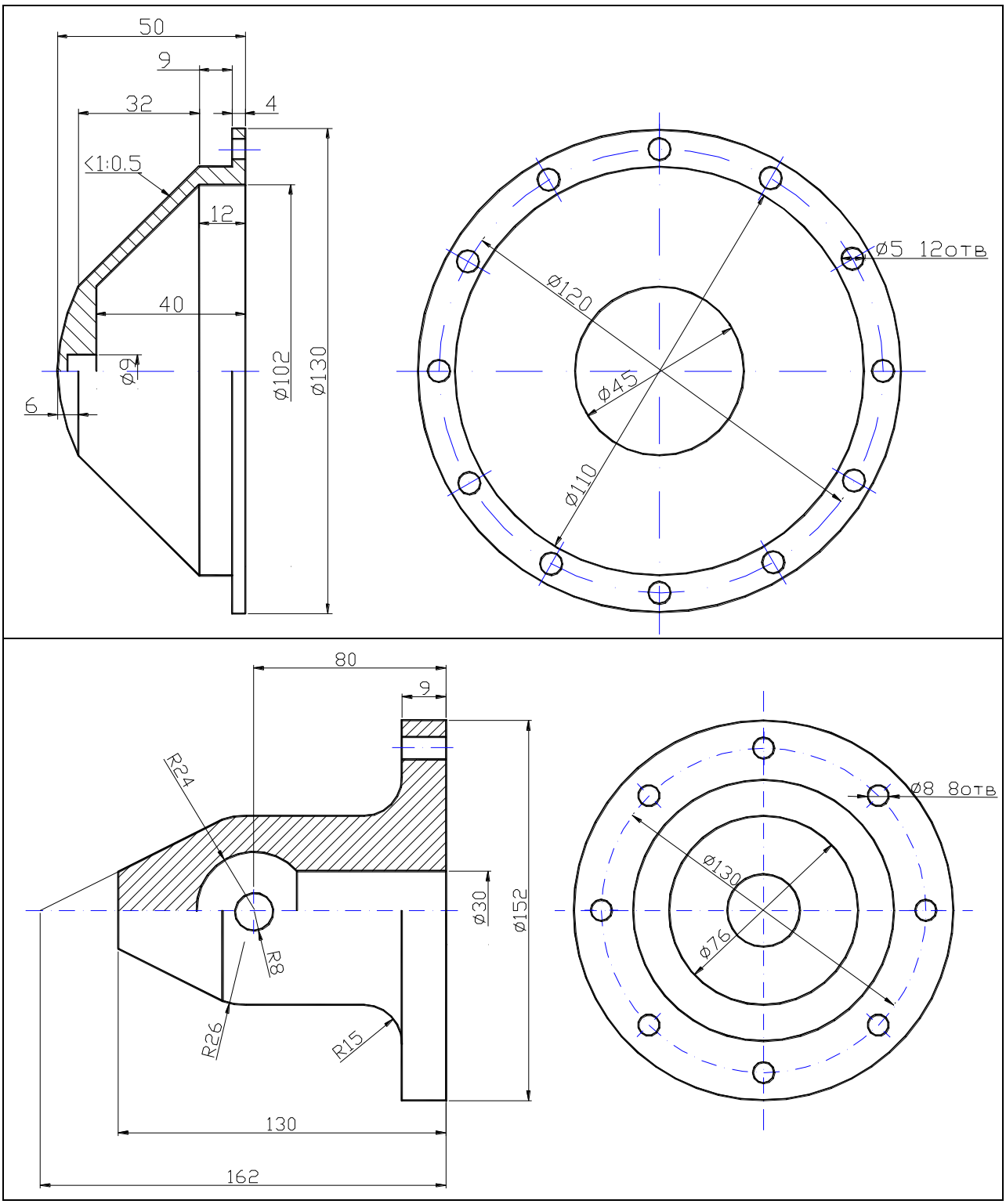


Рис. 10 – Завдання группа ЛС, бригада 11, бригада 12.

Завдання для групи ЛД

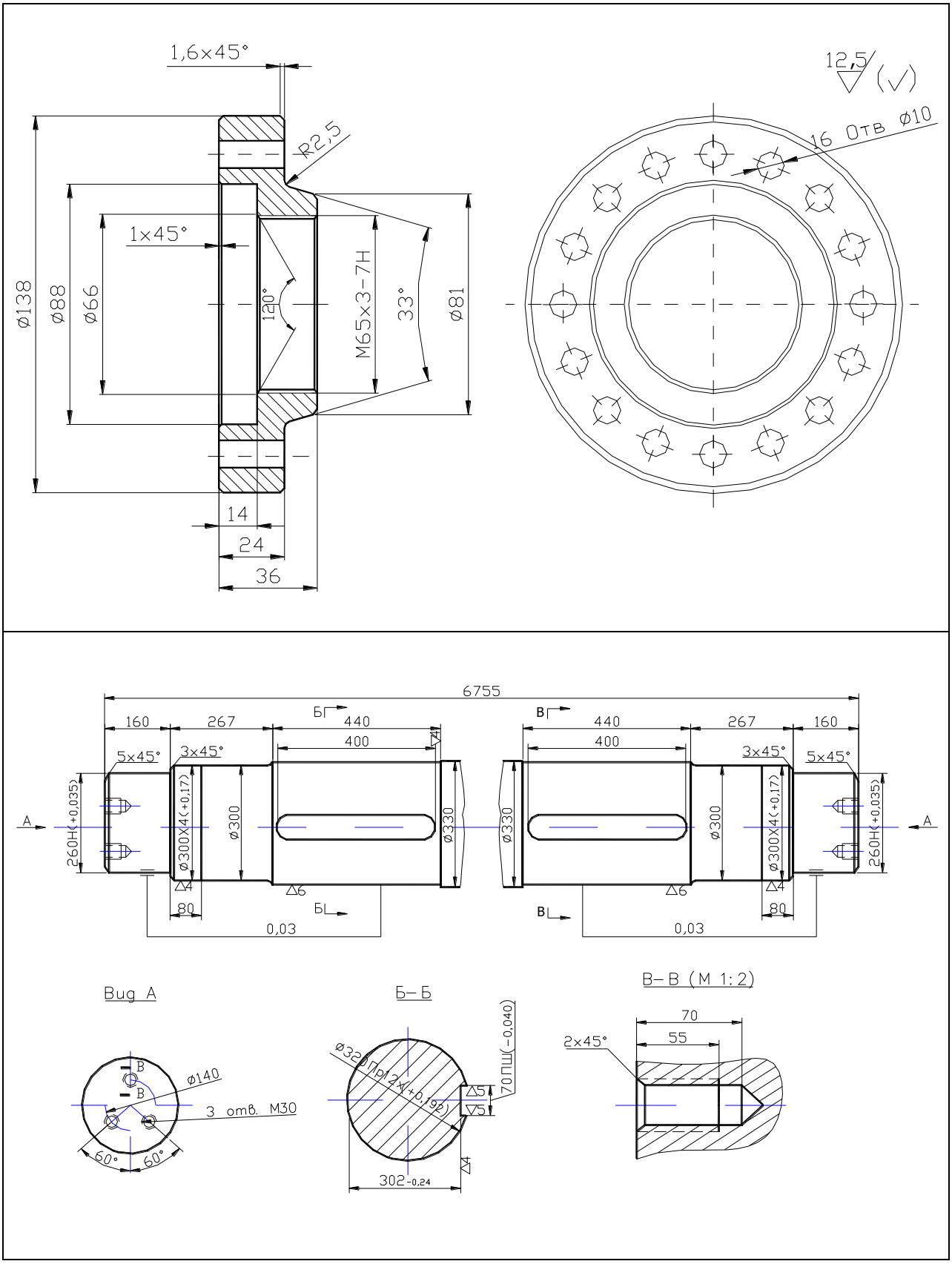


Рис. 11 – Завдання группа ЛД, бригада 1, бригада 2.

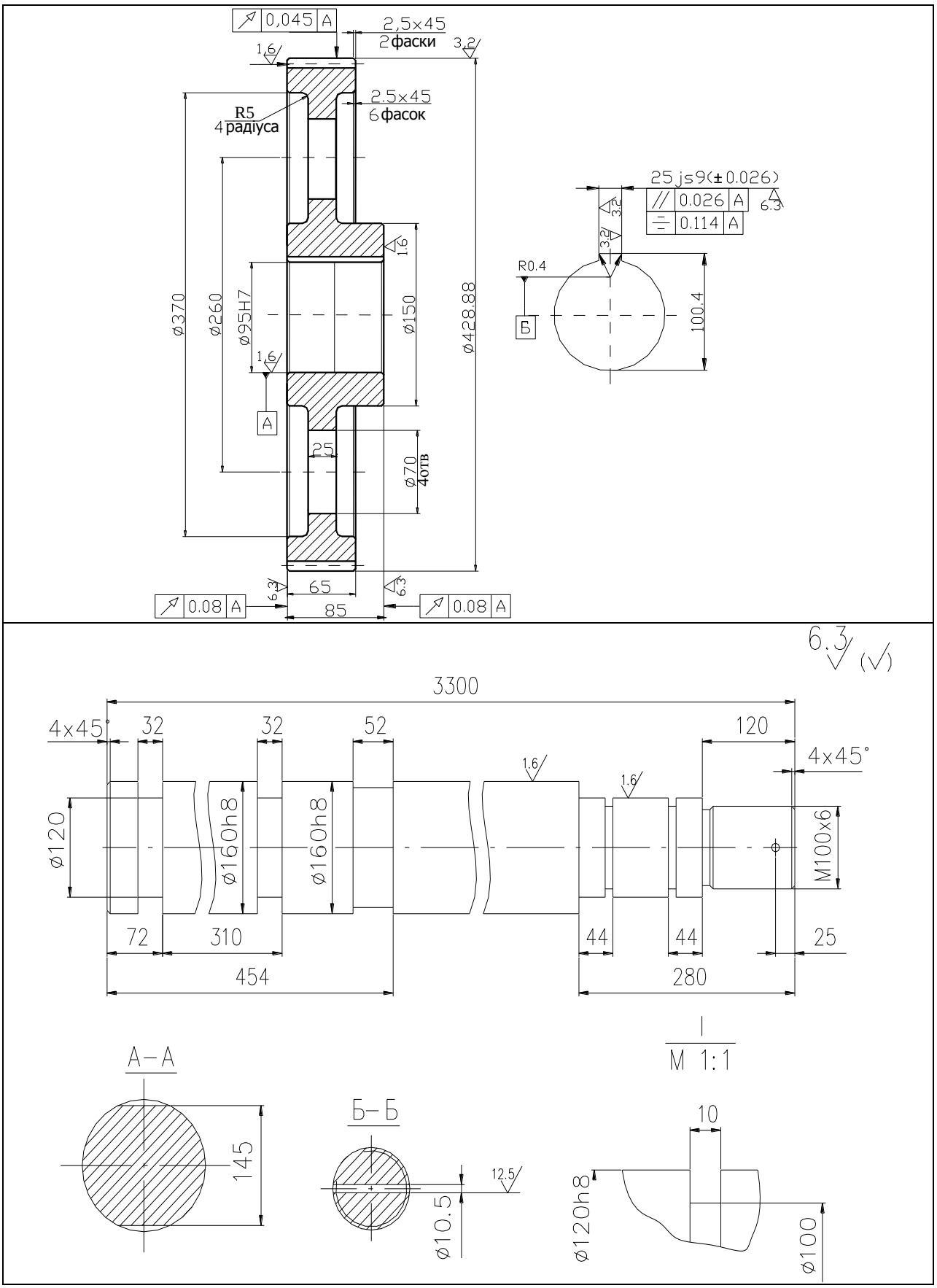


Рис. 12 – Завдання группа ЛД, бригада 3, бригада 4.

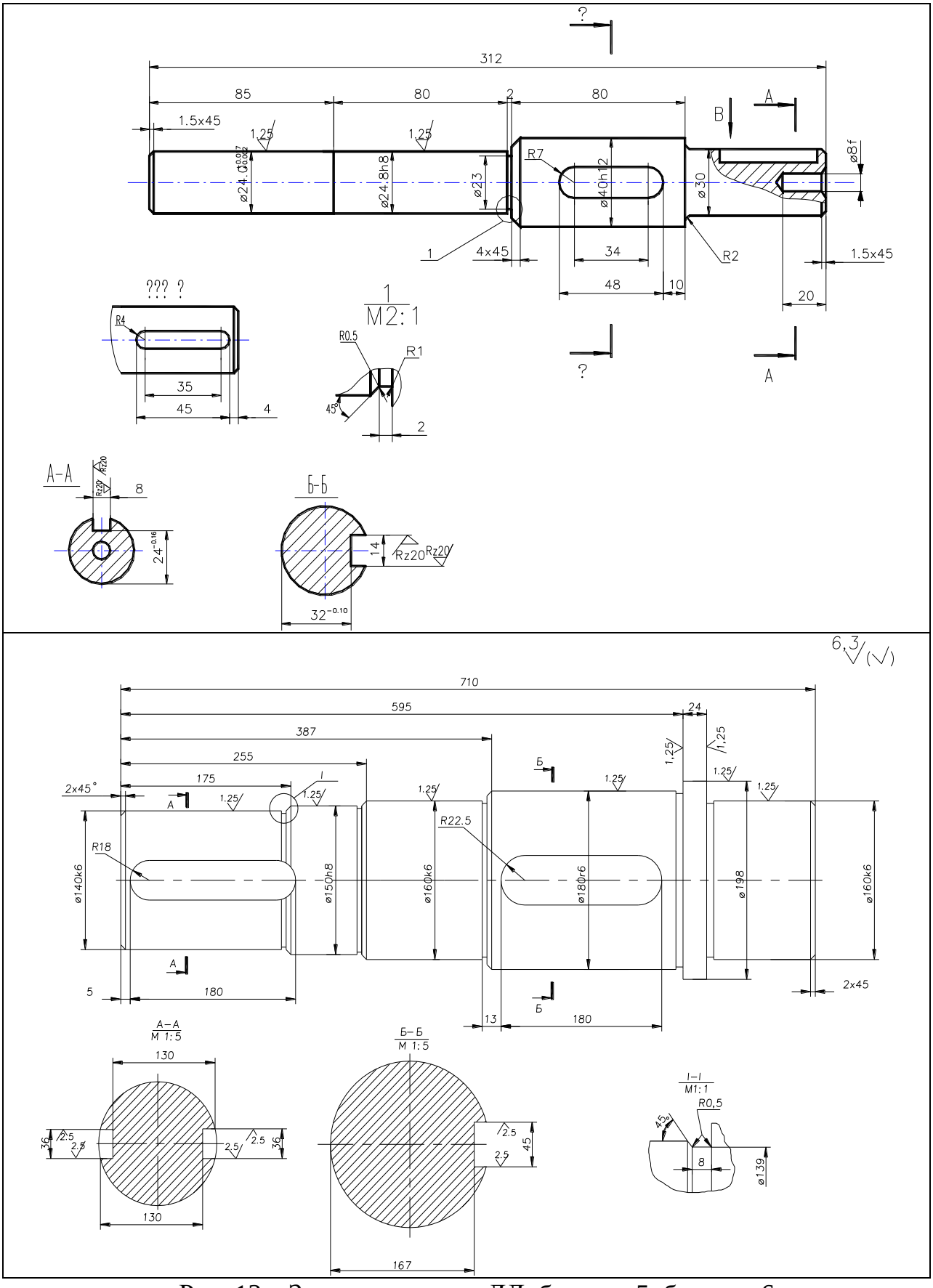


Рис. 13 – Завдання группа ЛД, бригада 5, бригада 6.

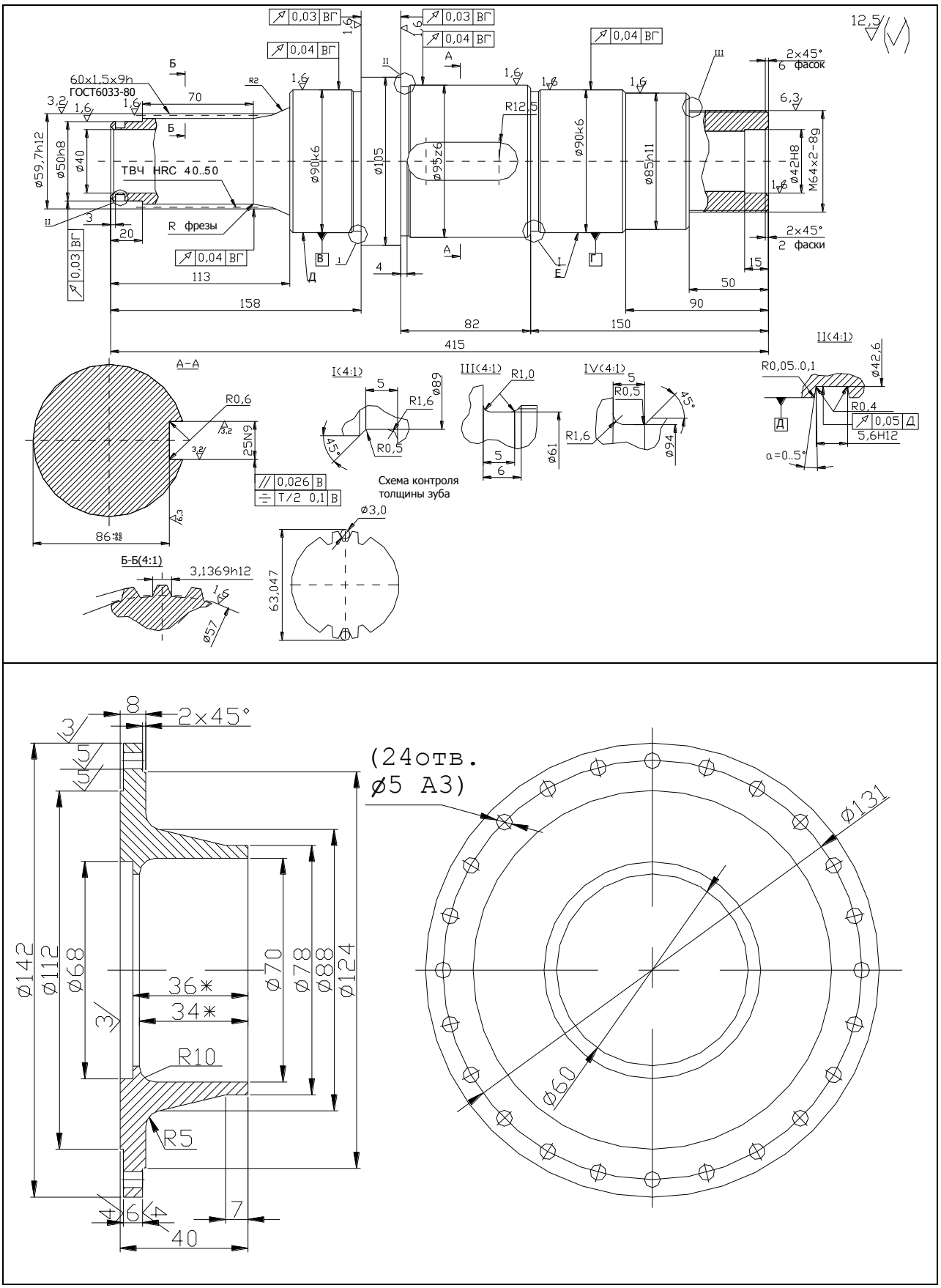
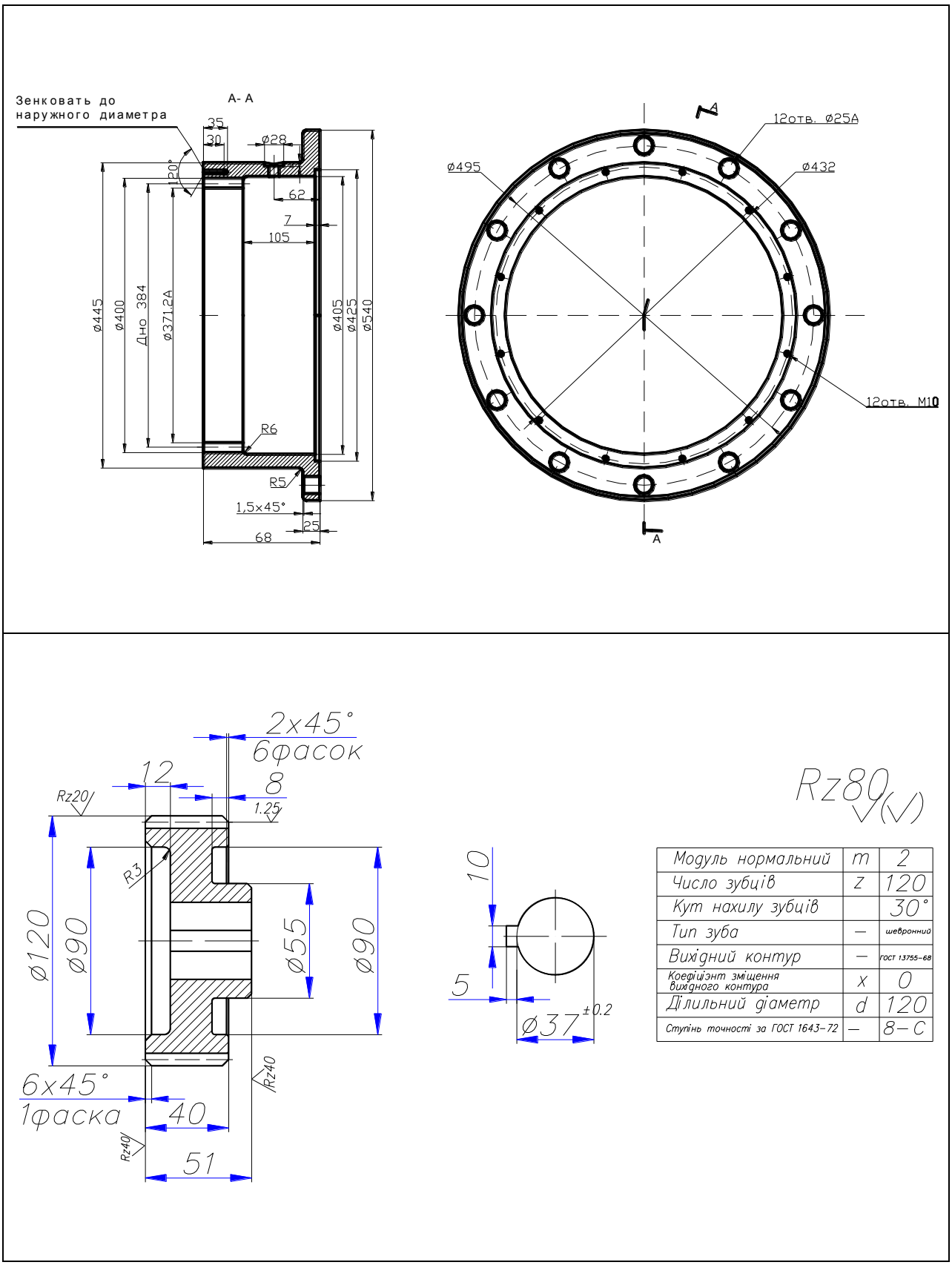


Рис. 14 – Завдання группа ЛД, бригада 7, бригада 8.



Модуль нормальний	m	2
Число зубців	z	120
Кут нахилу зубців		30°
Тип зуба	—	шевронний
Вихідний контур	—	гост 13755-68
Коефіцієнт зміщення вихідного контура	x	0
Ділильний діаметр	d	120
Ступінь точності за ГОСТ 1643-72	—	8-С

Рис. 15 – Завдання группа ЛД, бригада 9, бригада 10.

Завдання для групи ЛП

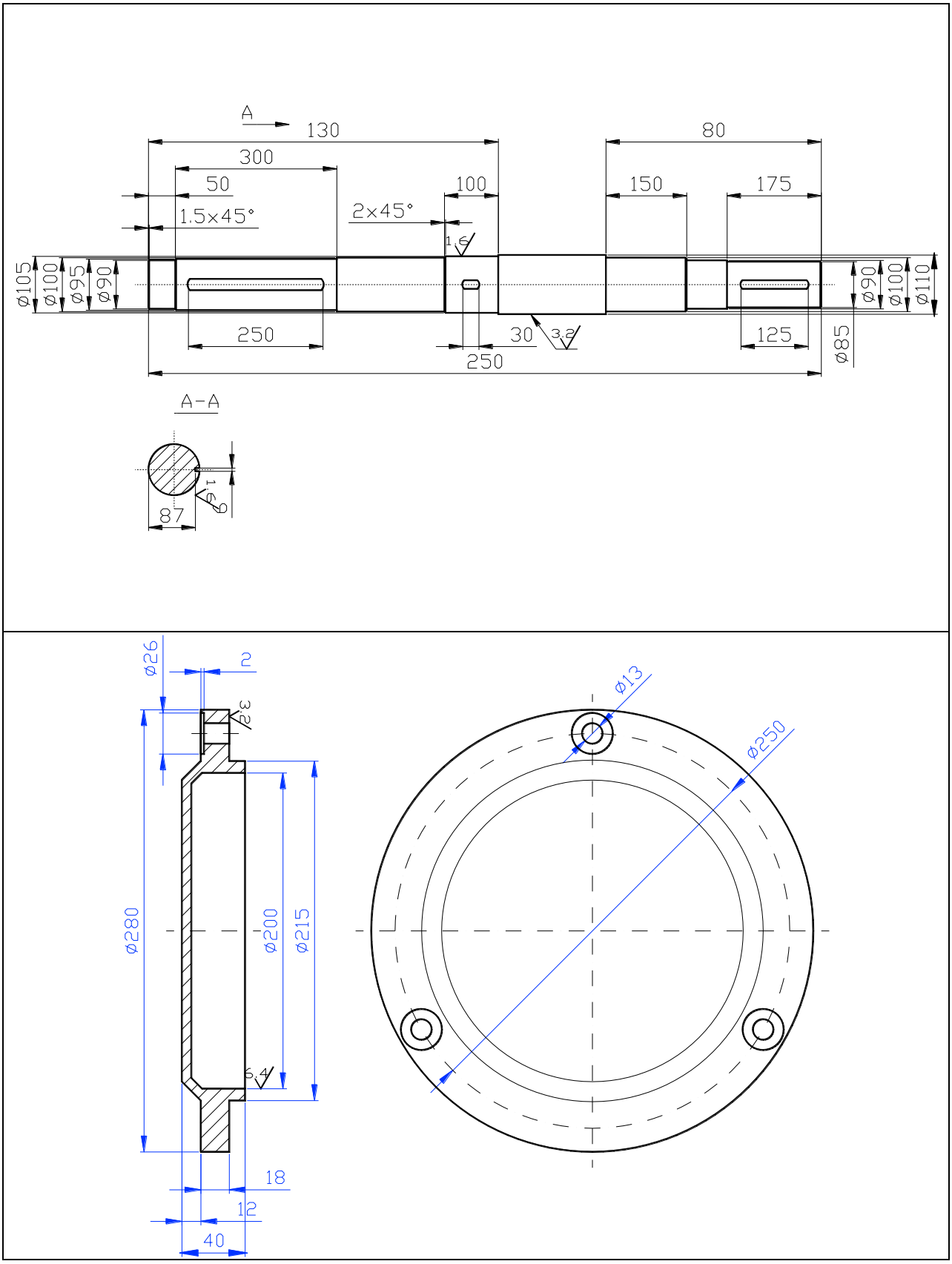


Рис. 16 – Завдання группа ЛП, бригада 1, бригада 2.

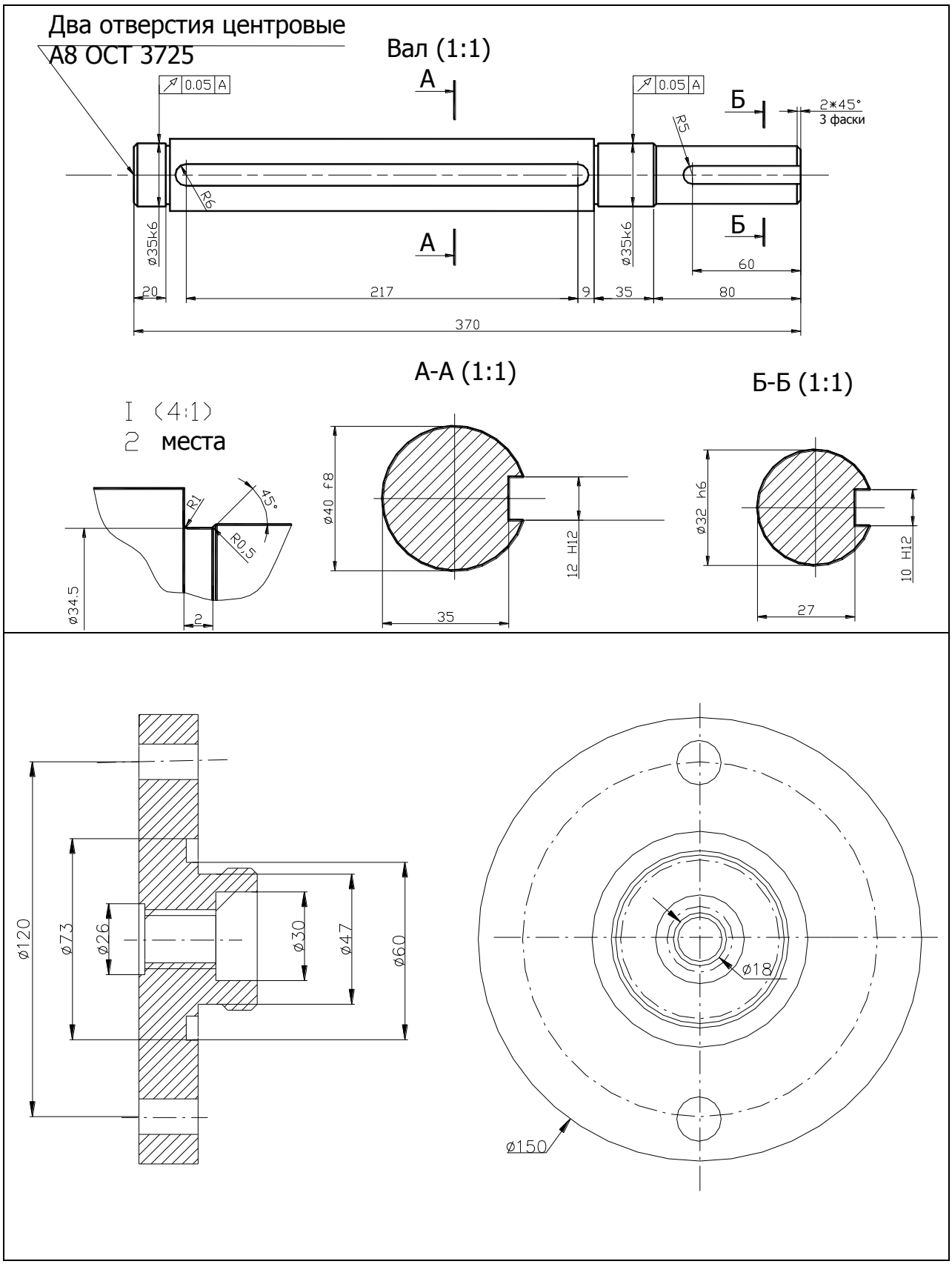


Рис. 17 – Завдання группа ЛП, бригада 3, бригада 4.

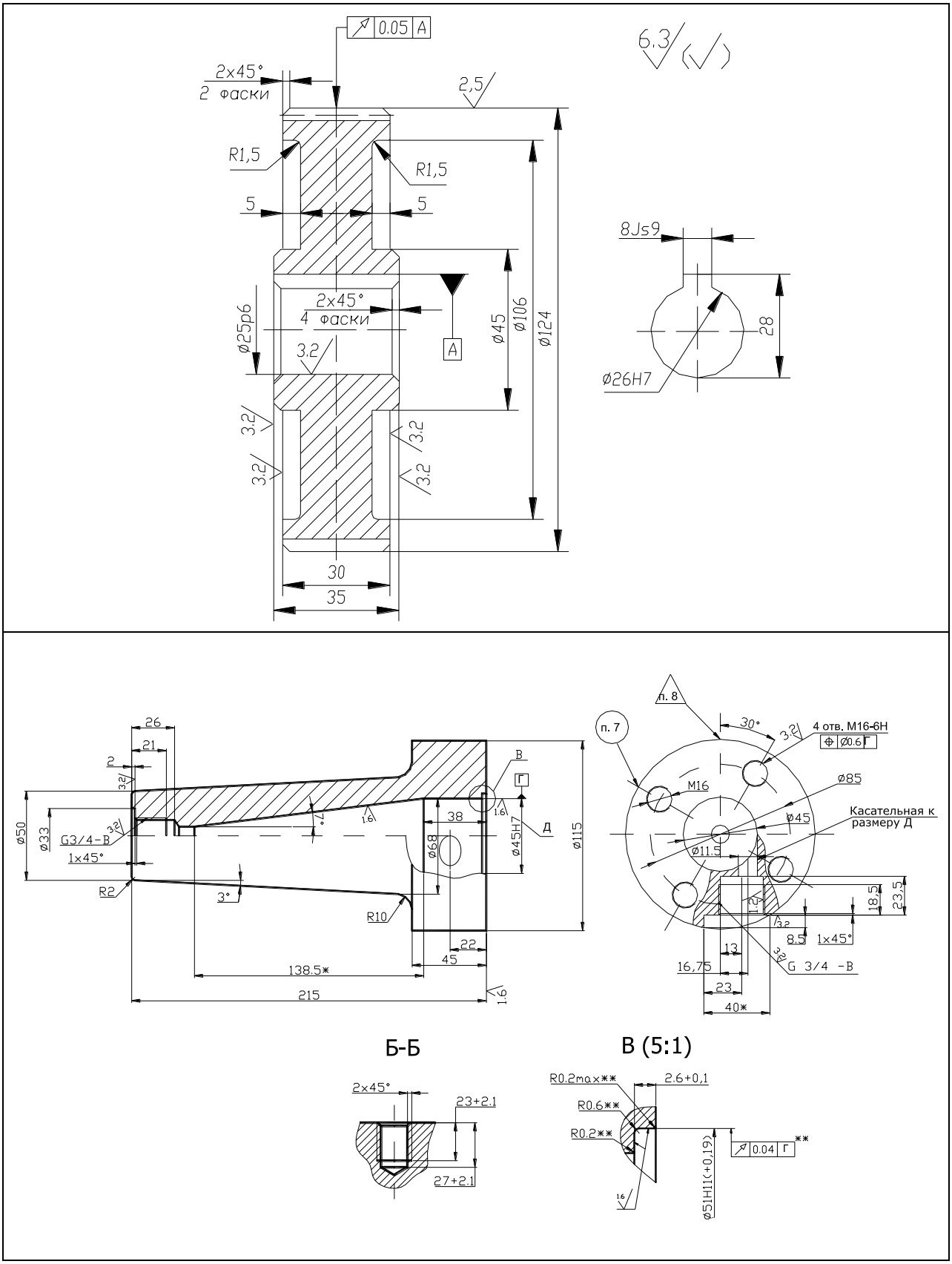


Рис. 18 – Завдання группа ЛП, бригада 5, бригада 6.

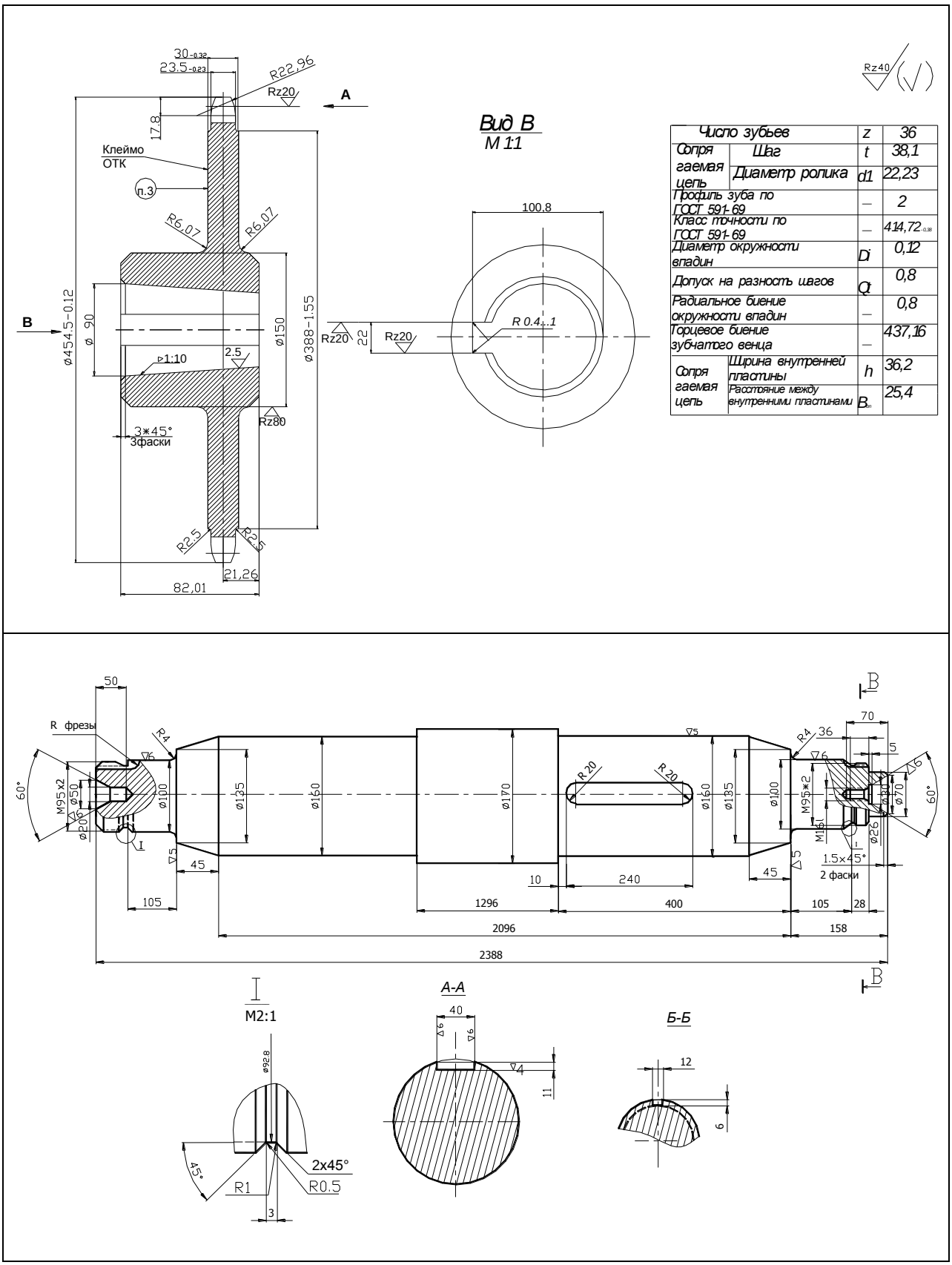


Рис. 19 – Завдання группа ЛП, бригада 7, бригада 8.

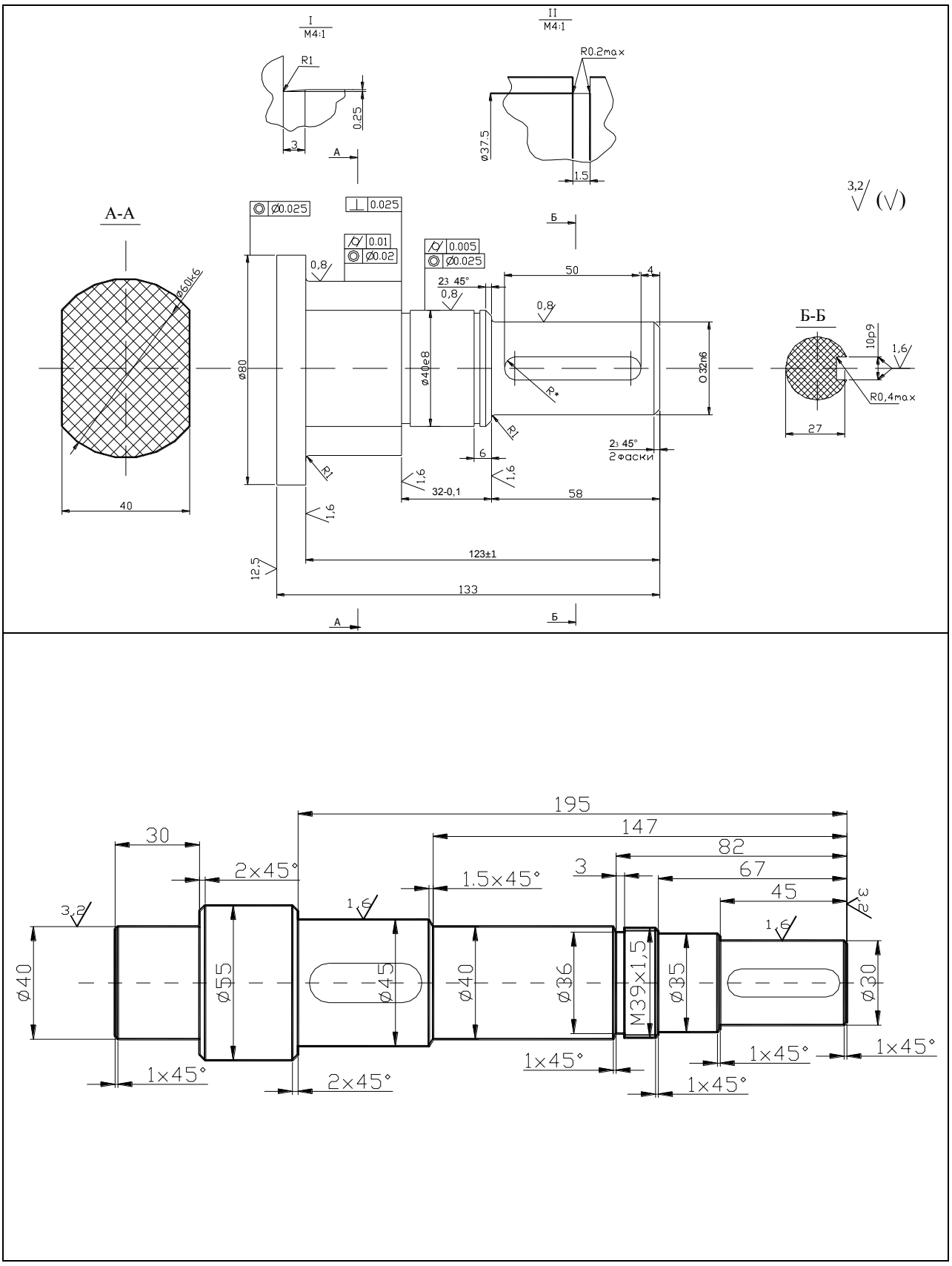


Рис. 20 – Завдання группа ЛП, бригада 9, бригада 10

Завдання для групи ЛУ

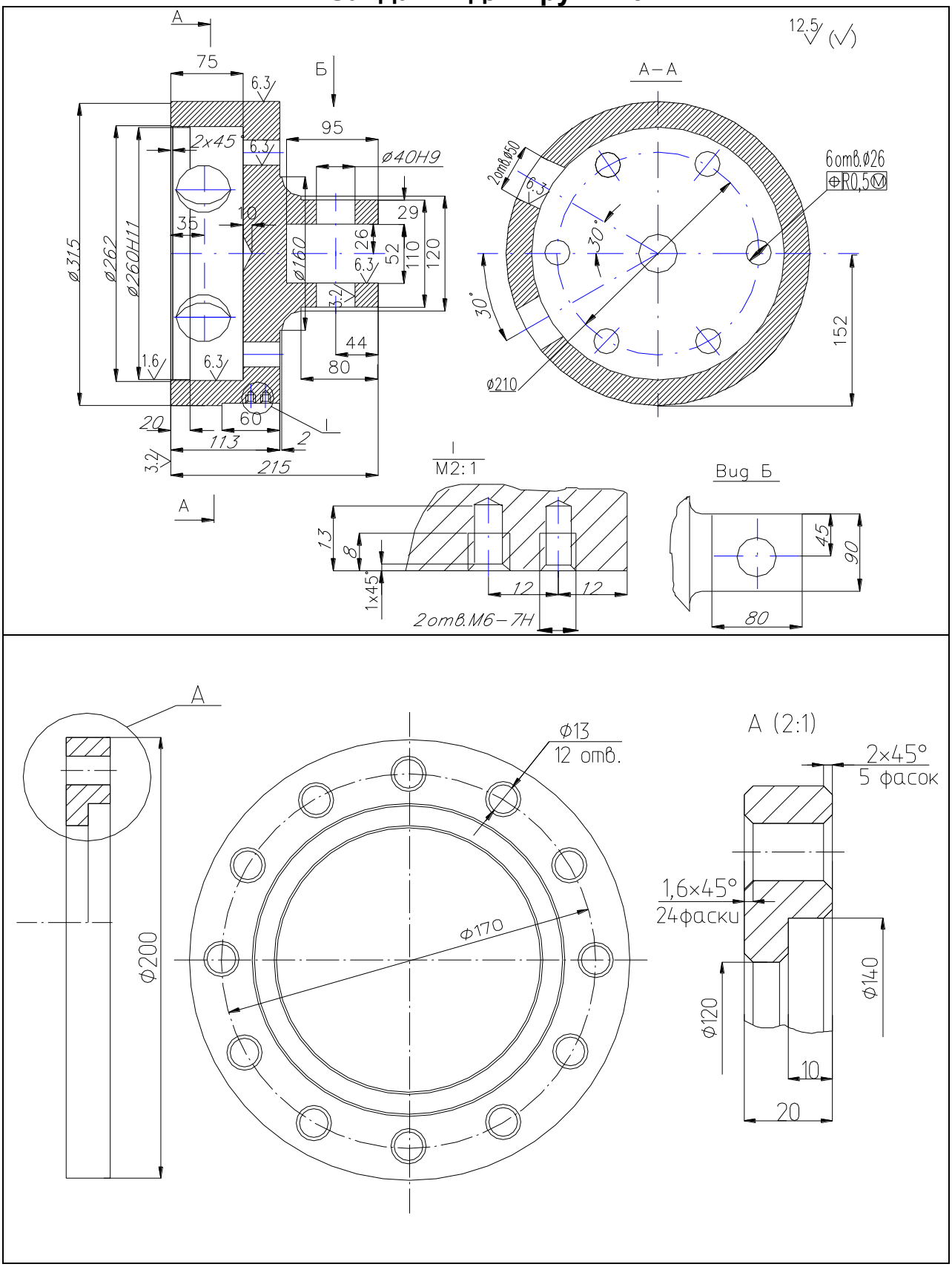


Рис. 21 – Завдання группа ЛУ, бригада 1, бригада 2

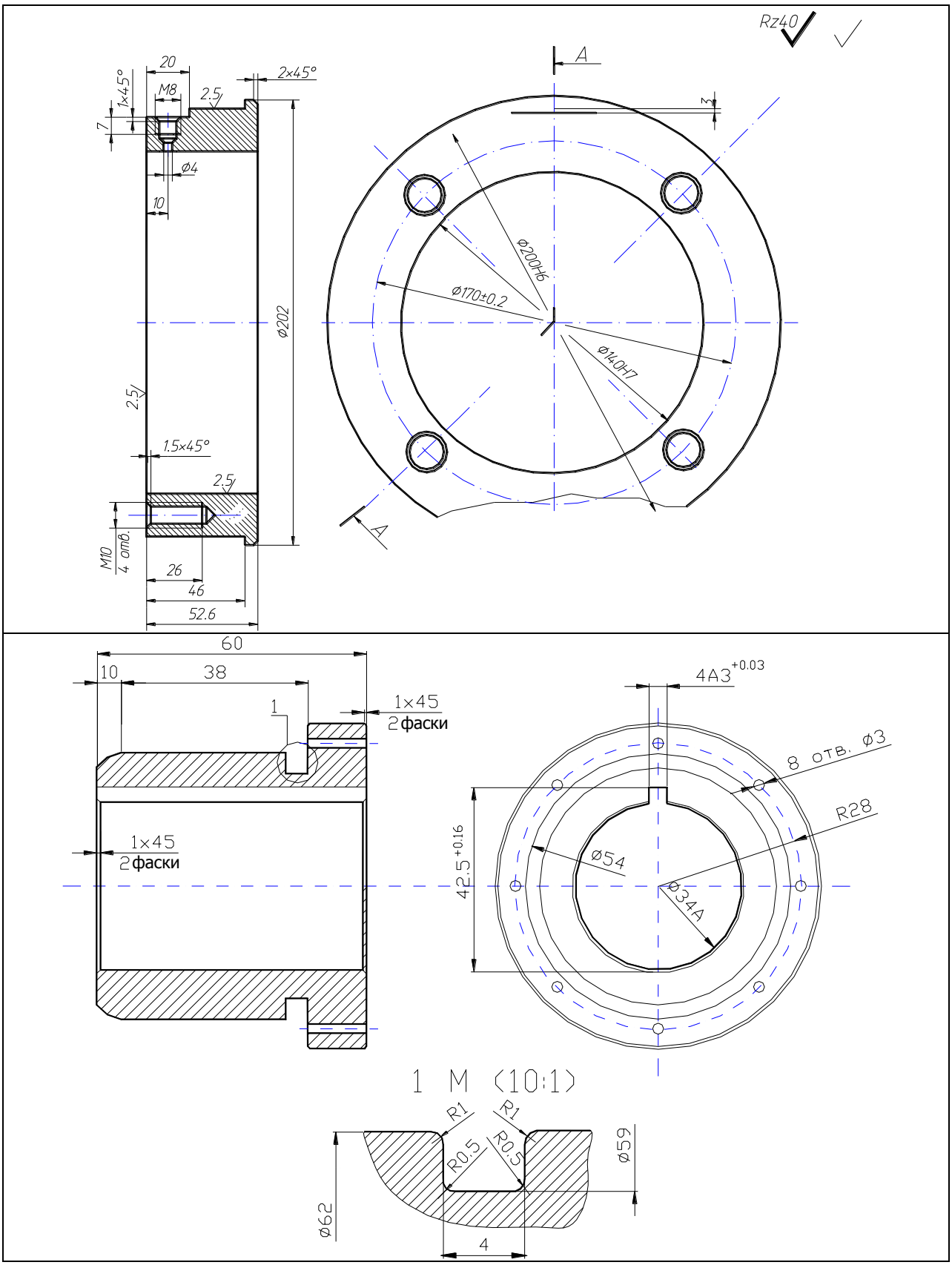


Рис. 22 – Завдання группа ЛУ, бригада 3, бригада 4

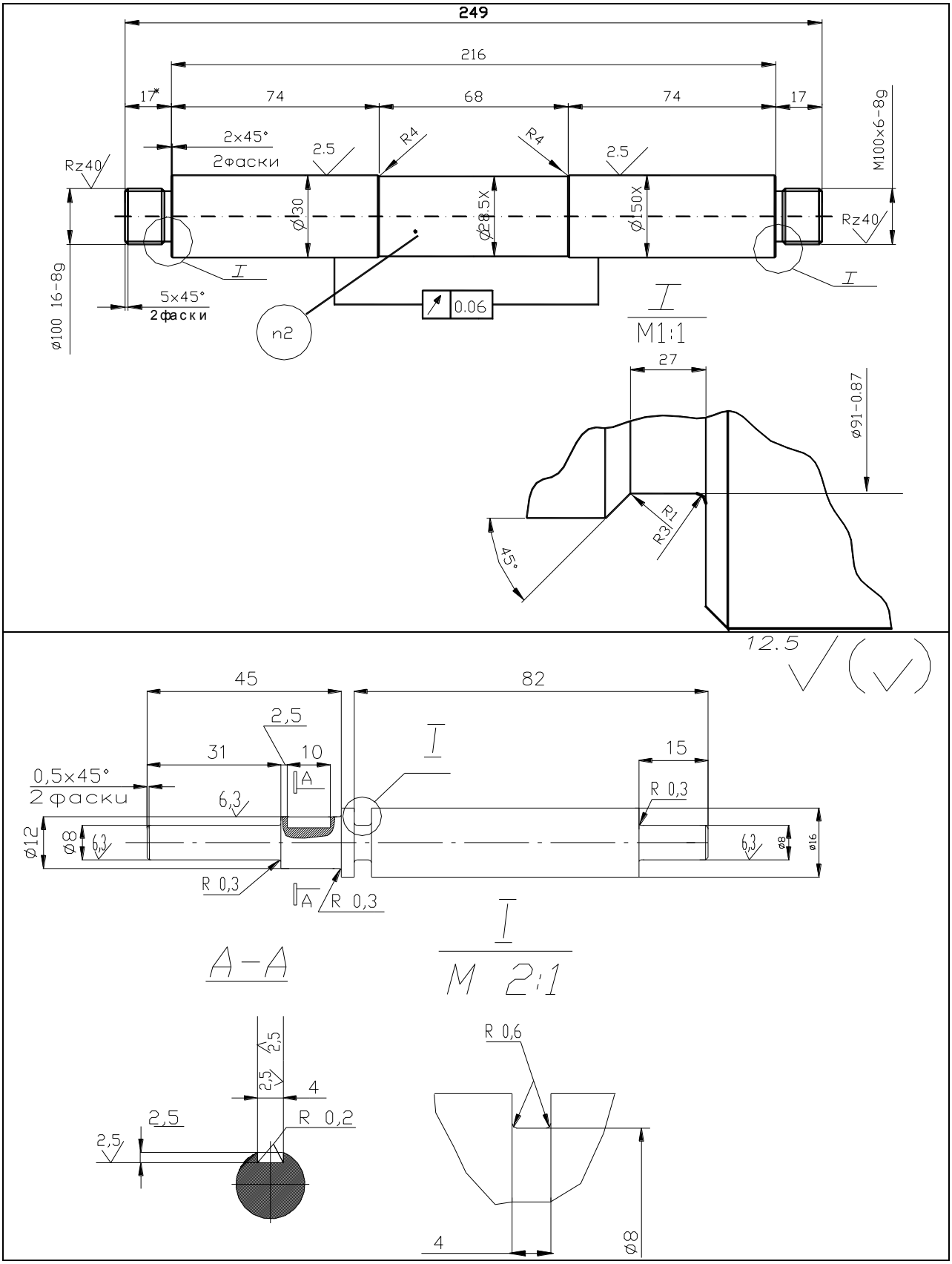


Рис. 23 – Завдання группа ЛУ, бригада 5, бригада 6

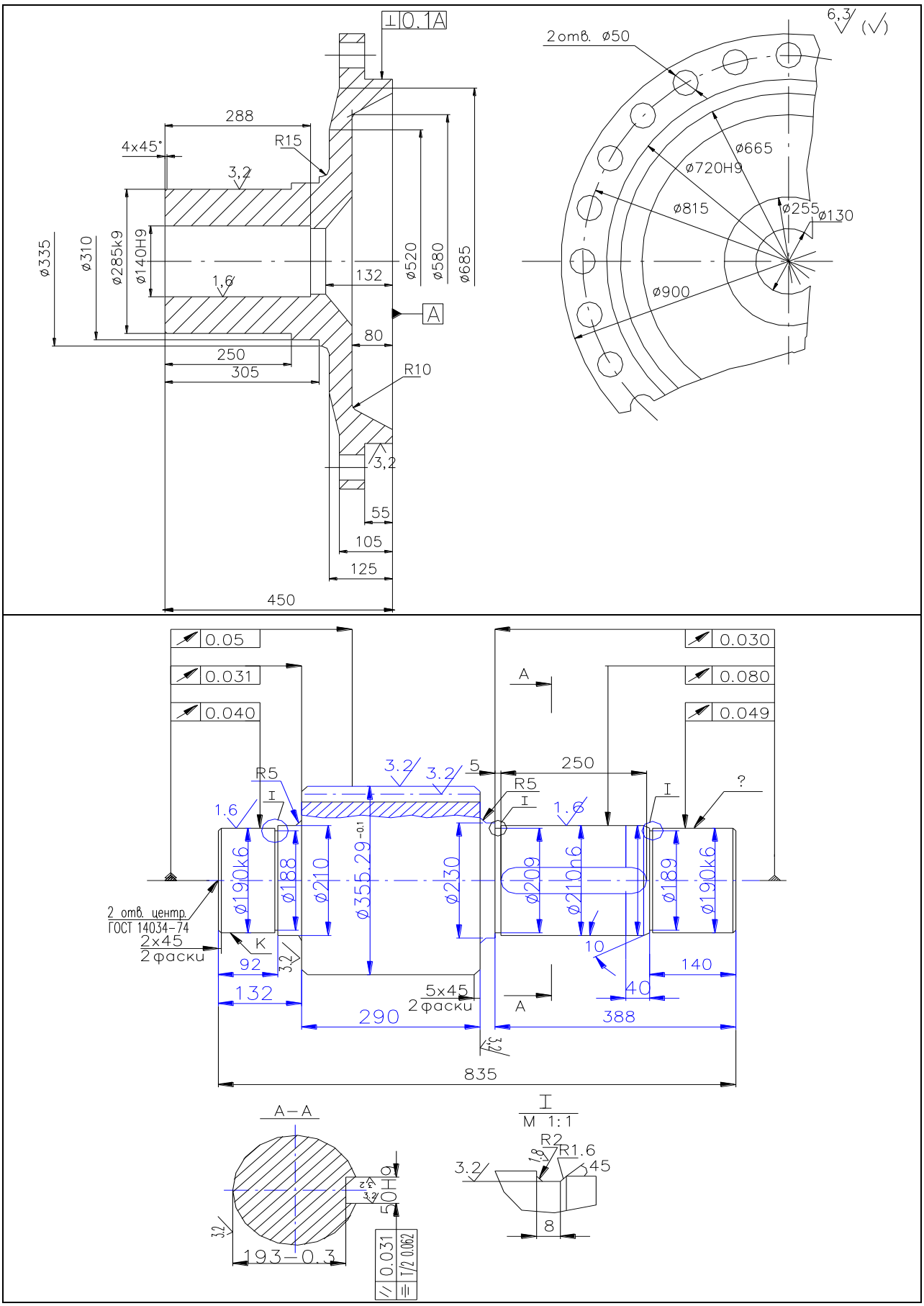


Рис. 24 – Завдання группа ЛУ, бригада 7, бригада 8

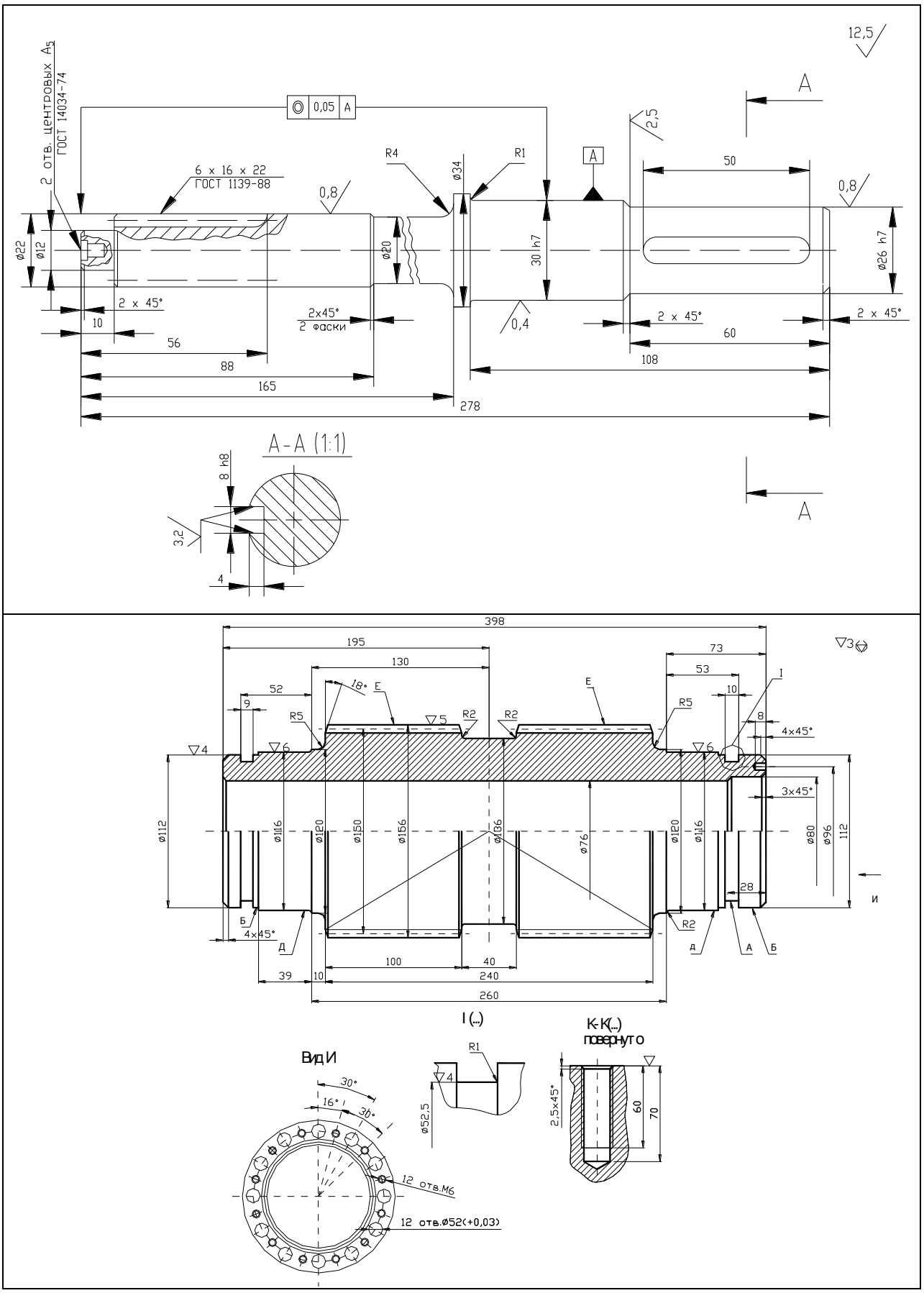


Рис. 25 – Завдання группа ЛУ, бригада 9, бригада 10

3. DCL – МОВА КЕРУВАННЯ ДІАЛОГОМ

Діалогові вікна в AutoCAD визначаються текстовими файлами, написаними мовою DCL. Ці файли мають розширення .dcl і містять опис способу відображення вікна на графічному моніторі та його склад: клавіші, списки, ковзні шкали, кнопки вибору тощо. Правила конструювання діалогових вікон задають обмеження на розмір і місце розташування. Розташування елементів вікна дуже схоже на розташування абзаців у сформатованому тексті, тому не потрібно задавати точні координати фрагментів вікон.

Діалогове вікно з меню AutoCAD можна викликати через функцію AutoLISP чи зовнішню функцію, які керують діалоговим вікном.

У кожному діалоговому вікні міститься одне чи кілька полів, що визначають його функції. До основних типів полів належать базові поля: клавіші, кнопки, текстові поля, ковзні шкали, поля списків, поля зображень. Поля можуть обрамляти рамки. Можна комбінувати поля, створюючи ряди і стовпчики.

Кожне діалогове вікно розглядають як деревоподібну структуру, вершина якої представляється мовою DCL як *dialog*. Керування вікнами здійснюють атрибути полів. Можна визначити нові поля (прототипи) і групи полів, що не зв'язані зі звичайними діалоговими вікнами. На прототипи можна посилатися та змінювати у разі необхідності їхні атрибути і попередньо визначені поля. Об'єднання використовують тільки для зовнішніх посилань. Їхні атрибути змінювати не можна.

Опис діалогового вікна здійснюють мовою DCL, що відбиває його деревоподібну структуру.

3.1 Попередньо визначені активні поля

Попередньо визначені поля безпосередньо підтримують засоби програмувальних діалогових вікон AutoCAD. Їхні визначення містяться у файлі *base.dcl* у вигляді коментарів. Під час вибору активного поля діалогове вікно сповіщає про це додаток (програму, написану мовами *Lisp* чи C++), що керує діалоговим вікном. Подібна операція називається *викликом з поверненням*. Будь-яке попередньо визначене активне поле має відповідний видимий зовнішній (наприклад, розкриття списку чи закриття діалогу при вказівці кнопки *OK*) чи внутрішній ефект. У цьому випадку виробляється код причини, зміст якого залежить від типу ініційованого поля.

BUTTON

Клавіша – Button: Поле діалогового вікна, що нагадує звичайну клавішу клавіатури. Кнопки призначені для виконання дій, що мають зовнішній ефект для користувача: закриття діалогового вікна, поява ще одного вікна і т. д. Усі діалогові вікна містять кнопку *OK* чи її еквівалент, що дозволяє виконати дії вікна, а багато вікон – кнопку скасування, що

дозволяє користувачу залишити діалогове вікно без внесення яких-небудь змін.

Атрибути

label: Мітка – рядок, узятий у лапки (значення за замовчуванням немає). Визначає напис, що з'являється на кнопці.

is_default: Можливі значення: *true* чи *false* (за замовчуванням – *false*). Якщо значення *true*, кнопка є кнопкою за замовчуванням і вказується, коли користувач натискає клавішу <ENTER>. Якщо користувач працює в текстовому полі чи списку кнопки зображення, що мають атрибут *allow_accept*, установлений у *true*, кнопка за замовчуванням також вибирається указанням кнопки чи виконанням подвійного указання за допомогою кнопки вибору пристрою вказівки. Кнопку за замовчуванням не вибирають при натисканні кнопки виконання, якщо підсвічена інша кнопка: у цьому разі буде обрана підсвічена кнопка. Кнопку за замовчуванням виводять на екран іншим способом, наприклад, за допомогою додаткової рамки. Тільки одна кнопка в діалоговому вікні може мати значення атрибута *is_default*, що дорівнює *true*.

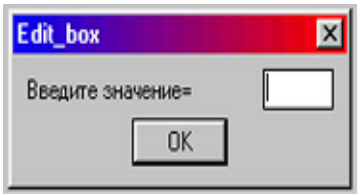
is_cancel: Можливі значення: *true* чи *false* (за замовчуванням – *false*). Якщо значення *true*, кнопку вибирають під час натискання клавіші скасування (наприклад, <ESC> чи <Ctrl+C>).

Тільки одна кнопка в діалоговому вікні може мати значення атрибута *is_cancel*, що дорівнює *true*. Кнопка, що має встановлений у *true* атрибут *is_cancel*, закриває діалогове вікно після виконання чи дії виклику з поверненням.

Приклад:

:button{ label = "Теор.чертеж"; key = "TEORCS";is_default = true;}

EDIT_BOX



Текстове поле – Edit_box: Поле, в яке користувач може вводити чи редагувати текстовий рядок. Якщо текст, який уводять, перевищує розміри текстового поля, його можна “прокрутити” у горизонтальному напрямку.

Атрибути

Label: Значення – рядок, узятий у лапки (за замовчуванням – порожні “ “). Текст виводиться ліворуч від поля. Якщо він визначений, атрибут вирівнюється вліво по ширині текстового поля.

Edit_width: Можливі значення: ціле чи дійсне число. Ширина частини поля, що редагується, обмеженого рамкою текстового поля в одиницях ширини символу. Якщо атрибут не зазначено чи встановлено у нуль і ширину поля не обмежено, рамка розширюється настільки, наскільки це можливо. Якщо значення атрибута не дорівнює нулю, рамка вирівнюється вправо всередині зайнятого полем простору. Якщо необхідно витягнути поле,

можна при компонуванні вставити пробіли між атрибутом і частиною, що редагується, поля засобами *PDB*.

Edit_limit: Значення – ціле число (за замовчуванням – 132); максимум – 256. Максимум – максимальна кількість символів, яку може ввести користувач у текстове поле. Коли цієї кількості досягли, драйвер монітора відкидає додаткові символи (крім <Backspase> чи <Del>) і видає сигнал.

Value: Значення – рядок, узятий у лапки (за замовчуванням – порожні “ “). Вихідне значення – текстове, розміщене в рамці. Значення вирівнюють уліво в редагованій частині поля.

Значення текстового поля завжди закінчується порожнім уведенням (\0 чи *EOS*). Якщо користувач уводить більше символів, ніж установлено атрибутом *edit_limit*, і при цьому необхідно усікти рядок, додається порожній символ.

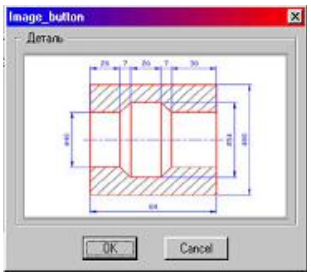
Allow_accept: Можливі значення: *true* чи *false* (за замовчуванням – *false*). Якщо значення дорівнює *true* і користувач натиснув клавішу виконання (як правило, клавішу <ENTER>), кнопка за замовчуванням (якщо є) стає “натиснутою”. Кнопкою за замовчуванням уважають ту, атрибут *is default* якої встановлений у *true*.

Приклад:

:edit_box{label="Толщ.линь";edit_width=6;key="hlin";}

IMAGE_BUTTON

Кнопка зображення – Image_button: Поле з графічним зображенням. Коли користувач вибирає клавішу зображення, програма одержує координати точки, у якій відбувся вибір. Це доцільно, якщо рисунок невеликий і вибір різних його областей має різне значення.



Атрибути

Color: Колір фону малюнка, визначений як номер кольору AutoCAD чи одне із символьних імен (за замовчуванням – 7):

dialog_line – поточний колір діалогового вікна;

dialog_forground – поточний колір символів діалогового вікна (для тексту);

dialog_background – поточний колір фону діалогового вікна;

graphic_background – поточний колір фону графічного екрана; AutoCAD (звичайно аналог 0);

black – колір AutoCAD (чорний) на чорному фоні зображується як білий;

red – колір AutoCAD 1 (червоний);

yellow – колір AutoCAD 2 (жовтий);

green – колір AutoCAD 3 (зелений);

cyan – колір AutoCAD 4 (блакитний);

blue – колір AutoCAD 5 (синій);

magenta – колір AutoCAD 6 (фіолетовий);

white – колір AutoCAD 7 (білий);
graphic_forground на білому фоні зображується як чорний.

Allow_accept: Можливі значення: *true* чи *false* (за замовчуванням – *false*). Якщо значення дорівнює *true* і користувач натиснув клавішу виконання (зазвичай, клавіша <ENTER>), клавіша за замовчуванням (якщо є) стає "натиснутою". Кнопкою за замовчуванням вважають ту, атрибут *is_default* якої встановлений у *true*.

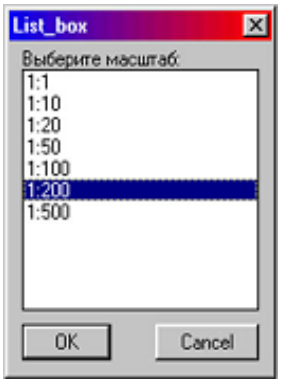
Aspect_ratio: Відношення ширини зображення до його висоти (ширина, ділена на висоту). Якщо значення дорівнює нулю (0.0), то поле зміщується під розміри зображення. Можливе значення з плаваючою точкою (значення за замовчуванням немає).

Полю зображення необхідно присвоїти точні значення атрибутів *width* і *height* чи один із цих атрибутів і значення *aspect_ratio*.

Приклад:

:image_button {key="comp1";color=0; width = 30; aspect_ratio = 0.7;}

LIST_BOX



Поле списку – List_box: Поле, що містить ряди текстових рядків. Наявність такого поля дасть можливість користувачу вибрати необхідний пункт зі списку. Як правило, список має змінну довжину, але поля списків можуть використовуватися і для списків фіксованої довжини (при виборі рядка списку вона підсвічується). Поле списку може містити рядків більше, ніж уміщує поле вікна; у цьому разі праворуч від списку поля з'являється ковзна шкала (її ввімкнено тільки тоді, коли список має більше пунктів, ніж можна ввести за один раз). Переміщуючи візира ковзної шкали, вказуючи на стрілки, можна переглянути весь список. Залежно від додатка користувач може вибрати кілька рядків списку.

Атрибути

Label: Текст, виведений над полем списку. Значення – рядок, узятий у лапки (значення за замовчуванням немає).

Multiple_select: Можливі значення: *true* чи *false* (за замовчуванням – *false*). Якщо *true*, у полі списку можна вибрати і підсвітити кілька пунктів. Якщо *false*, у полі списку може бути обраний тільки один пункт, а вибір іншого пункту скасує вибір попереднього.

List: Визначає початковий вибір рядків, розміщених у полі списку. Рядки відокремлюють символом нового рядка (\n). Символ табуляції (\t) може бути у кожному рядку. Значення – рядок, узятий у лапки (значення за замовчуванням немає).

Tabs: Значення – рядок, узятий у лапки, що містить цілі числа чи числа з плаваючою точкою, відокремлені пробілами (значення за замовчуванням немає). Кожне число – це величина, що визначає розташування табуляцій в

одиницях ширини символу. Ці значення використовують для вертикального вирівнювання стовпців тексту в полі списку (див. діалогове вікно керування шарами AutoCAD, *acad_mylayer* у файлі *acad*. *dcl*).

Value: Значення – рядок, узятий у лапки, може містити нуль (“0”) чи цілі числа, відокремлені пробілами (значення за замовчуванням немає). Кожне ціле (починаючи з нуля) представляє спочатку обраний пункт списку. Якщо значення атрибута *multiple_select* дорівнює *false*, атрибут *value* не може містити більше одного числа.

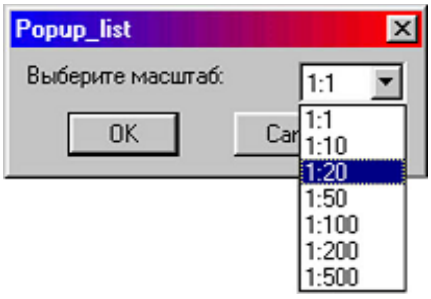
Якщо рядок порожній (“ “), спочатку не буде обраний жоден пункт. У цьому разі цей атрибут можна взагалі не визначати.

Allow_accept: Можливі значення: *true* чи *false* (за замовчуванням – *false*). Якщо значення дорівнює *true* і користувач натиснув клавішу виконання (як правило, клавішу <ENTER>), кнопка за замовчуванням стає “натиснутою”. (Кнопкою за замовчуванням уважають ту, атрибут *is_default* якої встановлений у *true*).

Приклад:

:list_box{width=20;height=35;fixed_height=true;key="list";allow_accept=true;}

POPUP_LIST



Список, що розкривається – Popup_list: Список, що розкривається, еквівалентний полю списку. Коли діалогове вікно з’являється перший раз, список, що розкривається, знаходиться в закритому стані і на екрані має вигляд скоріше кнопки з напрямленою вниз стрілкою праворуч. При виборі напису чи стрілки список розкривається й дозволяє вибрати необхідний пункт у межах вікна. Розкритий список, як правило, відображається цілком, у противному разі він буде мати ковзну шкалу праворуч, що функціонально аналогічна ковзній шкалі поля списку. Коли розкривається закритий список, то вибраний текст з'являється в його видимій частині. У списках, що розкриваються, не можна одночасно вибрати декілька елементів.

Атрибути

Label: Значення – рядок, узятий у лапки (значення за замовчуванням немає). Це текст, виведений ліворуч від списку, що розкривається. Якщо значення атрибута *label* визначено, воно вирівнюється вліво в межах поля *Popup_list*.

Edit_width: Можливі значення: ціле чи дійсне число. Ширина частини списку в одиницях ширини символу, ширина рамки, яка описує єдиний пункт, коли список, що розкривається, закритий. Список не містить необов'язкову мітку ліворуч і стрілку (чи ковзну шкалу) праворуч. Якщо атрибут не зазначено чи встановлено у нуль і ширину поля не задано, рамка

розширюється настільки, наскільки це можливо. Якщо значення атрибута не дорівнює нулю, рамка вирівнюється вправо в середині зайнятого полем простору. Якщо потрібно витягнути поле, можна під час компонування вставити пробіли між атрибутом і частиною поля, що редагується.

Value: Значення – рядок, узятий у лапки, що містить ціле число (за замовчуванням – 0). Ціле число (починаючи з нуля) є поточним обраним пунктом у списку (пункт, що виводиться на екран при закритому списку).

List: Значення – рядок, узятий у лапки (значення за замовчуванням немає). Визначає початковий вибір рядків, розміщених у полі списку. Рядки відкремлено символом нового рядка (\n). Символ табуляції (\t) може бути у кожному рядку.

Tabs: Значення – рядок, узятий у лапки, що містить цілі числа чи числа з плаваючою точкою відокремлені пробілами (значення за замовчуванням немає). Кожне число – це величина, яка визначає розташування табуляцій в одиницях ширини символу. Ці значення використовують для вертикального вирівнювання стовпців тексту у списку, що розкривається.

Приклад:

:popup_list { edit_width=15; key="OPROJ";}

RADIO_BUTTON

Кнопка вибору – Radio_button: Одна чи група кнопок, об'єднаних у стовпці чи ряд вибору. Кнопки вибору функціонально еквівалентні кнопкам радіопанелей: можна вибрати одну кнопку і, поки вона буде натиснута, будь-яка інша кнопка буде вимкнена. Праворуч від кнопки вибору може з'явитися необов'язкова мітка *label*. Кнопки вибору з’являються тільки в стовпцях чи рядах вибору. Якщо кнопка вибору розміщена поза стовпцем чи рядом вибору, повідомляється про помилку.

Атрибути

Label: Текст, виведений ліворуч від кнопки вибору. Значення – рядок, узятий у лапки (значення за замовчуванням немає).

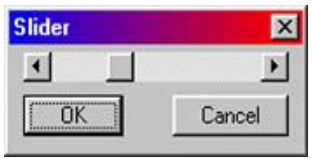
Value: Рядок, узятий у лапки (значення за замовчуванням немає). Якщо значення атрибута “1”, кнопку вибору ввімкнено; якщо “0” – вимкнено; всі інші значення аналогічні “0”.

Якщо з якихось причин кілька кнопок мають атрибут *value*, що дорівнює “1”, буде ввімкнена тільки остання з них. Така ситуація може виникнути лише в користувацьких *DCL*-файлах.

Приклад:

:radio_button{ key="PP"; label = "сечение №1"; width = 1;}

SLIDER



Ковзна шкала – Slider: Використовують для одержання числового значення. Користувач може переміщувати візир ковзної шкали вліво чи вправо (нагору чи вниз) для одержання величини, призначення якої залежить від додатка. Ця величина повертається як рядок, що містить ціле число визначеної точності зі знаком. У додатку це значення можна масштабувати.

Атрибути

Min_value і **Max_value:** Значення – цілі числа, що визначають діапазон повороту значень ковзної шкали. Мінімальне значення за замовчуванням *Min_value* – 0. Максимальне значення за замовчуванням *Max_value* – 10 000. Діапазон має бути визначений знаковим 16-бітовим цілим, тобто від мінус 32 768 до 32 767.

Значення атрибута *Min_value* може бути більше, ніж значення *Max_value.* На деяких платформах це змінює як послідовність появи на екрані цих значень, так і послідовність їхнього повернення ковзною шкалою під час переміщення візира.

Small_increment і **Big_increment:** Значення – цілі числа, що визначають значення, які використовуються при керуванні збільшенням значення ковзної шкали. Значення за замовчуванням для атрибута *Big_increment* – десята від повного діапазону, значення за замовчуванням для *Small_increment* – сота від повного діапазону. Значення мають бути в діапазоні, обумовленому атрибутами *Min_value* і *Max_value*. Ці атрибути є необов'язковими.

Layout: Ковзна шкала може бути орієнтована горизонтально чи вертикально (за замовчуванням – горизонтально). Для горизонтальних шкал значення збільшується зліва направо, для вертикальних – знизу нагору.

Value: Значення – рядок, узятий у лапки, і який утримує поточне (ціле) значення ковзної шкали (за замовчуванням – *Min_value*).

Приклад:

```
:slider {key = "ms"; min_value = 8; max_value = 60; layout = vertical;
        action = "0";height = 18; fixed_height = true; }
```

TOGGLE

Перемикач – Toggle: Оперує двійковими величинами (“0” чи “1”). Його зображено невеликим прямокутником із необов'язковою міткою (*label*) праворуч. Мітка *X* з’являється чи зникає в прямокутнику за вказівкою перемикача. Перемикач дозволяє користувачу бачити й змінювати його стан “увімкнено_вимкнено”.

Атрибути

Label: Текст, виведений ліворуч від кнопки перемикача. Значення – рядок, узятий у лапки.

Value: Визначає початковий стан перемикача. Значення – рядок, узятий у лапки, (за замовчуванням – “0”). Якщо значення дорівнює “0”, кнопка перемикача порожня (не позначена). Якщо значення дорівнює “1”, кнопка виводиться з міткою *X*.

Приклад:

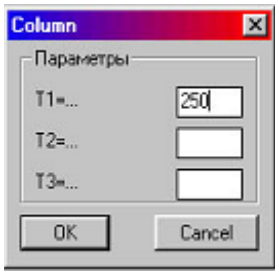
```
:toggle{ key = "VBB";}
```

3.2 Попередньо визначені активні групи полів

Поля можна групувати в ряди чи стовпці. Згруповані ряди й стовпці під час формування загального діалогового вікна розглядають як єдине поле, вони можуть розміщуватись у рамці та мати необов'язкову мітку (група без рамки не може мати мітку). Завдяки такому групуванню зручно розкладати поля у діалоговому вікні.

Ряд чи стовпець можна визначити для зовнішнього використання як елемент діалогового вікна. Групи називають *об'єднаннями*, оскільки вони можуть містити інші поля. Не можна змінювати атрибути об'єднання, якщо воно використовується як посилання в діалоговому вікні. У файлі *base.dcl* визначено кілька стандартних об'єднань.

COLUMN



Стовпець – Column: Стовпець чи будь-який вид поля (крім кнопки вибору), включаючи ряди й інші стовпці. Усі елементи стовпця розташовуються вертикально в порядку розташування в *DCL*-файлі.

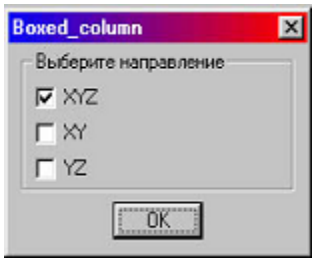
Атрибути

Стовпці без рамки не має додаткових атрибутів, крім стандартних атрибутів компонування.

Приклад:

```
:column{
  …
}
```

BOXED_COLUMN



Стовпець у рамці – Boxed_column: Стовпець, що має намальовану навколо нього рамку. Діалогове вікно розташоване, як і стовпець, у рамці. Якщо стовпець має у рамці мітку, вона може з'явитися вище від рамки чи бути вбудованою в неї. Якщо мітки немає чи є вона пробілом (” “) або порожньою (“”), зображується тільки рамка.

Атрибути

Label: Значення – рядок, узятий у лапки (за замовчуванням – “ ”). Атрибут *label* виводиться як поле у верхньому лівому куті стовпця, розміщеного в рамці.

Приклад:

:boxed_column{
…
}

ROW

Ряд – Row: Аналогічний стовпцеві, але його поля розташовані горизонтально в порядку розташування в *DCL*-файлі.

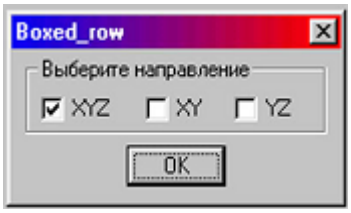
Атрибути

Ряд без рамки не має додаткових атрибутів, крім стандартних атрибутів компонування.

Приклад:

:row{
…
}

BOXED_ROW



Ряд у рамці – Boxed_row: Ряд, що має намальовану навколо нього рамку. Якщо в рядах рамки є мітка, вона може з'явитися вище від рамки чи бути вбудованою в неї. Якщо мітки немає або вона є пробілом (“ ”) чи порожнью (“”), зображується тільки рамка.

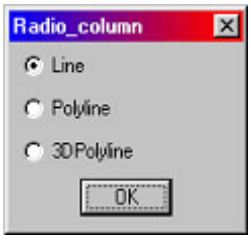
Атрибути

Label: Значення – рядок, узятий у лапки (за замовчуванням – “ ”). Атрибут *label* виводиться як поле у верхньому лівому куті ряду, розміщеного у рамці.

Приклад:

:boxed_row{
…
}

RADIO_COLUMN



Стовпець вибору – Radio_column: Стовпець, що містить поля кнопок вибору. За один раз можна вибрати тільки одну з кнопок. Це фіксований набір взаємовиключних альтернатив. На відміну від звичайних стовпців, стовпці вибору можуть виконувати дії.

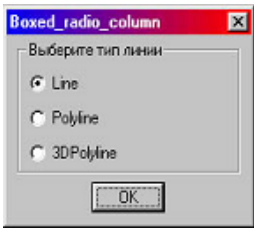
Атрибути

Value: Значення – рядок, узятий у лапки, що містить значення ключа (*key*) обраної поточної кнопки вибору.

Приклад:

```
:radio_column {
    :radio_button{ key="Lin";label = "Line";}
    :radio_button{ key="Plin";label = "Polyline";}
    :radio_button{ key="3DPlin";label = "3DPolyline";}
}
```

BOXED_RADIO_COLUMN



Стовпець вибору в рамці – Boxed_radio_column: Стовпець вибору, що має намальовану навколо нього рамку. Мітка трактується аналогічно мітці стовпця в рамці.

Атрибути

Label: Значення – рядок, узятий у лапки (за замовчуванням – “ ”). Атрибут *label* виводиться як поле у верхньому лівому куті стовпця, розміщеного в рамці.

Value: Значення – рядок, узятий у лапки, що містить значення ключа (*key*) обраної поточної кнопки вибору.

Приклад:

```
:boxed_radio_column {
    :radio_button{ key="Lin";label = "Line";}
    :radio_button{ key="Plin";label = "Polyline";}
    :radio_button{ key="3DPlin";label = "3DPolyline";}
}
```

RADIO_ROW

Ряд вибору – Radio_row: Аналогічний ряду вибору і містить поля кнопок вибору, але за один раз можна вибрати тільки одну кнопку. Ряд вибору може виконувати дії.

Атрибути

Value: Значення – рядок, узятий у лапки, що містить значення ключа (*key*) обраної поточної кнопки вибору.

Приклад:

```
:radio_row {
    :radio_button{ key="Lin";label = "Line";}
    :radio_button{ key="Plin";label = "Polyline";}
    :radio_button{ key="3DPlin";label = "3DPolyline";}
}
```

BOXED_RADIO_ROW

Ряд вибору в рамці – Boxed_radio_row: Ряд вибору, розміщений у рамці. Мітка трактується так само, як і мітка стовпця в рамці.

Атрибути

Label: Значення – рядок, узятий у лапки (за замовчуванням – “ ”). Атрибут *label* виводиться як поле у верхньому лівому куті ряду, розміщеного в рамці.

Value: Значення – рядок, взятий у лапки, що містить значення ключа (*key*) обраної поточної кнопки вибору.

Приклад:

```
:boxed_radio_row {
    :radio_button{ key="Lin";label = "Line";}
    :radio_button{ key="Plin";label = "Polyline";}
```

```
    :radio_button{ key="3DPlin";label = "3DPolyline";}
}
```

3.3 Декоративні й інформаційні поля

IMAGE

Зображення – Image: Прямокутник, усередині якого відображається векторне зображення.

Атрибути

Color: Колір фону малюнка, визначений як номер кольору AutoCAD чи одне із символьних імен (за замовчуванням – 7):

dialog_line – поточний колір діалогового вікна;
dialog_forground – поточний колір символів діалогового вікна (для тексту);
dialog_background – поточний колір фону діалогового вікна;
graphic_background – поточний колір фону графічного екрана AutoCAD (як правило, аналог 0);
black – колір AutoCAD (чорний) на чорному фоні зображується як білий;
red – колір AutoCAD 1 (червоний);
yellow – колір AutoCAD 2 (жовтий);
green – колір AutoCAD 3 (зелений);
cyan – колір AutoCAD 4 (блакитний);
blue – колір AutoCAD 5 (синій);
magenta – колір AutoCAD 6 (фіолетовий);
white – колір AutoCAD 7 (білий);
graphic_forground на білому фоні зображується як чорний.

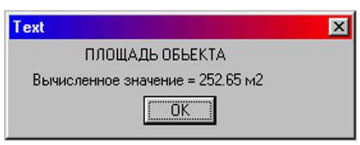
Aspect_ratio: Відношення ширини зображення до його висоти (ширина, ділена на висоту). Якщо значення дорівнює 0 (0.0), то поле зміщується під розміри зображення. Можливе значення з плаваючою точкою (значення за замовчуванням немає).

Полю зображення слід присвоїти точні значення атрибутів *width* і *height* чи один із цих атрибутів і значення *aspect_ratio*.

Приклад:

```
:image { key = "comp1"; color = 7; width =  30; aspect_ratio = 0.7;}
```

TEXT



Напис – Text: Текстовий рядок, який використовують для виведення заголовка поля та діалогового вікна чи з інформаційною метою.

Атрибути

Label: Відображуваний на екрані текст. Значення – рядок, узятий у лапки, (значення за замовчуванням немає). Під час компонування поля напису його ширина більшою від значення атрибута *width*, визначеного в *DCL*-файлі, чи ширини, обумовленої атрибутом *label*, якщо його задано. Якщо жодного з них не задано, видається повідомлення про помилку.

Value: Аналогічно *label* визначає рядок, виведений у поле напису, але не впливає на компонування полів. Якщо повідомлення є незмінним, варто визначити атрибут *label* і не визначати *width* і *value*. У противному разі визначається атрибут *value* й атрибуту *width* присвоюється досить велике значення. Після виведення діалогового вікна на екран не можна змінити розмір його полів; якщо під час виведення функції *set_tile* напису присвоєно більш довге значення, ніж ширина вікна, напис буде усічено.

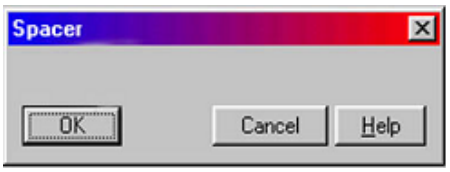
Is_bold: Можливі значення: *true* чи *false* (за замовчуванням – *false*). Якщо *true*, напис виводиться напівжирним шрифтом.

Приклад:

:text {label=" New/Work";} :text {label="" ;} :text {label="Old";}

SPACER

Розділювач – Spacer: Порожнє поле. Використовують тільки з метою компонування, він впливає на розмір і розташування суміжних полів.



Атрибути

Немає.

Приклад:

spacer;

або

:spacer{width=20;};

3.4 Визначені атрибути

Атрибути полів визначають їх розміщення та функціональність і схожі на змінні мови програмування. Атрибут складається з імені та значення. Значення атрибутів можуть бути:

- цілими – це числова величина (ціла чи дійсна), що являє собою розмір в одиницях чи ширини висоти символів;
- дійсними – це числова величина, у якій наявність незначного нуля в цілій частині є обов'язковою;
- рядковими – це рядок тексту, взятий у лапки (“ ”). Якщо рядок містить лапки, перед ним слід поставити похилу риску: \”. Рядок може містити інші *ESC*-послідовності:
  \” – лапки;
  \\ – зворотну похилу риску;
  \n – новий рядок;

\t – горизонтальну табуляцію;

- зарезервованими словами – це ідентифікатор, що складається з текстових символів і починається з літери (наприклад, *true* чи *false* – зарезервовані слова, необхідні в багатьох атрибутах). Зарезервоване слово залежить від регістра: *True* не дорівнює *true*. Імена атрибутів також залежать від регістра. Додатки завжди одержують атрибут у вигляді рядка, тому, якщо як додаток використовують числові величини, вони мають бути перетворені з рядкового вигляду.

Деякі атрибути єдині для всіх полів. Визначення атрибутів є необов'язковим; більшість атрибутів мають значення за замовчуванням і використовуються у разі невизначеного атрибута. Інші атрибути спеціально призначені для визначеного типу полів (наприклад, колір фону зображення).

Можна визначити власні атрибути з іменами, що не конфліктують з жодним із стандартних імен атрибутів. Ім'я атрибута так само, як і ключове слово, може містити літери, цифри чи символ підкреслення (_). Першим символом має бути літера.

Значення визначених користувачем атрибутів мають відповідати типам атрибутів полів. Атрибути, їх поля та призначення наведено в табл. 6.

Таблиця 6

Ім'я атрибута	**Поля**	**Призначення (якщо визначено чи *true*)**
1	2	3
Action	Усі активні поля	Активний вираз AutoLISP
Aignment	Усі	Горизонтальне чи вертикальне положення в групі
Allow_accept	Текстове поле, кнопка зображення та поле списку	Активізує кнопку *is_default*, коли це поле обране
Aspect_ratio	Зображення, кнопка зображення	Коефіцієнт корекції зображення
Big_increment	Ковзна шкала	Найбільший крок переміщення
Children_alignment	Ряд, стовпець, ряд вибору, ряд у рамці, стовпець у рамці, ряд вибору в рамці та стовпець вибору в рамці	Вирівнювання елемента, що належить до групи
Children_fixed_heigth	Ряд, стовпець, ряд вибору, ряд у рамці, стовпець у рамці, ряд вибору в рамці та стовпець вибору в рамці	Висота елемента, що належить до групи (не збільшується при компонуванні)
Children_fixed_width	Ряд, стовпець, ряд вибору, ряд у рамці, стовпець у рамці, ряд вибору в рамці та стовпець вибору в рамці	Ширина елемента, що належить до групи (не збільшується при компонуванні)
Color	Зображення, кнопка зображення	Колір фону зображення
Edit_limit	Текстове поле	Максимальна кількість символів, що може ввести користувач

1	2	3
Edit_width	Текстове поле, що розкривається, список	Ширина частини, що редагується, поля
Fixed_heigth	Усі	Висота не збільшується при компонуванні
Fixed_width	Усі	Ширина не збільшується при компонуванні
Height	Усі	Висота поля
Initial_focus	Елемент діалогу	Ключ поля з початковим підсвічуванням
is_bold	Напис	Виводиться як напівжирний текст
is_cancel	Кнопка	Кнопка активізується при натисканні комбінації клавіш <Ctrl+C>
is_default	Кнопка	Кнопка активізується при натисканні клавіші <ENTER>
is_enabled	Усі активні поля	Поле завжди доступне
is_tab_stop	Усі активні поля	Поле може стати активним при натисканні клавіші табуляції
Key	Усі активні поля	Ім’я (ключ), використовуване додатком
Label	Ряд у рамці, стовпець у рамці, ряд вибору в рамці, стовпець вибору в рамці, кнопка, примітив діалогу, текстове поле, поле списку, що розкривається, список, кнопка вибору, напис і перемикач	Відображувана мітка поля
Layout	Ковзна шкала	Ковзна шкала (вертикальна чи горизонтальна)
List	Поле списку, що розкривається список	Початкові значення, відображувані в списку
Max_ value	Ковзна шкала	Максимальне значення ковзної шкали
Min_ value	Ковзна шкала	Мінімальне значення ковзної шкали
Mnemonic	Усі активні поля	Мнемонічний символ для поля
Multiple_select	Поле списку	Поле списку, яке дозволяє вибрати кілька пунктів
Small_increment	Ковзна шкала	Малий крок переміщення
Tabs	Поле списку, що розкривається, список	Табуляція при відображенні списку
Value	Текст, активні поля (за винятком кнопок)	Початкове значення поля
Width	Усі	Ширина поля

3.5 Структура *DCL*-файлу

DCL-файли, крім діалогових вікон, можуть визначати прототипи об'єднання, які можуть містити визначення з інших *DCL*-файлів. *DCL*-файл може складатися з трьох частин, котрі розташовуються у довільному порядку (деяких частин може і не бути):

– посилання на інші *DCL*-файли. Містять директиви для виклику;
– визначення прототипів полів, об'єднання у стовпці та ряди, на них можна посилатися;
– визначення діалогових вікон.

У комплект постачання AutoCAD входять файли *base.dcl* і *acad.dcl*. Вони можуть бути використані як приклад під час створення *DCL*-файлів.

Файл *base.dcl* містить визначення базових, попередньо визначених полів і типів полів. У ньому також містяться визначення прототипів загального використання. Попередньо визначені поля не можуть бути перевизначені. Цей файл не слід модифікувати! Помилки у файлі *base.dcl* будуть переривати появу як стандартних діалогових вікон AutoCAD, так і діалогових вікон додатків користувача.

Файл *acad.dcl* містить визначення всіх стандартних діалогових вікон, використовуваних стандартною версією AutoCAD. Можна відредагувати цей файл, якщо необхідно змінити появу стандартних діалогових вікон.

3.5.1. Синтаксис мови DCL

Мову DCL використовують для визначення нових і об'єднання існуючих полів у діалогові вікна. Нові вікна створюються визначеннями полів. Якщо визначення з'являється поза діалоговим вікном, воно є прототипом (визначенням поля) чи об'єднанням (елементами, що належать до групи), які можна використовувати в діалогових вікнах для посилань. Кожне посилання на визначення успадковує атрибути вихідного поля. Якщо посилаються на прототип, можна змінювати значення атрибутів чи додавати нові атрибути; у разі посилань на об'єднання атрибути не можуть бути змінені чи додані.

Якщо використовують кілька екземплярів полів із декількома загальними атрибутами, простіше визначити прототип, що містить загальні атрибути. Тоді в кожному посиланні на прототип можна змінювати чи додавати нові атрибути. У такому разі не потрібно перелічувати список усіх загальних атрибутів при кожному посиланні на поле. Оскільки атрибути успадковуються саме таким чином, то під час створення діалогового вікна простіше створити посилання на поля (особливо посилання на попередньо визначені поля), ніж створювати нові поля.

Коментарі. Коментарям у *DCL*-файлах передують дві зворотні похилі риски – "//". Усе, що з'являється між "//" і кінцем рядка, ігнорується. *DCL* також дозволяє використовувати коментарі, прийняті в мові C++. Вони мають форму

“/*”, текст коментаря – “*/”. Початкові “/*” і кінцеві “*/” символи можуть знаходитися на різних рядках.

Кнопки виходу з діалогового вікна. Файл *base.dcl* забезпечує кілька об'єднань стандартних кнопок для виходу з діалогового вікна чи його видалення. Ці стандартні об’єднання варто використовувати для збереження одноманітності вигляду діалогових вікон серед різних додатків. Крім того, у версії файлу *base.dcl* ці кнопки визначені для кожної платформи з урахуванням її особливостей, так, що використання цих визначень гарантує сумісність із кожною платформою.

Ok_only. Аналогічна кнопці *OK* у вікні попереджень. Ключ *key* кнопки *OK* має значення “accept”.

Ok_cancel. Команда, яка викликає *OK* і *Cancel*, що є стандартною комбінацією для діалогових вікон. Ключ *key* кнопки скасування має значення “cancel”.

Ok_cancel_help. Групу *ok_cancel* скомбіновано зі стандартною кнопкою *Help*. Ключ *key* кнопки допомоги має значення “help”.

3.5.2. Керування діалоговими вікнами

Опис діалогового вікна мовою DCL є статичною картиною форми вікна. Усі дії під час діалогу та після закриття вікна визначає керуюча програма виклику діалогового вікна й обробки результатів діалогу. У посібнику розглянуто керування діалоговими вікнами тільки через AutoLISP. Спочатку описано застосовані для цього функції AutoLISP, потім розглянуто особливості їх використання в керуючих програмах. Такі керуючі програми можуть містити не тільки рядки керування діалогом, але й виконувати визначені прикладні задачі користувача, однак більш доцільно функції керування діалоговими вікнами оформляти у вигляді окремих підпрограм, викликаних із прикладної задачі. У цьому разі полегшується режим налагодження діалогу та з'являється можливість використання функцій в інших прикладних програмах.

3.6 Функції AutoLISP для діалогових вікон

3.6.1. Відкриття й закриття *DCL*-файлів

(load-dialog імені-файлу)

Завантажує зазначений *DCL*-файл. Повертає ціле значення (*dcl_id*), яке використовується в наступних викликах функцій *new_dialog* і *unload_dialog*. В ім’я файлу не слід установлювати розширення.

(unload-dialog dcl_id)

Вивантажує зазначений *DCL*-файл. Завжди повертає *nil*.

3.6.2. Відкриття й закриття діалогових вікон

(new_dialog ім’я-вікна dcl_id [[дія] точка])

Починає керування діалоговим вікном, виводить його на екран і може визначити дію за замовчуванням. Аргумент імені-вікна є рядком і визначає діалогове вікно, а аргумент *dcl_id* визначає *DCL*-файл (це значення визначається викликом функції *load_dialog*). Додаток має викликати функцію *new_dialog* перед викликом *start_dialog*. Усі установки, такі, як присвоєння полям значень, створення зображень чи списків для полів списку та зв'язування дій з полями (через виклики функції *action_tile*), мають відбуватися після виклику функції *new_dialog* і перед викликом *start_dialog*. Дія за замовчуванням виконується після указання користувачем активного поля, що не має присвоєного функцією *action_tile* чи визначеного в *DCL*-файлі, функції виклику з поверненням. У разі успішного завершення функції (*new_dialog*) повертає *t*, у противному разі – *nil*.

Аргумент дії, який має бути заданий, якщо визначено аргумент “точка”, є рядком, що містить вираз AutoLISP для використання як дії за замовчуванням. Якщо аргумент дії визначати не потрібно, варто присвоїти йому порожній рядок (“”). Якщо наявний аргумент “точка”, то він є списком (*X Y*) точки розташування діалогового вікна на екрані. Точка зазвичай визначає положення діалогового вікна при його першій ініціалізації. Координати положення вікна слід задавати після виклику функції *done_dialog* для повторного відкриття вікна. Це дозволяє відкривати вікна заново, в тому місці, де їх було закрито. Якщо точка визначена як (-1-1), діалогове вікно відкриється в центрі графічного екрана AutoCAD.

Завжди потрібно перевіряти повернуте функцією *new_dialog* значення. Виклик функції *start_dialog*, якщо функція *new_dialog* повернула значення помилки, може призвести до непередбачених наслідків.

(start_dialog)

Починає діалог у діалоговому вікні. Діалогове вікно має бути спершу ініціалізоване попереднім викликом функції *new_dialog*. Воно залишається активним, поки вираз чи діюча функція виклику з поверненням не викликають функцію *done_dialog*; зазвичай виклик *done_dialog* пов’язаний з полем, що має ключ “accept” (кнопка *OK* чи).

Функція *start_dialog* не має аргументів. Вона повертає аргумент “стан”, переданий у функцію *done_dialog*. Значенням за замовчуванням є “1”, якщо користувач указав кнопку ; “0” – якщо користувач указав кнопку *Cancel*; “-1”, якщо вікна були закриті викликом функції *term_dialog*. Але якщо функція *done_dialog* установила стан у значення більше 1, функція *start_dialog* поверне значення, зміст якого визначається додатком.

(done_dialog [стан])

Завершує діалог у діалоговому вікні та забирає його з екрана. Функція має викликатися з виразу чи дії, функції виклику з поверненням. Ця функція також повертає поточне положення (*X Y*) діалогового вікна. Якщо функція з поверненням визначена для клавіш із значенням ключа “accept” чи “done”,

функція виклику з поверненням обов'язково має викликати функцію *done_dialog*. Якщо цього не зробити, користувач може не вийти з діалогового вікна. Якщо з цими клавішами не позв'язана функція виклику з поверненням і використовуються стандартні клавіші виходу, AutoCAD обробляє їх автоматично і набуває значення 1 (або визначеного додатком значення), інакше функція *start_dialog* поверне значення 0, що аналогічно скасуванню вікна.

Аргумент "стану" не є обов'язковим. Якщо він визначений, то має бути позитивним цілим, яке функція *start_dialog* повертає як 1 для *OK* і як 0 – для *Cancel*. Функція *done_dialog* повертає список координат точки (*X Y*), яка визначає положення діалогового вікна, що залишив користувач. Цей список можна використовувати в наступному виклику функції *new_dialog* для повторного відкриття діалогового вікна в обраному користувачем місці.

(term_dialog)

Завершує діалог у всіх активних у цей момент діалогових вікнах, як у випадку переривання користувачем. Якщо при відкритих *DCL*-файлах виконання додатка переривається, AutoCAD автоматично викликає функцію *term_dialog*. Цю функцію використовують переважно для закриття вкладених діалогових вікон. Функція *term_dialog* завжди повертає *nil*.

3.6.3. Ініціалізація вираження дії та функцій виклику з поверненням

(action_tile ключ вираження_дії)

Присвоює дію, що буде виконуватися після вибору користувачем визначеного поля. Аргумент "ключ" є ім'ям поля, яке буде викликати дію. Дія, присвоєна функцією *action_tile*, заміщує дію за замовчуванням діалогового вікна чи атрибут поля *action*, якщо вони визначені в *DCL*-файлі.

Аргументи "ключ" і "вираження_дії" є рядками. Аргумент "вираження_дії" обчислюється після вибору поля. "Вираження_дії" може звертатися до поточного значення поля (атрибут поля *value*), визначеного як *$value*; до його імені, визначеного як *$key*; до даних, установлених функцією *client_data_tile*, визначених як *$data*; до коду причини виклику, визначеного як *$x* і *$y*.

3.6.4. Обробка – полів і атрибутів

(mode_tile ключ режим)

Установлює режим для даного поля. Аргумент "ключ" визначає поле. Аргумент "режим" є цілим числом, його значення показано в табл. 7.

Таблиця 7

Значення	Результат аргументу
0	Поле ввімкнено (доступне)
1	Поле вимкнено (недоступне)
2	Вибрати поле
3	Підсвітити вміст текстового поля
4	Увімкнути/Вимкнути підсвічування зображення

Для аргументу “режим” можна визначити тільки одне ціле значення. Аргумент “ключ” є рядком.

(get_attr ключ атрибут)

Запитує значення *DCL* зазначеного атрибута. Аргумент “ключ” визначає поле, аргумент “атрибут” – ім’я атрибута в *DCL*-описі поля. Значення повертається як установлене в описі поля; воно не відображає зміни стану поля, що можуть відбутися після введення користувачем чи виклику функції *set_tile*.

Функція *get_attr* повертає значення атрибута як рядок. Аргументи “ключ” і “атрибут” є рядками.

(get_tile ключ)

Запитує значення зазначеного поля. Аргумент “ключ” визначає поле. Функція *get_tile* повертає значення поля як рядок. Аргумент “ключ” є рядком.

(set_tile ключ значення)

Установлює “значення” для зазначеного поля. Аргумент “ключ” визначає поле, аргумент “значення” – присвоює значення. Усі аргументи є рядками.

3.6.5. Задання полів списків і списків, що розкриваються

(start_list ключ [операція] індекс])

Запускає обробку зазначеного поля чи списку, що розкривається, визначеним аргументом “ключ”. Аргумент операції є цілим числом із такими можливими значеннями (табл. 8).

Таблиця 8

Операція	Результат
1	Змінити обраний вміст списку
2	Додати новий пункт у список
3	Замінити список на новий (за замовчуванням)

Аргумент “індекс” ігнорується, якщо функція *start_dialog* не викликається з кодом зміни 1, тоді індекс визначає пункт списку для зміни наступним викликом функції *add_list*. Значення “індекс” починається з нуля. Аргументи “операція” й “індекс” не обов’язкові. Якщо не визначений аргумент “операція”, він за замовчуванням встановлюється в 3. Якщо не визначені “операція” та “індекс”, “індекс” за замовчуванням встановлюється в 0. Аргумент “ключ” є рядком.

(add_list елемент)

Додає заданий рядок у поточний список чи замінює пункт списку на елемент. Аргумент “елемент” є рядком. Він завершує обробку поточного списку.

(end_list)

Вихід з команд задання списків.

3.6.6. Створення зображень

(dimx_tile ключ)

Функції повертають розміри поля в одиницях діалогового вікна, які використовують функції *vector_image*, *fill_image* і *slide_image*, котрі вимагають

задання абсолютних координат. Аргумент “ключ” визначає поле. Координати повертаються як максимально дозволені в поле. Оскільки координати відраховуються від нуля, функції повертають значення на 1 менше, ніж розміри за *X*- чи *Y*-напрямком. Функція *dimx_tile* повертає ширину поля, функція *dimy_tile* – його висоту. Для обох функцій аргумент “ключ” є рядком.

(start_image ключ)

Запускає процес створення зазначеного зображення в поле, визначене аргументом “ключ”, який є рядком.

(vector_image x1 y1 x2 y2 колір)

Креслить вектор на поточному активному зображенні (відкритому функцією *start_image*) із точки (*x1 y1*) у точку (*x2 y2*). Параметр “колір” визначає номер кольору AutoCAD чи один із логічних кольорів (табл. 9).

Таблиця 9

Номер кольору	Мнемоніка ADI	Значення
-2	BGLCOLOR	Колір фону графічного екрана AutoCAD
-15	DBGCOLOR	Колір фону діалогового вікна
-16	DFGCOLOR	Колір діалогового вікна (для тексту)
-18	LINECOLOR	Колір лінії діалогового вікна

Початок (0 0) знаходиться в лівому верхньому куті зображення. Можна одержати координати правого нижнього кута зображення, викликавши функцію розмірів *dimx_tile*.

(fill_image x1 y1 x2 y2 колір)

Малює зафарбований прямокутник на активному зображенні.

Аргументи аналогічні функції *vector_image*.

(slide_image x1 y1 x2 y2 ім'я_слайда)

Відображає слайд AutoCAD на поточному активному зображенні. Слайд може бути не лише окремим, а й бібліотечними (*.slb). У разі виклику з бібліотеки спочатку вказується ім'я бібліотеки, потім у круглих дужках – ім’я слайда. Перший кут слайда, його точка вставки, визначається як (*x1 y1*), другий кут – (*x2 y2*). Початок (0 0) знаходиться в лівому верхньому куті зображення. Завершує створення поточного зображення.

(end_image)

Вихід з команд створення зображень.

3.6.7. Дані, пов’язані з програмним додатком

(client_data_tile ключ дані^-клієнти-дані)

Пов’язує керовані додатком дані з полем, визначеним аргументом “ключ”. Аргумент “ключ” є рядком. Визначені додатком дані задає аргумент, “клієнти-дані” також є рядком. Вираження дії може посилатися на рядок, визначений як *$data*.

3.7 Схема викликів функцій керування

3.7.1. Виклик з файлу *exsample.dcl*

Наведено приклад виклику та керування простим діалоговим вікном:

```
exsample: dialog {label = “Приклад діалогового вікна”;
: text {label = “Діалогове вікно” ; }
ok_only;
 }
```

Указаний текст знаходиться у файлі *exsample.dcl*. Функція AutoLISP, яка керує появою цього вікна, може мати такий вигляд:

```
(defun showalert ( / dcl_id)
   (setq dcl_id (load_dialog “exsample.dcl”))        ;Завантажити DCL-файл
   (if (not (new_dialog “exsample” dcl_id)) ;Ініціалізувати діалог
   (exit))                    ;(вийти, якщо не працює)
   (action_tile)              ;Пов'язати вираження дії
   “accept”                   ;з ключем кнопки OK
   “(done_dialog)”)           ;Закінчити діалог, якщо натиснуто кнопку OK
   (start_dialog)”)                ;Вивести діалогове вікно
   (unload_dialog dcl_id)     ;Вивантажити DCL-файл
)
```

Після виклику функції *start_dialog* діалогове вікно стає активним, поки користувач не підсвітить поле (зазвичай, кнопку), пов’язане з викликом вираження *done_dialog*. Виклик функції *action_tile* установлює зв'язок між полем (у цьому прикладі з кнопкою *OK*, ключ якої має значення “accept”) і вираженням дії. Саме через цей виклик функція *done_dialog* з’являється всередині виклику функції *action_tile* і перед викликом функції *start_dialog*. По суті, всі дії до функції *start_dialog* є коригувальними. Більш складні вікна потребують більшої кількості додаткових операторів між функціями *start_dialog* і *unload_dialog*, але послідовність викликів буде незмінною.

Наведений приклад демонструє послідовність викликів функцій:

1. Завантаження *DCL*-файлу функцією *load_dialog*.
2. Виклик функції *new_dialog* для виведення окремого діалогового вікна на графічний екран AutoCAD. Важливо перевіряти повернене функцією *new_dialog* значення. Виклик функції *start_dialog*, якщо функція *new_dialog* повернула помилку, може призвести до непередбачених результатів.
3. Ініціалізація діалогового вікна та встановлення значень полів, списків і зображень, якщо необхідно. На цьому етапі викликаються функції *set_tile* і *mode_tile* для встановлення стану полів та їхніх значень; *start_list*, *add_list*, *end_list* – для полів списків; *start_image*, *vector_image*, *fill_image*, *slide_image*, *end_image* – для зображень. Аналогічно можна використовувати функцію *action_tile* для зв'язку з діалоговим вікном і його компонентами.

4. Виклик функції *start_dialog* передає керування діалоговому вікну, в яке користувач може вводити дані.
5. Процес уведення даних користувачем (виклики з поверненням). На цьому етапі використовують функції *get_tile, get_attr, set_tile, mode_tile.*
6. Указання користувачем клавіші виходу викликає функцію *done_dialog*, яка, у свою чергу, повертає функцію *start_dialog*. Далі відбувається вивантаження *DCL*-файлу, забезпечуване функцією *unload_dialog.*

Якщо діалогове вікно (функцію *start_dialog* викликано) є активним, то не можна викликати деякі функції AutoLISP, що змінюють екран, який не має змінюватися, поки на ньому зображується діалогове вікно.

Якщо необхідно ввести дані в режимі графічного екрана (наприклад, вибрати точку чи примітив), слід тимчасово закрити діалогове вікно, викликавши функцію *done_dialog*, – графічний екран стане доступним. Після виконання вибору виконується повторний виклик вікна.

Перелік функцій AutoCAD, заборонених до виклику при відкритому діалоговому вікні:

1. Запити і команди AutoCAD: *command, osnap*.
2. Функції введення користувачем: *getint, getreal, getstring, getpoint, getcorner, getdist, getangle, getorient, getword*.
3. Функції керування екраном: *prompt, menucmd, redraw, graphscr, textscr, textpage*.
4. Графічні функції низького рівня: *grclear, grdraw, grread, grtext, grvacs*.
5. Функція набору вибору *ssget*.
6. Функції керування примітивами: *entmod, entmake, entdel, entsel, nentsel, entpd*.

3.7.2. Вираження дій

Для визначення наслідків вибору визначеного поля діалогового вікна варто зв’язати вираження AutoLISP із цим полем за допомогою виклику функції *action_tile*. Усередині дії часто буває необхідно одержати доступ до атрибутів *DCL*-файлу. Це забезпечують функції *get_attr* і *get_tile*: функція *get_attr* видає значення, збережені в *DCL*-файлі, а *get_tile* – поточні значення. Значення, пов’язані з обраним полем, виходять автоматично.

Вираження дії AutoLISP одержує доступ до змінних, котрі подано в табл. 10, й описують обране поле та його стан під час дії. Імена змінних зарезервовані, можна вживати тільки їх значення (вони мають сенс тільки тоді, коли доступні вираженню дії).

Таблиця 10

Змінна	Призначення	Примітка
$key	Атрибут *key* обраного поля	Застосовують до всіх дій
$value	Рядок поточного значення поля, схожий на рядок вікна редагування, “0” чи “1” для перемикача	Застосовують до всіх дій. Якщо поле є полем списку (чи списку, що розкривається) і не містить жодного пункту, змінна *$value*

		буде мати значення *nil*
$data	Дані, встановлені відразу після виклику *new_dialog* через виклик *client_data_tile* і керовані додатком	Застосовують до всіх дій. Не має значення доти, поки додаток не установить його викликом функції *client_data_tile*
$reason	Код причини, що повідомляє, який вибір користувача викликав дію. Використовується в полях *edit_box, list_box, image_button* і *slider*	Визначає причину дії. Її значення визначене для будь-якого виду дії, але його варто перевіряти, якщо дія пов'язана з полями *edit_box, list_box; image_button* і *slider*
$x	Координата *X* вибору *image_button*	Представляє координати *X* і *Y* точки, в якій користувач вибрав поле *image_button*
$y	Координата *Y* вибору *image_button*	Для інших цілей змінна не має значення. Координата *X* знаходиться в інтервалі, який повертає функція *dimx_tile*, координата *Y* – в інтервалі, котрий повертає функцієя *dimy_tile*

Змінна *new_tile* встановлюється в значення атрибута *key* обраного поля, тобто в "*edit_*". Змінну *$key* часто використовують усередині функцій для виконання дій, присвоєних декількома різними полями.

3.7.3. Тимчасове закриття діалогових вікон

У разі необхідності виходу з діалогового вікна для інтерактивного вибору на графічному екрані, варто тимчасово закрити діалогове вікно та після виконання дії на графічному екрані повернутися в діалогове вікно з повним його відновленням. Таку процедуру виконують за допомогою функції *done_dialog* з аргументом *status*.

Приклад: У *LISP*-програмі використовують кнопку вказання точки, що тимчасово закриває діалогове вікно для надання можливості введення точки на графічному екрані. При вказанні цієї кнопки відбувається закриття діалогового вікна з кодом стану 4:

```
(action_tile "pick_pt" "(done_dialog 4)")
```

Після повернення з функції *start_dialog* програма перевіряє код стану, що повертається, і, якщо потрібно, змінює координати точки:

```
(setq what_next (start_dialog))
(cond ((=what_next 4)
(progn
(setqpick_pt (getpoint "Базова точка вставки: "))
(setq x_pt (rtos (car (pick_pt) 2 4 ))
(setq y_pt (rtos (cadr (pick_pt) 2 4 ))
(setq z_pt (rtos (caddr (pick_pt) 2 4 ))
```

))
)

Тіло основної функції розміщене в циклі і функція *start_dialog* доти, доки користувач не натисне кнопку *OK* чи *Cancel*:

```
(defun c:mblock ...
...
(while (< 2 what_next)
...
(setq what_next (start_dialog)
(cond
...
                )
        ))
```

3.7.4. Ініціалізація режимів і значень полів

У разі зміни значення одного поля в полях, пов’язаних з ним, відбуваються зміни. Якщо ініціалізувати діалогове вікно, визначене поле може бути спочатку ввімкнене чи вимкнене і підсвічене чи не підсвічене. Ці операції виконуються за допомогою функції *mode_tile*. Установити значення поля можна за допомогою функції *set_tile*. Під час ініціалізації встановлюються поля списку і створюються зображення.

Приклад: Набір функцій установлює значення за замовчуванням для текстового поля і підсвічує його:

```
(setq name_str “Колінчастий вал”)
(set_tile “lastname” name_str)
(mode_tile “lastname” 2)
```

Функція може використовуватися для підсвічування вмісту текстового поля:

```
(mode_tile “lastname” 3)
```

Перш ніж вимкнути підсвічене поле, необхідно за допомогою функції *mode_tile* підсвітити інше поле.

Приклад “самовимкнення” поля – послідовність “сторінок” діалогових вікон, які перегортають натиканням кнопок “Наступний” чи “Попередній”. Коли вказують кнопку “Наступний” на передостанній “сторінці” після переходу, вона відключається. Те саме відбувається з кнопкою “Попередній” при переході на першу “сторінку”. В обох випадках використана кнопка відключається і підсвічує інше поле.

Припустимо, що перемикач з ім'ям “*group_on*” керує групою “*group*”: коли перемикач вимкнено, поля групи недоступні та не можуть бути змінені. У цьому разі для перемикача варто визначити таку дію:

```
(action_tile “group_on” “(mode_tile \”group\” (- 1 (atoi $value)))”)
```

Обчислення і виклик функції *atoi* у вираженні дії встановлюють значення аргументу *mode* функції *mode_tile*. Оскільки значення перемикача дорівнює 0, коли його вимкнено, і 1 – коли ввімкнено, обчислення змінює значення аргументу *mode* на протилежне і керує доступністю групи.

3.7.5. Установлення полів списків і списків, що розкриваються

При встановленні значень для поля чи списку, що розкривається, необхідно виконати таку послідовність операцій:

(start_list), (add_list) і (end_list)

Список можна змінити після його створення. Функцію *mapcar* слід використовувати для переведення списку з AutoLISP у відображуване поле списку:

```
(start_list “selections” операція)
(mapcar ‘add_list newnames)
(end_list)
```

3.7.6.Обробка значень списків

Під час оброблення списків значення поля *list_box* може містити початкові пробіли, особливо якщо обрано кілька пунктів. Спочатку варто перетворити значення до цілого, використовуючи функцію *atoi* чи *read*.

Приклад: Передбачається, що список “*justone*” дозволяє вибрати тільки один пункт списку. Оскільки функція *atoi* повертає 0, якщо рядок порожній, чи коли в ньому міститься “0”, то спочатку варто перевірити, чи є рядок порожнім.

```
(setq index (get_tile “justone”))
(cond
((/=index “”)
(= 2 (atoi index)) ; це третій пункт
...
            )
        )
```

Значення списку, що розкривається, ніколи не починається із символу пробілу, так що перетворення в цьому випадку виконувати не обов'язково. Списки, що розкриваються, не дозволяють виконувати множинний вибір. Якщо в поле списку можна вибрати одночасно декілька пунктів, програма має не тільки виконати перетворення, але і перебрати безліч значень у рядку.

Приклад: Функція *mk_list* повертає список, що містить тільки обрані користувачем поля з вихідного списку *displist*. Змінна *displist* визначена як глобальна. Передбачається, що функція *mk_list* викликається з поточним значенням *$value* поля списку.

```
(defun mk_list (readlist / count item retlist)
(setq count 1)
(while (setq item (read readlist))
(setq retlist (cons (nth item displist) retlist))
```

```
(while (and (/= " "(substr readlist count 1))
(while (and (/= """ (substr readlist count 1))
(+ " "(substr readlist count 1)))
(setq count (1+ count))
)
(setq readlist (substr readlist count))
                                )
(reverse retlist)
          )
```

3.7.7.Обробка зображень

Послідовність викликів для створення поля зображення та клавіші зображення аналогічні послідовності оброблення списку. Функція *start_image* починає створення зображення, функція *end_image* закінчує його. Однак визначення результату малювання здійснюється використанням декількох різних функцій, а не заданням різних аргументів:

– *vector_image* – креслить вектор (одна пряма лінія) на поточному зображенні;

– *fill_image* – малює зафарбований прямокутник;

– *slide_image* – відображає слайд AutoCAD.

Ці функції вимагають визначення абсолютних координат. Щоб задати їх коректно, варто знати розміри поля чи зображення клавіші зображення. Оскільки розміри полів визначаються під час компонування і забезпечують одержання ширини та висоти поля: *dimx_tile* і *dimy_tile*. Початок поля (0 0) завжди знаходиться у верхньому куті.

Приклад: Потрібно зафарбувати червоним кольором поле зображення "cur_color":

```
(setq width (dimx_tile "cur_color")
    height (dimy_tile "cur_color"))
(start_image "cur_color")
(fill_image "cur_color")
(fill_image 0 0 width height 1) ; 1 – червоний колір AutoCAD
(end_image)
```

Далі намалюємо навколо поля межу червоного кольору, а не зафарбований прямокутник (вектори креслять проти годинникової стрілки):

```
(setq width (dimx_tile "border")
    height (dimy_tile "border"))
(start_image "border")
(vector_image 0 0 0 height 1)
(vector_image 0 height width height 1)
(vector_image width height width 1)
```

```
(vector_image width 0 0 0 1)
(end_image)
```

Зобразимо зафарбоване зображення і накреслимо на ньому вертикальну лінію:

```
(setq width (dimx_tile “striple”)
height (dimy_tile “striple”))
(start_image “striple”)
(fill_image 0 0 width height 3); 3 – зелений колір AutoCAD
(setq x(/width 2.0)
(vector_image x 0 height 4); 4 – фіолетовий колір AutoCAD
(end_image)
```

Відображені за допомогою функції *slide_image* слайди можуть бути окремими слайдами-файлами (*.sld*). Розширення вказувати не потрібно.

Припустимо, що необхідно відобразити слайд *tppview.sld*:

```
(setq x (dimx_tile “view”)
    y (dimy_tile “view”))
(start_image “view”)
(slide_image 0 x y “topview”)
(end_image)
```

Вектори слайда часто малюють білим кольором, що є кольором фону зображення за замовчуванням. Якщо слайд після виведення не видно, треба змінити атрибут *color* на значення *graphic_background*.

3.7.8. Уведення кнопки зображення

Кнопку зображення можна обробляти як звичайну кнопку, тобто при її указанні викликається відповідна дія. Однак є можливість визначення кнопку так, що дія буде залежати від місця вказання в зоні кнопки. Для цього поле функції кнопки має одержати координати вказівки. Координати задаються в діапазоні розміру зображення і повертаються функціями визначення розмірів.

Приклад: Припустимо, що кнопка зображення має дві прямокутні області різного кольору. Потрібно визначити, яку з областей указав користувач. Якщо кнопку поділено навпіл горизонтальною лінією, варто перевіряти тільки один розмір:

```
(action_tile “image_set” “(pick_shade” $key $value $y)”)
...
(defun pick_shade (key val y)
(setq threshold (/dimy_tile key) 2))
(if (yy threshold)
(setq result “Світлий”)
(setq result “Темний”))
)
```

3.7.9. Обробка ковзних шкал

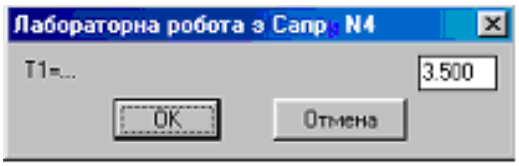
Дію ковзної шкали перевіряє код причини. Як приклад покажемо базову схему функції керування ковзною шкалою. Вона викликається з пов'язаного з нею вираження дії. Поле *slider_info* ця функція використовує для виведення в десятковій формі поточного значення стану шкали і є текстовим полем, що дозволляє користувачу керувати станом ковзної шкали, безпосередньо вводячи значення. Якщо користувач увів значення стану ковзної шкали, то:

```
(action_tile “my_slider” “(slider_action $value $reason)”)
(action_tile “my_info” “(ebox_action $value $reason)”)
...
(defun slider_action (val why)
     (if (or (=why 2) (=why 1))(set_tile “slider_info” val) )
)
(defun ebox_action (val why)
     (if (or (= why 2) (= why 1)) (set_tile “myslider” val) )
)
```

Наведемо приклад розроблення простого діалогового вікна:

Файл dial.dcl

```
dialog_lr : dialog { label = "Лабораторна робота з Сапр N4" ;
  :row {:text { abel="T1=...";}:edit_box{edit_width=5;edit_limit=5;key="tt1";}}
    ok_cancel ;
}
```



Файл dial.lsp

```
(defun c:dialog ()
     (setq rfile (load_dialog "dial.dcl"))
     (if (minusp rfile)(setq rfile (load_dialog "dial.dcl")))
     (new_dialog "dialog_lr" rfile)
     (setq t1 3.5)
     (set_tile "tt1" (rtos t1 2 3))
     (action_tile "tt1" "(progn (setq t1 (atof $value)))")
     (action_tile "accept" "(setq pozdlg (done_dialog 1))")
     (action_tile "cancel" "(done_dialog 0)")
     (setq rslt (start_dialog))
     (print t1)(princ)
)
```

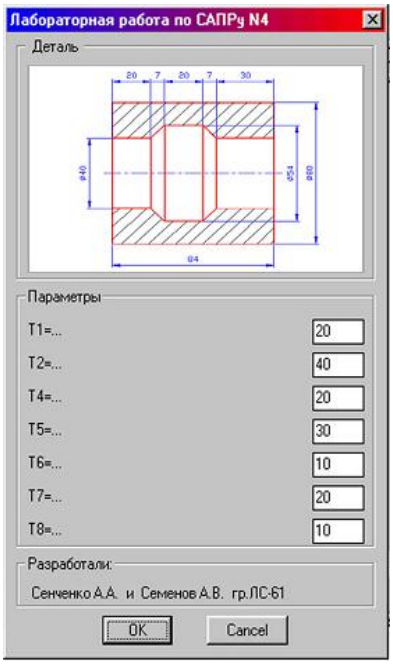
ІНДИВІДУАЛЬНІ ЗАВДАННЯ ДЛЯ СТВОРЕННЯ ДІАЛОГОВИХ ВІКОН

Приклад виконання індивідуального завдання.

Текст програми мовою DCL:

```
dialog_lr : dialog {label="Лабораторна робота з Сапр N4";
        :boxed_row {label = " Деталь " ;
        :image_button {key="detal";
         color=black;aspect_ratio=0.9;width=30;}
         }
   : boxed_row {label = "Параметри" ;
      : column {
            :row {:text {label="T1=...";}
                  :edit_box{edit_width=5;edit_limit=5;key="tt1";}}
            :row {:text {label="T2=...";}
                  :edit_box{edit_width=5;edit_limit=5;key="tt2";}}
            :row {:text {label="T4=...";}
                  :edit_box{edit_width=5;edit_limit=5;key="tt4";}}
            :row {:text {label="T5=...";}
                  :edit_box{edit_width=5;edit_limit=5;key="tt5";}}
            :row {:text { label="T6=...";}
                  :edit_box{edit_width=5;edit_limit=5;key="tt6";}}
            :row {:text { label="T7=...";}
                  :edit_box{edit_width=5;edit_limit=5;key="tt7";}}
            :row {:text { label="T8=...";}
                  :edit_box{edit_width=5;edit_limit=5;key="tt8";}}
                }
      }
    :row {:boxed_column {label = "Розробили:" ;
        :row{:text { label="Сенченко А.А. і Семенов А.В. гр.ЛС-61";}}}}
        ok_cancel ;
}
```

Результатом виконання цієї програми в AutoCAD є діалогове вікно:



Текст програми мовою AutoLISP:

```
(defun c:dialog ()
(setq rfile (load_dialog "D:\\Dialog\\Dialog_lr.dcl"))
(if (minusp rfile)(setq rfile (load_dialog "Dialog_lr.dcl")))
(new_dialog "dialog_lr" rfile)
(setq t1 20.0)(set_tile "tt1" "20")
(setq t2 40.0)(set_tile "tt2" "40")
(setq t4 20.0)(set_tile "tt4" "20")
(setq t5 30.0)(set_tile "tt5" "30")
(setq t6 10.0)(set_tile "tt6" "10")
(setq t7 20.0)(set_tile "tt7" "20")
(setq t8 10.20)(set_tile "tt8" "10")
(setq orx (dimx_tile "detal") aby (dimy_tile "detal"))
   (start_image "detal")
     (slide_image -20 0 (+ 50 orx) (+ 0 aby) "Image1.sld")
   (end_image)
(action_tile "tt1" "(progn (setq t1 (atof $value)))")
(action_tile "tt2" "(progn (setq t2 (atof $value)))")
(action_tile "tt4" "(progn (setq t4 (atof $value)))")
(action_tile "tt5" "(progn (setq t5 (atof $value)))")
(action_tile "tt6" "(progn (setq t6 (atof $value)))")
(action_tile "tt7" "(progn (setq t7 (atof $value)))")
(action_tile "tt8" "(progn (setq t8 (atof $value)))")
(action_tile "accept" "(setq pozdlg (done_dialog 1))")
(action_tile "cancel" "(done_dialog 0)")
(setq rslt (start_dialog))
(princ)
)
```

Завдання 7. Створення вікна з діючою кнопкою Button

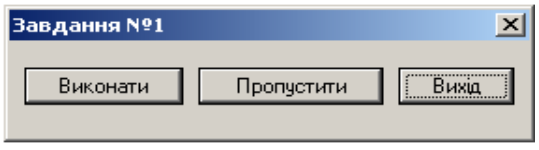
Мета: навчитися, використовуючи функції функціональної мови AutoLISP і мови *DCL*, створювати діючі програми.

Порядок виконання

1. Розробити програму мовою DCL.
2. Розробити програму функціональною мовою AutoLISP.
3. Виконати довільну тестову програму, натиснувши кнопку.
4. Зробити кнопку активною та пасивною.
5. Виконати програму.
6. Вивести на друк результат.

Контрольні запитання

1. Попередньо визначені активні поля.
2. Попередньо визначені активні групи полів.
3. Синтаксис мови DCL.
4. Кнопки виходу з діалогового вікна



Завдання 8. Створення вікна з діючим текстовим полем Edit_box

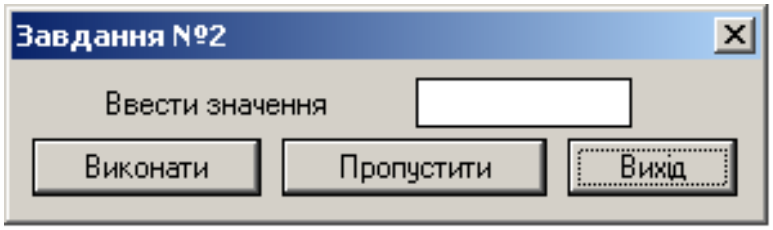
Мета: навчитися, використовуючи функції функціональної мови AutoLISP і DCL, створювати діючі програми.

Порядок виконання

1. Розробити програму мовою DCL.
2. Розробити програму функціональною мовою AutoLISP.
3. Виконати довільну тестову програму, натискнувши кнопку.
4. Зробити діюче текстове поле активним і пасивним.
5. Виконати програму.
6. Вивести на друк результат.

Контрольні запитання

1. Ініціалізація виражень дії та функцій виклику з поверненням.
2. Обробка полів і атрибутів.
3. Задання полів списків і списків, що розкриваються.
4. Дані, пов’язані з програмним додатком.
5. Функції, заборонені під час дії діалогового вікна.



Завдання 9. Створення вікна з діючою кнопкою зображення Image_button

Мета: Навчитися, використовуючи функції функціональної мови AutoLISP і DCL, створювати діючі програми.

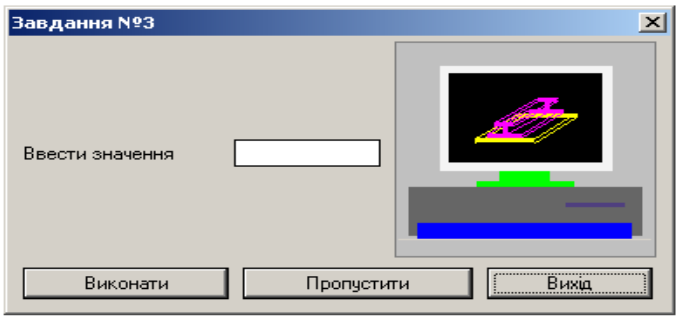
Порядок виконання

1. Розробити програму мовою DCL.
2. Розробити програму функціональною мовою AutoLISP.
3. Виконати довільну тестову програму, натиснувши кнопку.
4. Увести імя файлу та відобразити його зображення в *Image_button*.
5. Виконати програму.
6. Вивести на друк результат.

Контрольні запитання

1. Відкриття і закриття діалогових вікон.

2. Декоративні й інформаційні поля.
3. Керування діалоговими вікнами.
4. Відкриття і закриття DCL-файлів.



Завдання 10. Створення вікна з діючим полем списку List_box

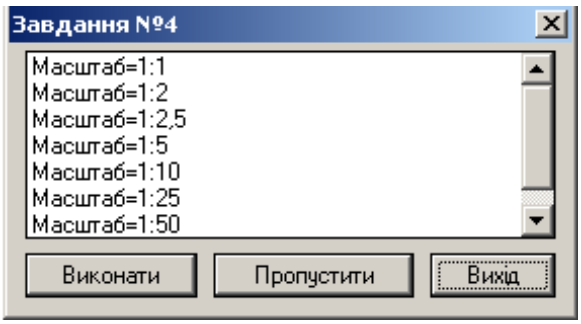
Мета: навчитися, використовуючи функції функціональної мови AutoLISP і мови DCL, створювати діючі програми.

Порядок виконання

1. Розробити програму мовою DCL.
2. Розробити програму функціональною мовою AutoLISP.
3. Вибрати зі списку *List_box* потрібний рядок і вивести його на друк.
4. Виконати програму.
5. Вивести на друк результат.

Контрольні запитання

1. Ініціалізація виражень дії та функцій виклику з поверненням.
2. Задання полів списків і списків, що розкриваються.
3. Створення зображень.
4. Тимчасове закриття діалогових вікон.



Завдання 11. Створення вікна з списком що розкривається, Popup_list

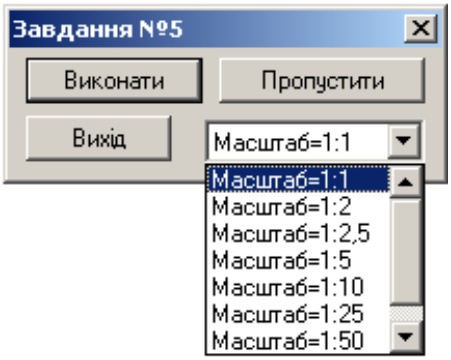
Мета: навчитися використовуючи функції функціональної мови AutoLISP і мови DCL, створювати діючі програми.

Порядок виконання

1. Розробити програму мовою DCL.
2. Розробити програму функціональною мовою AutoLISP.
3. Виконати довільну тестову програму, натиснувши кнопку.
4. Вибрати зі списку *Popup_list* потрібний рядок і вивести його на друк.
5. Виконати програму.
6. Вивести на друк результат.

Контрольні запитання

1. Попередньо визначені активні поля.
2. Синтаксис мови DCL.
3. Кнопки виходу з діалогового вікна.
4. Керування діалоговими вікнами.
5. Відкриття і закриття *DCL*-файлів.



Завдання 12. Створення діалогового вікна з кнопкою вибору Radio_button

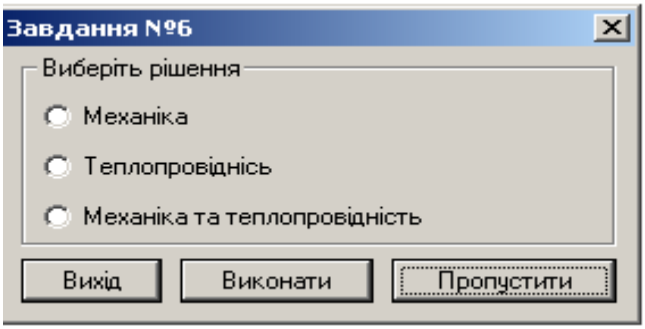
Мета: навчитися, використовуючи функції функціональної мови AutoLISP і мови DCL, створювати діючі програми.

Порядок виконання

1. Розробити програму мовою DCL.
2. Розробити програму функціональною мовою AutoLISP.
3. Виконати довільну тестову програму, натиснувши кнопку.
4. Зробити першу кнопку активною та пасивною, вивести результат на друк.
5. Виконати програму.
6. Вивести на друк результат.

Контрольні запитання

1. Обробка полів і атрибутів.
2. Задання полів списків і списків, що розкриваються.
3. Дані, пов’язані з програмним додатком.
4. Функції, заборонені під час дії діалогового вікна.



Завдання 13. Створення вікна з діючою ковзною шкалою Slider

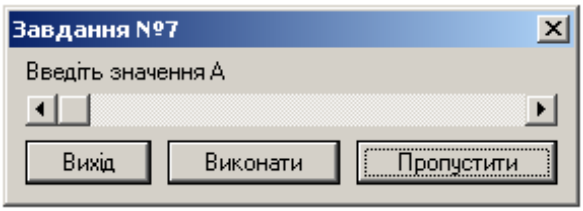
Мета: навчитися, використовуючи функції функціональної мови AutoLISP і мови DCL, створювати діючі програми.

Порядок виконання

1. Розробити програму мовою DCL.
2. Розробити програму функціональною мовою AutoLISP.
3. Виконати довільну тестову програму, натиснувши кнопку.
4. Увести довільне значення змінної *A* та вивести його на друк.
5. Виконати програму.
6. Вивести на друк результат.

Контрольні запитання

1. Попередньо визначені активні поля.
2. Декоративні й інформаційні поля.
3. Синтаксис мови DCL.
4. Кнопки виходу з діалогового вікна.
5. Керування діалоговими вікнами.
6. Відкриття і закриття діалогових вікон.



Завдання 14. Створення вікна з діючим перемикачем Toggle

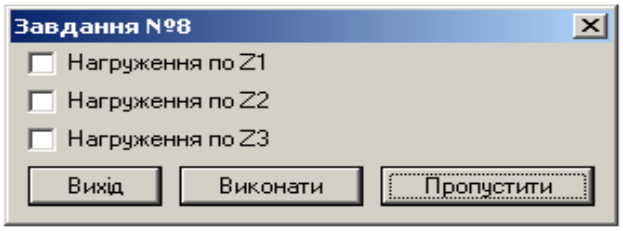
Мета: навчитися, використовуючи функції функціональної мови AutoLISP і мови DCL, створювати діючі програми.

Порядок виконання

1. Розробити програму мовою DCL.
2. Розробити програму функціональною мовою AutoLISP.
3. Виконати довільну тестову програму, натиснувши кнопку.
4. Зробити першу кнопку активною та пасивною, вивести результат на друк.
5. Виконати програму.
6. Вивести на друк результат.

Контрольні запитання

1. Ініціалізація виражень дії та функцій виклику з поверненням.
2. Обробка полів і атрибутів.
3. Завдання полів списків і списків, що розкриваються.
4. Дані, пов’язані з програмним додатком.
5. Функції, заборонені під час дії діалогового вікна.
6. Тимчасове закриття діалогових вікон.



Завдання 15. Створення діалогового вікна для введення формалізованих параметрів

Мета: навчитися, використовуючи функції функціональної мови AutoLISP і мови DCL, створювати діалогіві вікна з формалізованими параметрами.

Порядок виконання

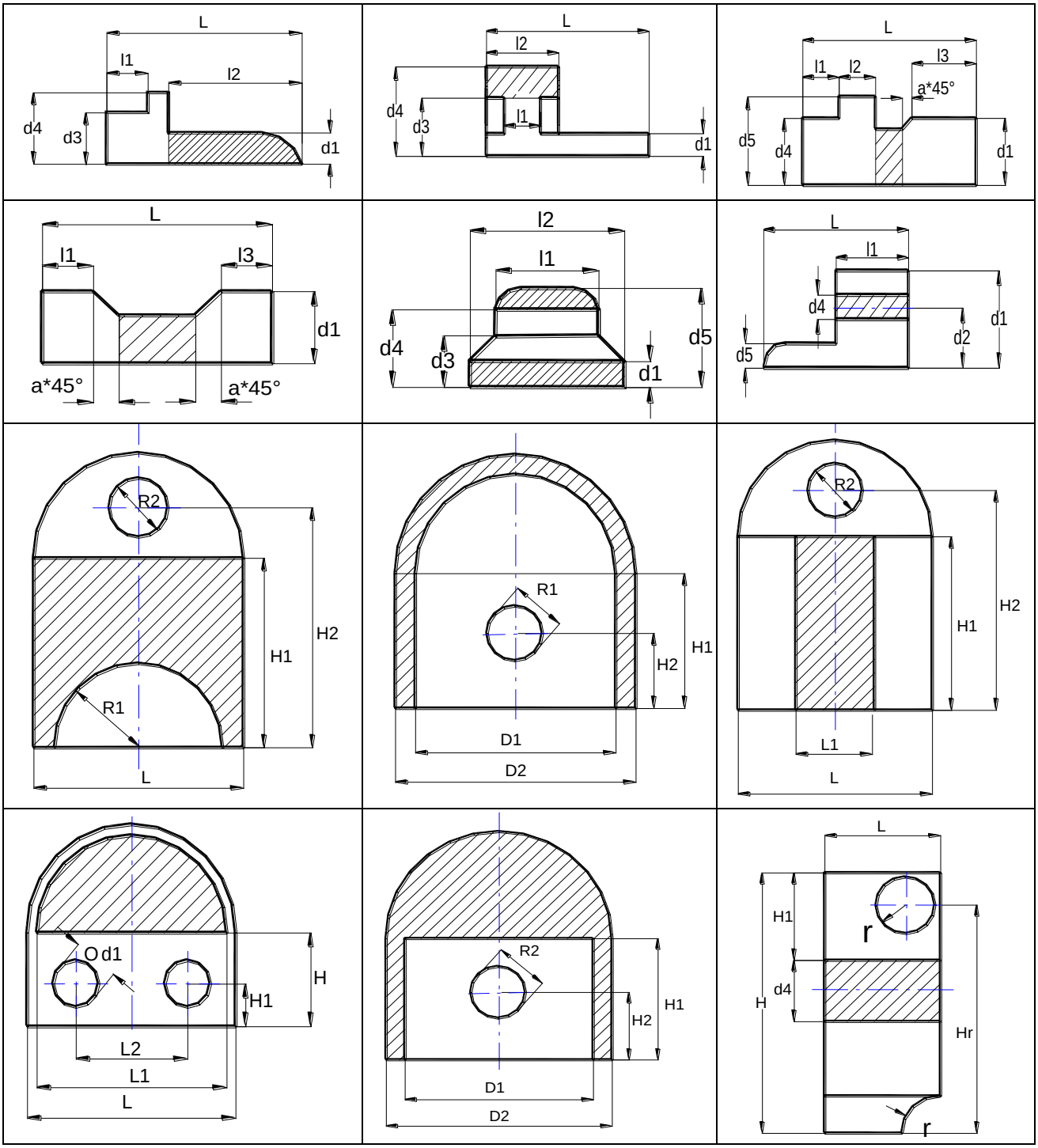
1. Визначити формалізованими параметри для ввдення по завданню 5.
2. Розробити діалогове вікно мовою *DCL*.
3. Розробити програму роботи з діалоговим вікном функціональною мовою AutoLISP.
4. Об’єднати програми діалогового вікна та програму побудови конструкції розробленої для завдання 5.
5. Виконати програму.
6. Вивести на друк результат.

Контрольні запитання

1. Відкриття і закриття діалогових вікон.
2. Синтаксис мови DLC.
3. Кнопки виходу з діалогового вікна.
4. Керування діалоговими вікнами.
5. Відкриття і закриття *DLC*-файлів.

ЗАВДАННЯ ДО ЕКЗАМЕНАЦІЙНИХ БІЛЕТІВ ПО КУРСУ САПР

На функціональній мові AutoLISP розробити програму, для побудови креслення приведеного на Рис. 26.



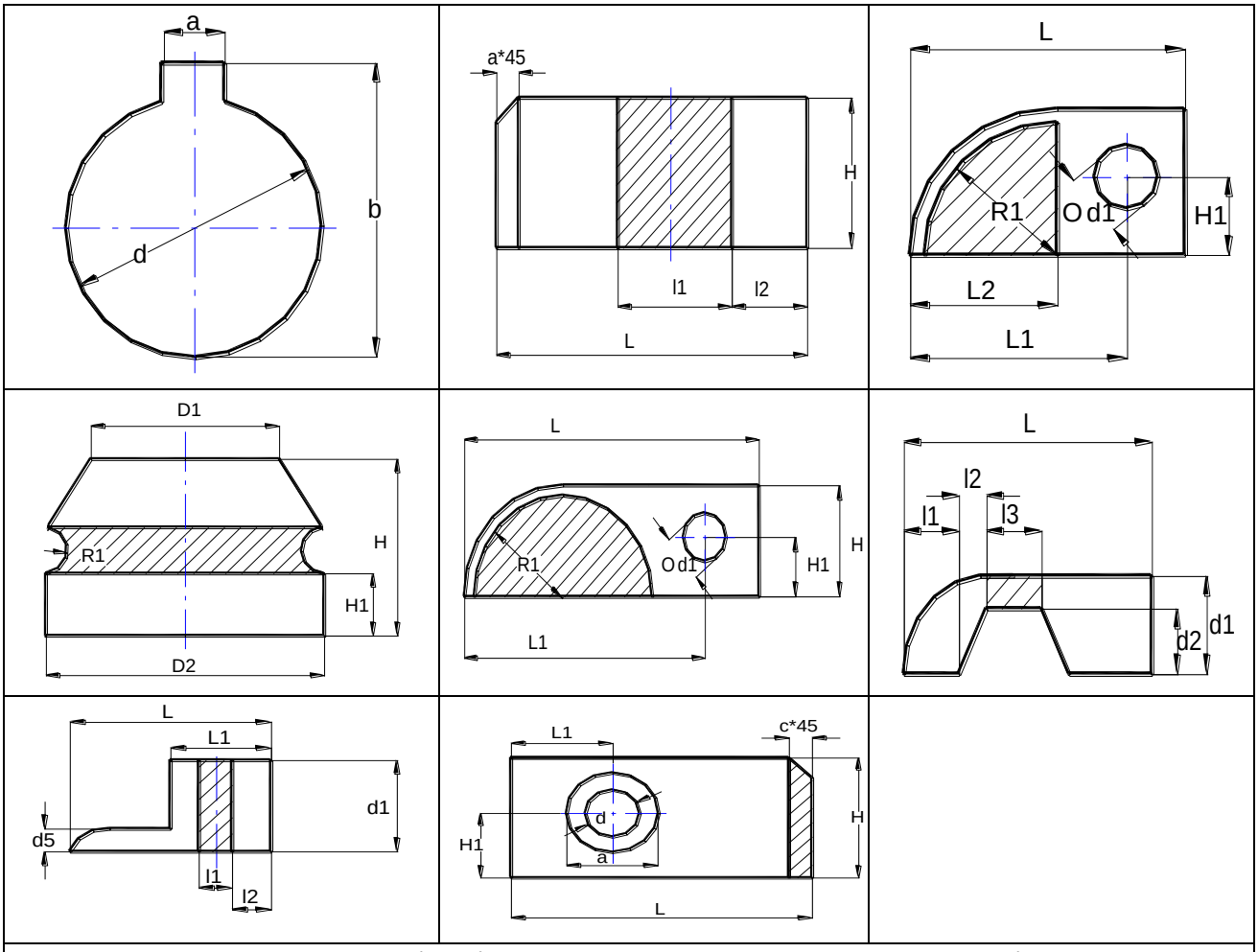
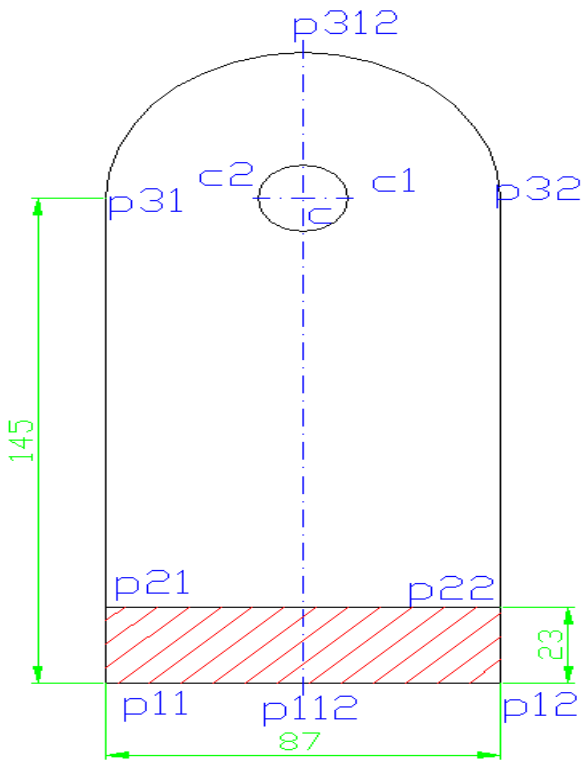


Рис. 26 Завдання для білетів по САПР, програмування на мові **AutoLISP**

Приклад виконання програми на функціональній мові AutoLISP

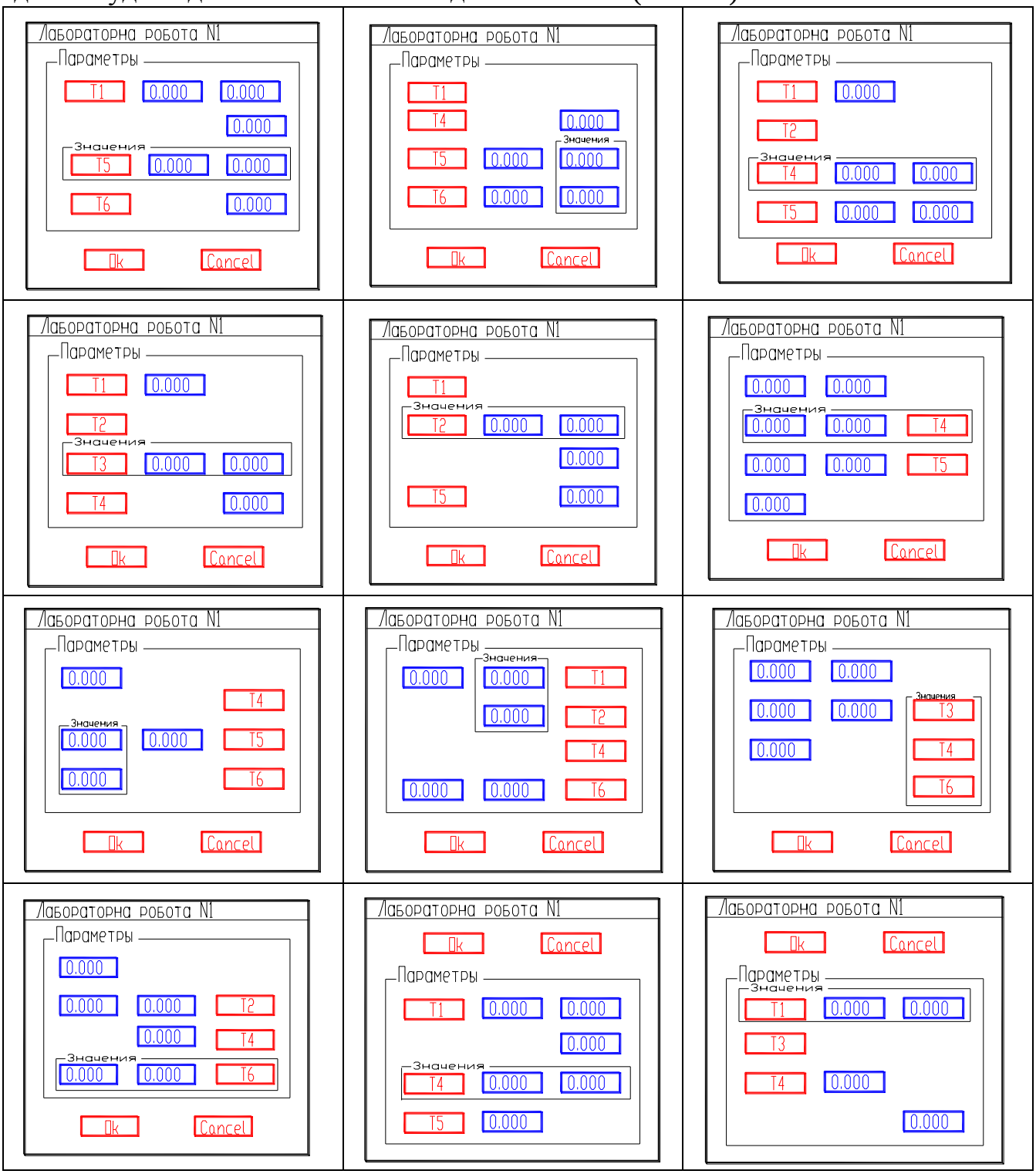
```
(defun c:primer()
; Введення даних
  (setq bp (getpoint "\n Ввести Вр: "))
  (setq P (getdist bp "\n Ввести Р: "))
  (setq k (getdist bp "\n Ввести k: "))
  (setq h (getdist bp "\n Ввести Н: "))
  ;  Визначення координат
   (setq d (* p 0.25)
        p11 bp
        p12 (polar p11 0 p)
        p21 (polar p11 (/ pi 2) k)
        p22 (polar p21 0 p)
        p31 (polar p11 (/ pi 2) h)
        p32 (polar p31 0 p)
        c (polar p31 0 (* p 0.5))
        c1 (polar c 0 (* d 0.5))
        c2 (polar c pi (* d 0.5))
        p112 (polar p11 0 (* p 0.5))
        p312 (polar p112 (/ pi 2) (+ h (* p 0.5)))
   )
```

```
  ; Креслення контура
  (command "_LAYER" "_M" "OSN" "")
  (command "_pline" p11 p12 p22 p21 "_c")(setq en1 (entlast))
  (command "_pline" p21 p31 "_a" p32 "_L" p22 "")
  (command "_pline" c1 "_a" "_ce" c c2 c1 "")
  ; Оси
  (command "_LAYER" "_M" "OSI" "_C" 5 "" "_L" "dashdotx2" "" "")
  (setvar "LTSCALE" 10)
  (command "_LINE" (polar p112 (* pi 1.5) (* (distance p112 p312) 0.02))
                   (polar p112 (* pi 0.5) (* (distance p112 p312) 1.02)) "")
  (command "_line" (polar c2 pi (* (distance c1 c2) 0.05))(polar c2 0(* (distance c1 c2) 1.05)) "")

  ; Штрихування
  (command "_LAYER" "_M" "SHCH" "_C" 1 "" "")
  (command "_HATCH" "_U" 45 5 "_n" en1 "")

  ; Розміри
  (command "_LAYER" "_M" "TEXT" "_C" 3 "" "")
  (setvar "dimtxt" 5)(setvar "dimasz" 5)(setvar "dimtad" 1)
  (setvar "dimtih" 0)(setvar "dimgap" 1.5)
  (command "_dim")
  (command "hor" p11 p12 (polar p11 (* pi 1.5) 15) (rtos p 2 0))
  (command "ver" p11 c (polar p11 pi 15) (rtos h 2 0))
  (command "ver" p12 p22 (polar p12 0 15) (rtos k 2 0))
  (command "_al" c1 c2 (polar c (* pi 0.5) d) (strcat"%%c" (rtos d 2 0)))
  (command "_exit")
)
```



Результат роботи програми

З допомогою мови керування діалогом – DCL розробити фрагмент програми, для побудови діалогового вікна введення значень (Рис. 27).



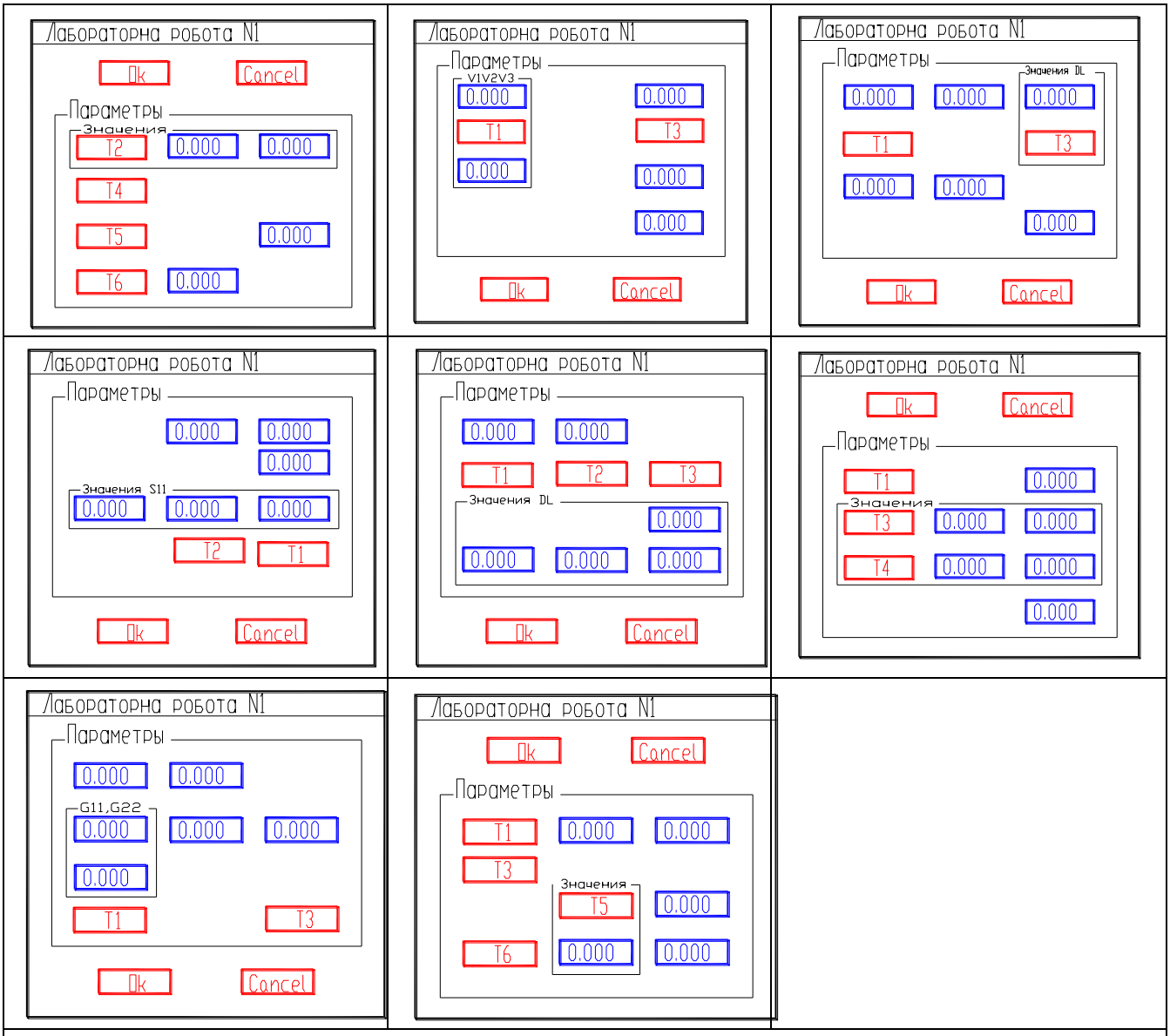
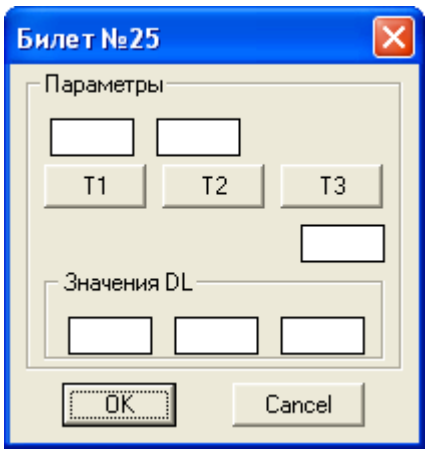


Рис. 27 Завдання для білетів по САПР, створення діалогових вікон

Приклади виконання програм для створення діалогових вікон

Приклад 1.

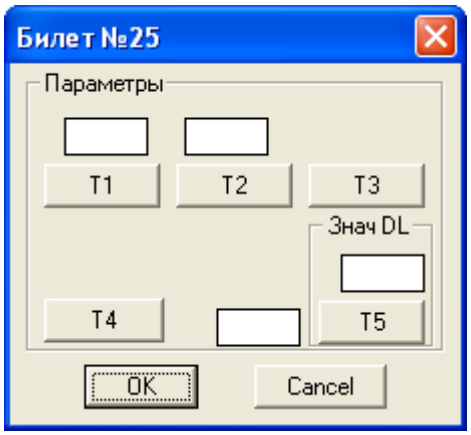


Виконання програми dial_1

```
dial_1:dialog{label="Билет №25";
 :boxed_column{label="Параметры";
  :row{
      :edit_box{edit_width=5;edit_limit=5;key="E1";}
      :edit_box{edit_width=5;edit_limit=5;key="E2";}
      :spacer{width=10;}
  }
  :row{
       :button{label="T1";key="T1";}
       :button{label="T2";key="T2";}
       :button{label="T3";key="T3";}
  }
```

```
 :row{
  :spacer{width=20;}
   :edit_box{edit_width=5;edit_limit=5;key="E3";}
 }
 :boxed_row{label="Значения DL";
   :edit_box{edit_width=5;edit_limit=5;key="E4";}
   :edit_box{edit_width=5;edit_limit=5;key="E5";}
   :edit_box{edit_width=5;edit_limit=5;key="E6";}
  }
 } // boxed_column
ok_cancel;
}
```

Приклад 2.



Виконання програми dial_2

```
dial_2:dialog{label="Билет №25";
 :boxed_column{label="Параметры";
 :row{
  :edit_box{edit_width=5;edit_limit=5;key="E1";}
  :edit_box{edit_width=5;edit_limit=5;key="E2";}
  :spacer{width=10;}
 }
 :row{
   :button{label="T1";key="T1";}
   :button{label="T2";key="T2";}
   :button{label="T3";key="T3";}
 }
 :row{
   :column{
     :spacer{width=10;}
     :button{label="T4";key="T4";}
    }
    :column{
      :spacer{width=10;}
      :edit_box{edit_width=5;edit_limit=5;key="E4";}
     }
     :column{label="Знач DL";
       :edit_box{edit_width=5;edit_limit=5;key="E5";}
       :button{label="T5";key="T5";}
      }
   } // row
 } //  boxed_column
ok_cancel;
}
```

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

1. *Руководство* пользователя системы AutoCAD.
2. Автоматизація графічно-конструкторських робіт у процесі проектування хімічного устаткування в системі AutoCAD: Навч.посіб. / В.Ю. Щербина, О.С. Сахаров, О.В. Гондлях, В.І. Сівецький. – К.: ІВЦ „Видавництво „Політехніка”, 2003. – 152с.: іл
3. Автоматизоване проектування черв’ячного устаткування: Навч. посіб. / В.І. Сівецький, В.Ю. Щербина – К.: ІВЦ “Видавництво «Політехніка»”, 2005. – 184с.: іл
4. САПР. Програмний комплекс АПРОКС в розрахунках машин та апаратів хімічних виробництв: Навч. посіб. / О.В. Гондлях, О.С. Сахаров, В.І. Сівецький, В.Ю. Щербина, Р.М. Пашинський, А.О. Чемерис. – К.: ТОВ
5. САПР. Інтегрована система моделювання технологічних процесів і розрахунку обладнання хімічної промисловості: Навч. посіб. / О.С.Сахаров, В.Ю.Щербина, О.В. Гондлях, В.І. Сівецький. – К.: ТОВ “Поліграф Консалтинг”, 2006. – 156с.: іл
6. САПР. Чисельне моделювання гідромеханічних процесів та НДС суцільних середовищ при термосилових навантаженнях: Навч. посіб. / О.С. Сахаров, В.Ю. Щербина, В.І. Сівецький, О.В. Гондлях – К.: ТОВ “Поліграф Консалтинг”, 2007. – 180с.: іл
7. *Кречко Ю. А*. AutoCAD: программирование и адаптация. – М.: ДИАЛОГ-МИФИ, 1995. – 240 с.
8. Сайт „Компьютерное конструирование” http://tomskcad.city.tomsk.net/CAD/CAD.htm.
9. Список функций Visual Lisp http://www.cad.dp.ua/stats/vlisp.php
10. Кудрявцев Е.М. AutoLISP. Программирование в AutoCAD 14 / Кудрявцев Е.М. -М.: «ДМК», 1999 - 368 с, ил.
11. Энкарначчо Ж., Шлехтендаль Э. Автоматизированное проектирование: Основные понятия и архитектура систем: Пер. с англ. – М.: Радио и связь, 1986. – 288 с.
12. 12.Бугрименко Г. А., Лямке В. Н., Шейбонене Э.-К. С. Автоматизация конструирования на ПЭВМ с использованием системы AutoCAD. – М: Машиностроение, 1993. – 336 с.
13. Хювенен Э., Сеппянен Й. Мир Лиспа: В 2 т. – М.: Мир, 1990.
14. *Гладков С. А*. Програмирование на языке Автолисп в системе САПР AutoCAD. – М.: ДИАЛОГ-МИФИ, 1991. – 96 с.
15. *Бугрименко А. С.* Автоматизация конструирования на ЭВМ с использованием системы AutoCAD. – М.: Машиностроение, 1993. – 336 с.
16. *Батбаро В. А., Заблоцкий Д. В., Кмеллер М. И.* AutoCAD. Полезные рецепты – М.: Радио и связь, 1994. – 208 с.
17. *Крюков А. П., Родионов А. Я.* Программирование на языке R-Lisp –М.: *Радио* и связь, 1991. – 192 с.
18. *Paulson J. Layer* upon layer of goodies with A'LISP // CAD user. – 1989. – N6. – P. 80.

ДОДАТКИ

Додаток 1. Список системних змінних

Системна змінна	Значення за замовчуванням	Опис
ACADLSPASDOC		Управління режимом завантаження файлу acad.lsp
ACADPREFIX	"C:\Program Files..." (read only)	Шляхи пошуку файлів підтримки й драйверів
ACADVER	"15.06" (read only)	Номер версії AutoCAD
ACISOUTVER	40	Код версії ACIS-файлів, створюваний командою ACISOUT
ADCSTATE		Ознака наявності на екрані Центра керування
AFLAGS	0	Сума встановлених бітових прапорів для команди ATTDEF
ANGBASE	0	Напрямок нульового кута в поточної СКК
ANGDIR	0	Код напрямку відліку кутів у поточної СКК
APBOX	0	Стан прицілу автоприв’язки
APERTURE	10	Розмір прицілу перехрестя курсору
AREA	0.0000 (read only)	Останнє обчислене значення площі
ATTDIA	0	Режим команди INSERT при уведенні значень атрибутів (з діалоговим вікном або без нього)
ATTMODE	1	Режим видимості атрибутів
ATTREQ	1	Режим використання командою INSERT атрибутів за замовчуванням
AUDITCTL	0	Режим створення файлу протоколу перевірки (з розширенням adt) при виконанні команди AUDIT
AUNITS	0	Кутові одиниці виміру
AUPREC	0	Точність у кутових розмірах
AUTOSNAP	55	Управління маркером, підказкою й магнітом автоприв’язки
BACKZ	0.0000 (read only)	Зсув задньої січної площини від площини цілі для поточного видового екрана в умовних одиницях
BINDTYPE	0	Форма імен залежних символів при впровадженні зовнішніх посилань і при редагуванні входжень
BLIPMODE	0	Режим видимості маркерів
CDATE	20090307.15233597(read only)	Поточна дата й час доби
CECOLOR	"BYLAYER"	Поточний колір
CELTSCALE	1.00000	Поточний масштаб типу ліній
CELTYPE	"BYLAYER"	Поточний тип лінії
CELWEIGHT	-1	Поточна вага ліній
CHAMFERA	10.00000	Перша довжина фаски
CHAMFERB	10.00000	Друга довжина фаски

CHAMFERC	20.00000	Довжина фаски
CHAMFERD	0	Кут фаски
CHAMMODE	0	Поточний метод створення фасок у команді CHAMFER
CIRCLE-PAD		Радіус, пропонований за замовчуванням у команді CIRCLE
CLAYER	"0"	Поточний шар
CLEARSCREENSTATE		Стан очищення екрана
CMDACTIVE	1(read only)	Бітовий код, що вказує, чи активні в цей момент звичайна команда, прозора команда, пакет команд й (або) діалогове вікно
CMDECHO	1	Режим луни-виводу (відображення підказок AutoCAD) при виконанні функцій command й vl-cmdf мови AutoLISP
CMDNAMES	"SETVAR" (read only)	Англійське ім'я поточної команди
CMLJUST	0	Поточний тип розташування мультиліній
CMLSCALE	20.0000	Поточний масштаб побудови мультиліній
CMLSTYLE	"STANDARD"	Поточний стиль мультиліній
COMPASS	0	Режим показу тривимірного компаса на поточному видовому екрані
COORDS	2	Режим відновлення координат у статусному рядку
CPLOTSTYLE	"ByColor"	Поточний стиль друку
CPROFILE	"<<VAnilLA>>"(read only)	Ім'я поточного профілю
СТАВШИ	"Model"	Назва поточної вкладки в малюнку (Model або один з листів)
CURSORSIZE	100	Розмір перехрестя у відсотковому відношенні до розміру екрана
CVPORT	2	Номер активного видового екрана
DATE	2454898.64147533(read only)	Поточна дата й час доби
DBMOD	4(read only)	Бітовий код, що визначає стан змін у малюнку з моменту останнього збереження
DCTCUST	"C:\Program	Шлях й ім'я файлу поточного допоміжного словника перевірки орфографії
DCTMAIN	"enu"	Ім'я файлу поточного основного орфографічного словника
DEFLPLSTYLE	"ByColor"	Стиль друку за замовчуванням для нових шарів
DEFPLSTYLE	"ByColor"	Стиль друку за замовчуванням для нових об'єктів
DELOBJ	1	Режим збереження або видалення примітивів, використовуваних для створення інших об'єктів
DEMANDLOAD	3	Реакція AutoCAD при виявленні в малюнку об'єктів, створених іншими додатками

DIASTAT	1(read only)	Режим виходу з останнього діалогового вікна
DIMADEC	0	Точність кутових розмірів
DIMALT	OFF	Режим альтернативних одиниць у розмірах
DIMALTD	3	Точність для альтернативних одиниць виміру
DIMALTF	0.0394	Коефіцієнт перерахування для альтернативних одиниць у розмірах
DIMALTRND	0.0000	Точність округлення альтернативних одиниць
DIMALTTD	3	Точність в альтернативному допуску
DIMALTTZ	0	Режим придушення нулів у допусках
DIMALTU	2	Формат одиниць для всіх видів альтернативних розмірів, за винятком кутових
DIMALTZ	0	Режим придушення нулів в альтернативних розмірних одиницях
DIMAPOST	""	Рядок, що приєднує в кінець будь-яких розмірів в альтернативних одиницях виміру, за винятком кутових
DIMASO	ON	Режим асоціативності розмірних об'єктів, як блоків (застаріла змінна)
DIMASSOC	2	Режим асоціативного зв'язку розмірів і вимірюваних об'єктів малюнка
DIMASZ	2.5000	Величина стрілок на кінцях розмірної лінії й винесення
DIMATFIT	3	Режим розміщення розмірного тексту й стрілок, якщо для одного й іншого недостатньо місця між лініями що виносяться
DIMAUNIT	0	Формат одиниць виміру для кутових розмірів
DIMAZIN	0	Режим придушення нулів у кутових розмірах
DIMBLK	""	Блок, використовуваний для стрілок на кінцях розмірних ліній і винесень
DIMBLK1	""	Стрілка на першому кінці розмірної лінії (при включенії змінній DIMSAH)
DIMBLK2	""	Стрілка на другому кінці розмірної лінії (при включенії змінній DIMSAH)
DIMCEN	2.5000	Режим нанесення маркерів центра й центрових ліній кіл і дуг у командах DIMCENTER, DIMDIAMETER й DIMRADIUS
DIMCLRD	0	Кольори розмірних ліній, стрілок і виносних ліній
DIMCLRE	0	Кольори виносних ліній розмірів
DIMCLRT	0	Кольори розмірного тексту
DIMDEC	2	Точність в основних одиницях
DIMDLE	0.0000	Величина виступу розмірної лінії за виносну лінію при використанні зарубок
DIMDLI	3.7500	Величина відступу розмірної лінії в базових розмірах
DIMDSEP	","	Символ десяткового роздільника при десятковому форматі виводу чисел
DIMEXE	1.2500	Величина виступу виносної лінії за розмірну лінію
DIMEXO	0.6250	Величина зсуву виносних ліній щодо заданих початкових точок
DIMFIT	3	Застаріла змінна (замінена на DIMATFIT й DIMTMOVE)
DIMFRAC	0	Формат дробів у випадках, коли DIMLUNIT має значення

		4 (архітектурні) або 5 (дробові)
DIMGAP	0.6250	Відстань, що залишається порожньою при зображенні розмірного тексту в розриві розмірної лінії
DIMJUST	0	Режим положення розмірного тексту по горизонталі
DIMLDRBLK	""	Тип стрілки на кінці винесення
DIMLFAC	1.0000	Масштабний коефіцієнт для лінійних розмірів
DIMLIM	OFF	Генерація розмірного тексту у форматі "межі"
DIMLUNIT	2	Формат одиниць для всіх видів розмірів, крім кутових
DIMLWD	-2	Вага для розмірних ліній
DIMLWE	-2	Вага для виносних ліній
DIMPOST	""	Обумовлене користувачем закінчення розмірного тексту
DIMRND	0.0000	Точність округлення всіх лінійних розмірів
DIMSAH	OFF	Керування використанням блоків розмірних стрілок
DIMSCALE	1.0000	Глобальний масштабний коефіцієнт, застосовується до всіх розмірних змінних, за винятком допусків, обмірюваних довжин, координат і кутів
DIMSD1	OFF	Режим придушення першої розмірної лінії
DIMSD2	OFF	Режим придушення другої розмірної лінії
DIMSE1	OFF	Режим придушення першої виносної лінії
DIMSE2	OFF	Режим придушення другий виносної лінії
DIMSHO	ON	Режим зміни розмірних об'єктів при відстеженні
DIMSOXD	OFF	Придушення зображення розмірної лінії за межами відповідних виносних ліній
DIMSTYLE	"ISO-25"(read only)	Поточний розмірний стиль
DIMTAD	1	Режим вертикальності тексту щодо розмірної лінії
DIMTDEC	2	Кількість десяткових знаків для значень допусків в основних одиницях
DIMTFAC	1.0000	Масштабний коефіцієнт для розрахунку висоти тексту дробових частин розмірів і допусків
DIMTIH	OFF	Режим орієнтації розмірного тексту для лінійних розмірів, радіусів і діаметрів, якщо текст уписується між виносними лініями
DIMTIX	OFF	Режим примусового розміщення розмірного тексту між виносними лініями
DIMTM	0.0000	Нижнє граничне відхилення або найменший граничний розмір, якщо системні змінні DIMTOL або DIMLIM включені
DIMTMOVE	0	Правила переміщення розмірного тексту при редагуванні
DIMTOFL	ON	Режим малювання розмірної лінії між виносними лініями, якщо стрілки розміщаються поза виносними лініями
DIMTOH	OFF	Режим орієнтації розмірного тексту поза виносними лініями
DIMTOL	OFF	Код проставляння допусків в основних розмірах
DIMTOLJ	0	Вирівнювання допусків щодо розмірного тексту по вертикалі
DIMTP	0.0000	Верхнє граничне відхилення або найбільший граничний розмір, якщо системні змінні DIMTOL або DIMLIM

		включені
DIMTSZ	0.0000	Величина насічки, зображуваної замість стрілки в лінійних розмірах, радіусах і діаметрах
DIMTVP	0.0000	Регулювання вертикального положення розмірного тексту над або під розмірною лінією
DIMTXSTY	"Standard"	Текстовий стиль для розміру
DIMTXT	2.5000	Висота розмірного тексту, якщо використовуваний текстовий стиль не має фіксовану висоту
DIMTZIN	8	Режим придушення нулів у допусках
DIMUNIT	2	Застаріла змінна (замінена на DIMLUNIT й DIMFRAC)
DIMUPT	OFF	Режим дії курсору при користувальницькому розташуванні тексту
DIMZIN	8	Режим придушення нулів в основних одиницях виміру
DISPSILH	0	Режим показу кромок силуету твердотільних об'єктів у каркасному режимі
DISTANCE	0.0000(read only)	Остання відстань, обчислена командою DIST
DONUTID	0.5000	Останній внутрішній діаметр кільця
DONUTOD	1.0000	Останній зовнішній діаметр кільця
DRAGMODE	2	Режим відображення об'єктів при буксируванні
DRAGP1	10	Частота регенерації об'єкта при відстеженні
DRAGP2	25	Частота регенерації об'єкта при швидкому відстеженні
DWGCHECK	0	Код необхідності перевірки, чи редагувався малюнок востаннє програмою, відмінною від AutoCAD
DWGCODEPAGE	"ANSI_1251"(read only)	Зберігає те ж значення, що й SYSCODEPAGE (для цілей сумісності)
DWGNAME	"Drawing1.dwg"(read only)	Ім'я поточного малюнка
DWGPREFIX	"D:\ACAD\MDT6\"(read only)	Шлях до поточного малюнка
DWGTITLED	0(read only)	Код, що визначає, чи привласнено поточному малюнку ім'я
EDGEMODE	0	Режим подовження об'єктів, обраних у якості ріжучих і граничних кромок у командах TRIM й EXTEND
ELEVATION	0.0000	Діючий рівень (зсув по поточній осі Z) побудови об'єктів у поточному видовому екрані
ENTEXTS	0	Код точності обчислення границь примітивів
EXPERT	1	Режим складності виведених підказок
EXPLMODE		Режим можливості розчленовування командою EXPLODE блоків з різними масштабними коефіцієнтами
EXTMAX	-1.0000E+20,-1.0000E+20,-1.0000E+20(read only)	Права верхня точка, у СКК, границь малюнка в поточному просторі
EXTMIN	1.0000E+20,1.0000E+20,1.0000E+20(read only)	Ліва нижня точка, у СКК, границь малюнка в поточному просторі

EXTNAMES	1	Код обмежень імен елементів символьних таблиць (типів ліній, шарів й ін.)
FACETRATIO	0	Режим щільності мережного представлення лекальних граней тіл
FACETRES	0.5000	Регулювання гладкості розфарбованих і тонованих об'єктів, а також об'єктів з подавленими прихованими лініями
FILEDIA	1	Режим використання діалогових вікон вибору файлів
FILLETRAD	10.0000	Поточний радіус сполучення
FILLMODE	1	Режим зафарбування мультиліній, смуг, фігур, всіх штрихувань (у тому числі суцільних) і поліліній ненульової ширини
FONTALT	"simplex.shx"	Ім'я файлу альтернативного шрифту
FONTMAP	"C:\Program	Ім'я файлу підстановки шрифтів
FRONTZ	0.0000(read only)	Зсув передньої січної площини для поточного видового екрана
FULLOPEN	1(read only)	Індикатор повного або часткового відкриття поточного малюнка
GFANG		Кут градієнтного заливання
GFCLR1		Перші кольори градієнтної заливання
GFCLR2		Другі кольори градієнтної заливання
GFCLRLUM		Співвідношення між світлом і тінню при градієнтному заливанню
GFCLRSTATE		Ознака використання одного або двох квітів при градієнтному заливанню
GFNAME		Номер зразка градієнтного заливання
GFSHIFT		Ознака зсуву центра заливання
GRIDMODE	0	Код стану режиму GRID
GRIDUNIT	10.0000,10.0000	Розмір комірки сітки на поточному видовому екрані по осях X й Y
GRIPBLOCK	0	Режим показу ручок усередині блоків
GRIPCOLOR	5	Кольори невибраних ручок
GRIPHOT	1	Кольори обраних ручок
GRIPHOVER		Кольори невибраної ручки при положенні над нею курсору
GRIPOBJLIMIT		Максимальна кількість одночасно показуваних ручок
GRIPS	1	Режим показу ручок обраних об'єктів
GRIPSIZE	3	Розмір ручки (у пікселях)
GRIPTIPS		Режим виводу підказок при положенні курсору над невибраною ручкою
HALOGAP	0	Допуск на розміри видимих об'єктів для команди HIDE
HANDLES	1(read only)	Режим присвоєння міток об'єктам
HIDEPRECISION	0	Режим точності приховання невидимих ліній і розфарбовування
HIDETEXT	ON	Режим обробки текстів командою HIDE
HIGHLIGHT	1	Режим підсвічування об'єктів при виборі
HPANG	0	Кут нахилу зразка штрихування за замовчуванням
HPASSOC		Режим асоціативності штрихувань і заливань
HPBOUND	1	Код типу об'єкта, створюваного командами BHATCH й

		BOUNDARY
HPDOUBLE	0	Режим подвійного штрихування (хрест-навхрест)
HPNAME	"ANGLE"	Ім'я зразка штрихування за замовчуванням
HPSCALE	1.0000	Масштаб штрихування за замовчуванням
HPSPACE	1.0000	Відстань між штриховими лініями за замовчуванням для користувальницьких штрихувань
HYPERLINKBASE	""	Шлях, використовуваний для відносних гіперпосилань
IMAGEHLT	0	Режим підсвічування растрових зображень при виборі
INDEXCTL	0	Режим створення шарового й просторового індексів
INETLOCATION	"http://www.autodesk.com"	Початкова URL-адреса браузера
INSBASE	0.00,0.00,0.00	Базова точка вставки, установлювана командою BASE
INSNAME	""	Ім'я блоку за замовчуванням для команди INSERT
INSUNITS	4	Одиниці виміру для блоків і растрових зображень, що беруться із Центра керування
INSUNITSDEFSOURCE	4	Одиниці виміру у файлі-джерелі
INSUNITSDEFTARGET	4	Одиниці виміру в цільовому малюнку
INTERSECTIONCOLOR		Кольори ліній перетинання мереж
INTERSECTIONDISPLAY		Ознака відображення ліній перетинання мереж при розфарбовуванні й приховуванні невидимих ліній
ISAVEBAK	1	Режим створення резервної копії малюнка (з розширенням bak)
ISAVEPERCENT	50	Розмір невикористовуваного простору усередині малюнка, при перевищенні якого виконується повне збереження
ISOLINES	4	Число утворюючих на криволінійних ділянках поверхні об'єкта
LASTANGLE	0(read only)	Кінцевий кут останньої побудованої дуги
LASTPOINT	0.00,0.00,0.00	Остання зазначена точка в поточній СКК
LASTPROMPT	"LASTANGLE	Останній текст, виведений у командний рядок
LAYOUTREGENCTL	2	Режими регенерації зображення на вкладках листів
LENSLENGTH	50.0000(read only)	Фокусна відстань (у міліметрах) при побудові перспективної проекції в поточному видовому екрані
LIMCHECK	0	Режим перевірки виходу за ліміти малюнка
LIMMAX	420.0000,297.0000	Правий верхній кут зони лімітів малюнка, у СКК
LIMMIN	0.0000,0.0000	Лівий нижній кут зони лімітів малюнка, у СКК
LISPINIT	1	Режим збереження в пам'яті функцій і змінних (LISP-символів) при відкритті нового малюнка (в одно-документному режимі)
LOCALE	"RUS"(read only)	Код ISO мови поточної версії AutoCAD
LOCALROO		Шлях до папки, використовуваної для зберігання

TPREFIX		настроювань користувача
LOGFILEMODE	0	Режим запису вмісту текстового вікна у файл журналу (з розширенням log)
LOGFILENAME	""(read only)	Ім'я файлу журналу (разом зі шляхом) для поточного малюнка
LOGFILEPATH	""	Шлях до файлів журналів всіх малюнків сеансу
LOGINNAME	""(read only)	Системне ім'я користувача
LTSCALE	1.0000	Глобальний масштаб типів лінії
LUNITS	2	Система виміру лінійних одиниць
LUPREC	4	Точність відображення лінійних величин
LWDEFAULT	25	Значення, що відповідає ваги лінії DEFAULT
LWDISPLAY	OFF	Режим відображення ваг ліній на екрані
LWUNITS	1	Одиниці виміру для ваг ліній (дюйми або міліметри)
MAXACTVP	64	Максимальна кількість видових екранів, які можуть бути активні одночасно
MAXSORT	200	Максимальна кількість рядків, яку можна відсортувати в діалогових вікнах при виводі списків (шарів і т.п.)
MBUTTONPAN	1	Режим дії третьої кнопки або коліщати пристрою вказівки (наприклад, миші)
MEASUREINIT	1	Система одиниць (британські або метричні) для нових малюнків
MEASUREMENT	1	Система одиниць (британські або метричні) для поточного малюнка
MENUCTL	1	Режим зміни сторінок екранного меню
MENUECHO	0	Сума бітових прапорів керування луною-виводом і запитами меню
MENUNAME	"D:\Acad\MDT6\Support\acad"(read only)	Ім'я файлу базового меню (включаючи шлях)
MIRRTEXT	1	Режим симетрування тексту в команді MIRROR
MODEMACRO	""	Додатковий текст, відображуваний на початку рядка стану
MTEXTED	"Bíympeííué"	Ім'я редактора (убудованою або зовнішнього), використовуваного командою MTEXT
MTEXTFIXED		Керування місцем розташування вікна редактори мультитекста
MTJIGSTRING		Текст, "що прив'язується" до курсору при вході у вікно мультитекста
MYDOCUMENTSPREFIX		Шлях до папки Мої документи поточного користувача
NOMUTT	0	Режим повного придушення повідомлень у командному рядку при роботі пакетів і функцій AutoLISP
OBSCUREDCOLOR	0	Кольори, що створюються для невидимих ліній
OBSCURED	0	Тип лінії, установлюваний для невидимих ліній

LTYPE		
OFFSETDIST	-1.0000	Поточний зсув у команді OFFSET
OFFSETGAPTYPE	0	Метод обробки з'єднань сусідніх сегментів командою OFFSET
OLEHIDE	0	Режим показу й печатки OLE-об'єктів
OLEQUALITY	1	Режим якості й печатки впроваджених OLE-об'єктів
OLESTARTUP	0	Режим завантаження батьківських додатків для OLE-об'єктів при печатці
ORTHOMODE	0	Поточний стан режиму ортогональності
OSMODE	0	Сума бітових прапорів поточних режимів об'єктної прив'язки
OSNAPCOORD	2	Режим придушення вводять координатами, що, що течуть режимів прив'язки
PALETTEOPAQUE		Керування прозорістю вікон і палітр
PAPERUPDATE	0	Режим виводу попередження при невідповідності розмірів листа формату друку
PDMODE	0	Поточний режим відображення точок
PDSIZE	0.0000	Розмір символу, що відображає точку
PEDITACCEPT	0	Автоматичне перетворення обраного об'єкта в полілінію в команді PEDIT
PELLIPSE		Режим малювання еліпсів (точних або апроксимованих)
PERIMETER	0.0000(read only)	Останнє обчислене значення периметра
PFACEVMAX	4(read only)	Максимальне число вершин на одну грань у команді PFACE
PICKADD	1	Режим додавання об'єктів до вже обраних
PICKAUTO	1	Режим автоматичного створення рамки вибору при вказані мишею по вільному місцю
PICKBOX	3	Висота прицілу вибору об'єктів у пікселах
PICKDRAG	0	Спосіб створення рамки вибору
PICKFIRST	1	Режим попереднього вибору об'єктів, для їхнього використання в наступній команді
PICKSTYLE	1	Режим вибору об'єктів за допомогою груп і за допомогою асоціативного штрихування
PLATFORM	"Microsoft	Тип платформи (операційної системи)
PLINEGEN	0	Режим генерації типу лінії у вершинах двовимірних поліліній
PLINETYPE	2	Керування типом створюваних двовимірних поліліній (компактних або докладних)
PLINEWID	0.0000	Ширина двовимірної полілінії за замовчуванням
PLOTROTMODE	2	Орієнтація креслення
PLQUIET	0	Керування виводом необов'язкових діалогових вікон і повідомлень про некритичні помилки при пакетному друкові й виконанні пакетних файлів

POLARADDANG	""	Додаткові кути полярного відстеження
POLARANG	90	Основний кут при полярному відстеженні
POLARDIST	0.0000	Крок полярної прив'язки
POLARMODE	0	Керування режимами полярного й об'єктного відстеження
POLYSIDES	4	Число сторін багатокутника, пропоноване за замовчуванням командою POLYGON
POPUPS	1(read only)	Код стану відеомонітора
PRODUCT		Назва програмного продукту
PROGRAM		Ім'я основного файлу програми
PROJECTNAME	""	Ім'я проекту, у якому шукаються зовнішні посилання й растрові зображення
PROJMODE	1	Режим проектування для операцій обрізки й подовження
PROXYGRAPHICS	1	Режим збереження об'єктів-заступників у малюнку
PROXYNOTICE	1	Режим виводу повідомлення при створенні об'єкта-заступника
PROXYSHOW	0	Режим показу об'єктів-заступників у малюнку
PROXYWEBSEARCH	1	Режим пошуку програми Object Enabler для обробки об'єктів-заступників
PSLTSCALE	1	Режим масштабування типів ліній у просторі листа
PSPROLOG	""	Ім'я розділу файлу acad.psf, який потрібно прочитати команді PSOUT
PSQUALITY	75	Керування якістю відображення зображень формату PostScript
PSTYLEMODE	1(read only)	Тип стилів друку (кольорові або іменовані) у поточному малюнку
PSTYLEPOLICY	1	Режим зв'язку кольорів об'єкта з його стилем друку
PSVPSCALE	0.00000000	Масштаб для знову створюваних видових екранів
PUCSBASE	""	Ім'я СКК, на якій базуються ортогональні системи координат (тільки для простору листа)
QTEXTMODE	0	Режим швидкого (контурного) відображення тексту
RASTERPREVIEW	1	Режим збереження разом з малюнком растрового зразка для попереднього перегляду
REFEDITNAME	""(read only)	Ім'я зовнішнього посилання або блоку, що редагує в цей момент командою REFEDIT
REGENMODE	1	Режим автоматичної регенерації малюнка
RE-INIT		Сума бітових прапорів переініціалізації дигітайзера, його порту й перезавантаження файлу acad.pgp
REMEMBERFOLDERS	1	Режим запам'ятовування шляхів до файлів малюнків
REPORTERROR		Керування виводом повідомлення про можливості відправлення повідомлення про помилку у фірму Autodesk
ROAMABLE		Шлях до папки, що дозволяє підтримувати пошук файлів у

ROOTPREFIX		мережі
RTDISPLAY	1	Керування показом растрових зображень при панорамуванні й зумуванні в реальному часі (команди ZOOM й PAN)
SAVEFILE	""(read only)	Ім'я файлу автоматичного збереження
SAVEFILEPATH	"C:\DOCUME~…\"	Шлях до папки, у яку здійснюється автоматичне збереження файлів сеансу AutoCAD
SAVENAME	""(read only)	Ім'я файлу, під яким збережений поточний малюнок (разом зі шляхом)
SAVETIME	120	Інтервал автоматичного збереження малюнка у хвилинах
SCREENBOXES	0(read only)	Кількість рядків у зоні екранного меню графічного екрана
SCREENMODE	3(read only)	Бітовий код, що вказує стан екрана AutoCAD (графічний, текстовий)
SCREENSIZE	1270.0000,807.0000(read only)	Розмір поточного видового екрана в пікселах (по осях X і У)
SDI	0	Режим AutoCAD (одне- або багатодокументний)
SHADEDGE	3	Режим показу ребер і граней при тонуванні
SHADEDIF	70	Код відношення розсіяної освітленості до освітленості дифузійного відбиття
SHORTCUTMENU	0	Режим доступу до контекстного меню
SHPNAME	""	Ім'я форми за замовчуванням
SIGWARN		Код виводу вікна з інформацією про електронний цифровий підпис
SKETCHINC	1.0000	Довжина сегментів у команді SKETCH
SKPOLY	0	Режим генерації відрізків або поліліній у команді SKETCH
SNAPANG	0	Кут повороту сітки крокової прив'язки (у поточної СКК) для поточного видового екрана
SNAPBASE	0.0000,0.0000	Початкова точка сітки крокової прив'язки для поточного видового екрана (у поточної СКК)
SNAPISOPAIR	0	Код поточної площини ізометрії для поточного видового екрана
SNAPMODE	0	Поточний стан режиму SNAP
SNAPSTYL	0	Стиль крокової прив'язки на поточному видовому екрані
SNAPTYPE	0	Тип крокової прив'язки
SNAPUNIT	10.0000,10.0000	Параметри сітки крокової прив'язки в поточному видовому екрані
SOLIDCHECK	1	Режим перевірки цілісності тіл
SORTENTS	96	Режим сортування об'єктів
SPLFRAME	0	Режим відображення каркасів сплайнів, згладжених сплайнами поліліній і згладжених мереж
SPLINESEGS	8	Код керування апроксимацією полілінії при згладжуванні її сплайном
SPLINETYPE	6	Тип кривої, що згладжує, в опції Spline команди PEDIT

STANDARDSVIOLATION	1	Режим виводу повідомлень про порушення стандартів при нормоконтролі
STARTUP		Керування типом вікна при створенні нових малюнків
SURFTAB1	6	Щільність у напрямку M для команд побудови мереж
SURFTAB2	6	Щільність у напрямку N для команд побудови мереж
SURFTYPE	6	Тип поверхні згладжування, що використається опцією Smooth команди PEDIT
SURFU	6	Щільність мережі в напрямку M для опції Smooth команди PEDIT
SURFV	6	Щільність мережі в напрямку N для опції Smooth команди PEDIT
SYSCODEPAGE	"ANSI_1251"(read only)	Системна кодова таблиця, задана у файлі acad.xmx
TABMODE	0	Керування використанням режиму Tablet
TARGET	0.00,0.00,0.00 (read only)	Положення мети в координатах СКК на поточному видовому екрані
TDCREATE	2454898.641224 13(read only)	Дата й час (місцеве) створення поточного малюнка
TDINDWG	0.00029676(read only)	Загальний час редагування
TDUCREATE	2454898.516224 13(read only)	Дата й час (всесвітнє) створення поточного малюнка
TDUPDATE	2454898.641224 13(read only)	Дата й час (місцеве) останнього збереження малюнка
TDUSRTIMER	0.00029677(read only)	Таймер користувача
TDUUPDATE	2454898.516224 13(read only)	Дата й час (всесвітнє) останнього збереження малюнка
TEMPPREFIX	""(read only)	Імена папок для зберігання тимчасових файлів
TEXTEVAL	0	Режим обробки LISP-виражень при запиті тексту
TEXTFILL	1	Режим заливання тексту, виконаного шрифтами True Typi, при друкові, експорті командою PSOUT і тонуванні
TEXTQLTY	50	Якість накреслення шрифтів True Type при виводі текстів на друк, експорті командою PSOUT і тонуванні
TEXTSIZE	2.5000	Висота букв, пропонована за замовчуванням для нового однорядкового тексту
TEXTSTYLE	"Standard"	Поточний текстовий стиль
THICKNESS	0.0000	Поточна тривимірна висота
TILEMODE	1	Режим простору листа або моделі
TOOLTIPS	1	Керування виводом спливаючих підказок
TPSTATE		Код видимості панелі Tool Palettes
TRACEWID	1.0000	Ширина смуги за замовчуванням
TRACKPAT	0	Керування показом пунктирних ліній при полярному й

H		об'єктному відстеженні
TRAYICONS		Керування показом значків повідомлень
TRAYNOTIFY		Керування виводом повідомлень
TRAYTIMEOUT		Тривалість показу повідомлень
TREEDEPTH	3020	Максимальна глибина розгалуження просторового індексу
TREEMAX	10000000	Максимальна кількість вузлів у просторовому індексі
TRIMMODE	1	Режим зміни обраних об'єктів у командах CHAMFER й FILLET
TSPACEFAC	1.0000	Відношення міжстрокового інтервалу до висоти багатострокового тексту
TSPACETYPE	1	Стиль міжстрокового інтервалу в багатостроковому тексті
TSTACKALIGN	1	Вирівнювання двоповерхового елемента мультитекста по вертикалі
TSTACKSIZE	70	Відношення висоти дробової частини двоповерхового тексту до висоти звичайного тексту (у відсотках)
UCSAXISANG	90	Значення кута за замовчуванням при повороті СКК навколо однієї з її осей за допомогою опцій X, Y або Z команди UCS
UCSBASE	""	Ім'я СКК, на якій базуються ортогональні системи координат
UCSFOLLOW	0	Установка виду в плані при переході від однієї СКК до іншої
UCSICON	3	Включення видимості піктограми СКК на поточному видовому екрані
UCSNAME	""(read only)	Ім'я поточної системи координат для поточного видового екрана в поточному просторі
UCSORG	0.00,0.00,0.00(read only)	Початок поточної системи координат для поточного видового екрана в поточному просторі
UCSORTHO	0	Режим відновлення ортогональної СКК у момент установки відповідного ортогонального виду
UCSVIEW	1	Режим збереження поточної СКК разом з іменованим видом
UCSVP	1	Режим зміни СКК на видових екранах слідом за системою координат, установленої на поточному видовому екрані
UCSXDIR	1.00,0.00,0.00(read only)	Напрямок осі X поточної СКК на поточному видовому екрані в поточному просторі
UCSYDIR	0.0000,1.0000,0.0000(read only)	Напрямок осі Y поточної СКК на поточному видовому екрані в поточному просторі
UNDOCTL	5(read only)	Сума бітових прапорів, що вказують стан Auto й Control команди UNDO
UNDOMARKS	0(read only)	Кількість міток, заданих в опції Mark команди UNDO для керування відміною
UNITMODE	0	Режим відображення чисел із дробовою частиною, у футах і дюймах й у топографічних одиницях
USERI1 (2-5)		Використовуються для зберігання цілих чисел, заданих

		користувачем
USERR1(2-5)		Використовуються для зберігання дійсних чисел, заданих користувачем
USERS1(2-5)		Використовуються для зберігання текстових даних користувача
VIEWCTR	531.1431,337.3533,0.00(read only)	Центр поточного видового екрана в координатах СКК
VIEWDIR	0.00,0.00,1.00 (read only)	Напрямок погляду на поточному видовому екрані, в МСК
VIEWMODE	0(read only)	Бітовий код режиму виду на поточному видовому екрані
VIEWSIZE	674.7066(read only)	Висота зображення на поточному видовому екрані
VIEWTWIST	0(read only)	Кут повороту виду для поточного видового екрана
VISRETAIN	1	Керування видимістю, кольорами, типом лінії, вагою лінії й стилями печатки (якщо PSTYLEPOLICY дорівнює 0) залежних від зовнішніх посилань шарів, а також збереженням шляхів доступу для вкладених посилань
VSMAX	3186.85,2024.12,0.00(read only)	Правий верхній кут віртуального екрана поточного видового екрана, виражений у координатах СКК
VSMIN	-2124.5724,-1349.4132,0.0000(read only)	Лівий нижній кут віртуального екрана поточного видового екрана, виражений у координатах СКК
WHIPARC	0	Режим гладкості кіл і дуг на екрані
WHIPTHREAD	ON	Режим перевірки наявності додаткових процесорів
WMFBKGND	OFF	Режим фона малюнка при експорті в метафайл Windows
WMFFOREGND		Режим призначення кольори при експорті в метафайл
WORLDUCS	1(read only)	Індикатор збігу поточної СКК із МСК
WORLDVIEW	1	Код СКК, використовуваної в командах установки видів
WRITESTAT	1(read only)	Указує, чи відкритий поточний файл малюнка тільки для читання
XCLIPFRAME	0	Видимість контурів підрізування зовнішніх посилань і блоків
XEDIT	1	Індикатор, чи може поточний малюнок брати участь в операції редагування входжень, будучи вставленим в інший малюнок як зовнішнє посилання
XFADECTL	50	Керування зниженням інтенсивності відображення об'єктів при редагуванні входжень
XLOADCTL	1	Керування завантаженням файлів зовнішніх посилань й їхніх копій
XLOADPATH	""	Шлях для зберігання тимчасових копій файлів зовнішніх посилань
XREFCTL	0	Визначає, чи створюються файли журналів зовнішніх посилань

XREFNOTIFY		Режим повідомлення про недозволені зовнішні посилання
ZOOMFACTOR	40	Керування швидкістю переміщення курсору

```
(defun MODERS ()
  (modes '(("cmdecho" 0) ("OSMODE" 0) ("APERTURE" 3) ("PICKBOX" 3)))
  (moder) (princ)
)
;=================================================
(defun MODES (a)
; Сохранение старых значений системных переменных и присвоение новых
  (setq MLST (mapcar '(lambda (x) (setq $ (getvar (car x)))
  (setvar (car x) (cadr x))  (list (car x) $)) a ))
)
;=================================================
(defun MODER ( / n)
; Сброс старых значений системных переменных
 (foreach n MLST (setvar (car n) (cadr n)))
)
```

Додаток 2. Алфавітний перелік функцій AutoLISP

Функція	Призначення
+	додавання
-	віднімання
*	множення
/	ділення
=	рівність
/=	нерівність
<	менше
<=	менше або дорівнює
>	більше
>=	більше або дорівнює
~	доповнення до -1
1+	збільшення на 1
1-	зменшення на 1
abs	абсолютна величина
acad_colordlg	виклик діалогового вікна з палітрою кольорів
acad_helpdlg	виклик довідки
acad_strlsort	сортування списку рядків
action_tile	дія кнопки DCL-файлу
add_list	додавання в список діалогового вікна, що розкривається
alert	виклик вікна з повідомленням
alloc	розмір сегмента пам'яті
and	логічне "і"
angle	обчислення кута
angtof	переведення кута в дійсне число
angtos	переведення кута в символьну форму
append	додавання до списку
apply	застосування операції до елементів списку
arx	список завантажених ARX додатків
arxload	завантаження ARX-додатка
arxunload	вивантаження ARX-додатка
ascii	обчислення коду буквеного цифрового символу
assoc	вибір по ключовому елементу списку
atan	арктангенс
atof	переведення із символьної форми в дійсну
atoi	переведення із символьної форми в ціле число
atom	перевірка символу
atoms-family	перелік символів
autoarxload	автозавантаження ARX додатка
autoload	автозавантаження
Boole	булева операція
boundp	перевірка значення символу
caaar,caddr	(car (car (car))) , (car (cdr (cdr))) ,
caar,cadr	(car (car)) , (car (cdr))
car	вибір першого елемента списку
cdr	залишок списку без першого елемента

chr	отримання буквено-цифрового символу по його коду
client_data_tile	прив'язка користувальницьких даних
close	закриття файлу
command	передача команд і параметрів у командний рядок AutoCAD
cond	перевірка умов
cons	створення списку
cos	косинус
cvunit	переведення в інші одиниці виміру
defun	визначення нової функції
defun-q	визначення функції через список
defun-q-list-ref	визначення списку тіла функції
defun-q-list-set	передача списку в тіло функції
dictadd	додавання до словника
dictnext	перехід на наступний запис словника
dictremove	видалення зі словника
dictrename	перейменування в словнику
dictsearch	пошук у словнику
dimx_tile	горизонтальний розмір поля
dimy_tile	вертикальний розмір поля
distance	відстань
distof	переведення рядка в дійсний вид
done_dialog	закриття діалогового вікна
end_image	кінець операції над графічною кнопкою
end_list	кінець операції над списком, що розкривається
entdel	видалення (відновлення) примітива
entget	визначення списку примітива
entlast	визначення останнього елемента рисунка
entmake	створення примітива
entmakex	створення примітива або неграфічного об'єкта
entmod	зміна примітива
entnext	перехід до наступного примітива (підпримітиву)
entsel	вибір об'єкта по вказаній точці
entupd	перемальовування зміненого примітива
eq	рівність об'єктів
equal	рівність із допуском
error	обробка помилок
eval	обчислення списку як функції
exit	вихід
exp	натуральний антилогарифм
expand	розширення пам'яті
expt	зведення в ступінь
fill_image	заповнення графічної кнопки
findfile	пошук файлу
fix	усікання дійсного числа до цілого
float	перетворення числа в дійсне
foreach	застосування операції до елементів списку
function	визначення функції з можливостями оптимізації
gc	зборка сміття

gcd	найбільший загальний дільник
get_attr	значення атрибута, задане полю діалогового вікна в DCL-файлі
get_tile	поточне значення атрибута діалогового вікна
getangle	уведення кута
getcfg	уведення параметрів додатка
getcname	визначення синонімів імен команд
getcorner	уведення точки з рамкою
getdist	уведення відстані
getenv	значення змінної оточення
getfiled	виклик діалогового вікна пошуку файлу
getint	уведення цілого числа
getkword	уведення опції зі списку
getorient	уведення кута
getpoint	уведення точки
getreal	уведення дійсного числа
getstring	уведення рядка
getvar	значення системної змінної
graphscr	перехід у графічний екран
grclear	очищення видового екрана
grdraw	відображення вектора
grread	читання даних через довільний пристрій уведення
grtext	запис у пункт меню або зону графічного екрана
grvecs	малювання множини векторів
handent	ім'я примітива, що відповідає мітці
help	виклик розділу довідки
if	умовний оператор
initdia	керування діалоговим вікном наступної команди
initget	установка опцій функцій уведення
inters	перетинання відрізків
itoa	перетворення цілого числа в символьну форму
lambda	опис локальної функції
last	визначення останнього елемента списку
layoutlist	перелік листів файлу малюнка
length	довжина списку
list	формування списку
listp	перевірка на список
load	завантаження файлу з LISP-програмами
load_dialog	завантаження DCL-файлу
log	натуральний логарифм
logand	побітове логічне "і"
logior	побітове логічне "або"
Ish	побітовий зсув
mapcar	застосування функції до кожного елемента списку
max	максимум
mem	параметри пам'яті
member	перевірка на входження в список
menucmd	операція з пунктом меню
menugroup	операція із групою меню

min	мінімум
minusp	перевірка на відємність
mode_tile	перемикання режиму поля діалогового вікна
namedobjdict	ім'я основного словника неграфічних даних
nentsel	доступ до даних складного об'єкта із запитом
nentselp	доступ до даних складного об'єкта без запиту
new_dialog	виклик діалогового вікна
not	логічне заперечення
nth	вибір елемента списку по номеру
null	перевірка на nil
numberp	перевірка на число
open	відкриття файлу
or	логічне "або"
osnap	зміна режиму об'єктної прив'язки
polar	побудова точки по відстані й напрямку, що задається кутом
prini	вивід у файл або на екран
princ	вивід у файл або на екран
print	вивід у файл або на екран
progn	програмна дужка для функції if
prompt	вивід повідомлення
quit	вихід
quote	цитування (звертання до списку не як до функції)
read	читання з рядка
read-char	читання символу із клавіатури
read-line	читання рядка з файлу
redraw	перемальовування примітивів
regapp	реєстрація додатка
rem	залишок від ділення
repeat	цикл із заданою кількістю повторень
reverse	перевернення списку
rt05	перетворення дійсного числа в символьну форму
set	присвоєння значення через адресу символу
set_tile	задання значення змінної діалогового вікна
setcfg	запис даних додатку
setenv	завдання змінної оточення
setfunhelp	реєстрація довідкової команди
setq	присвоєння значення символу AutoLISP
setvar	присвоєння значення системній змінній AutoCAD
setview	установка виду у видовому екрані
sin	синус
slide_image	заповнення графічної кнопки слайдом
snvalid	перевірка імені таблиці символів
sqrt	квадратний корінь
ssadd	додавання об'єкта в набір вибору
ssdel	видалення об'єкта з набору
ssget	формування набору по запиту або ознаці
ssgetfirst	визначення обраних об'єктів
sslength	кількість елементів у наборі

ssmemb	перевірка на приналежність об'єкта до набору
ssname	визначення об'єкта з набору
ssnamex	інформація про спосіб створення набору
sssetfirst	включення ручок в об'єктах
startapp	запуск додатка
start_dialog	запуск діалогового вікна завантаженого DCL-файлу
start_image	початок операції із графічною кнопкою
start_list	початок операції над списком, що розкривається
strcase	перетворення регістра в рядку символів
strcat	конкатенація (зчеплення) рядків
strlen	довжина рядка
subst	заміна елемента в списку
substr	визначення підстроки
tablet	робота із планшетом
tbinext	перехід до наступного символу в таблиці
tbiobjname	пошук об'єкта в таблиці символів
tbisearch	пошук символу в таблиці
term_dialog	примусове завершення (закриття) діалогового вікна
terpri	перехід курсору на новий рядок
textbox	обчислення обмежуючого прямокутника для напису
textpage	перемикання в режим текстового екрану
textscr	перехід у текстове вікно
trace	початок трасування
trans	перетворення точки в іншу систему координат
type	визначення типу символу
unload_dialog	вивантаження DCL-файлу
untrace	закінчення трасування
vector_image	малювання відрізка усередині графічної кнопки
ver	визначення версії AutoLISP
vl-acad-defun	перевизначення функції як зовнішнього додатка
vl-acad-undefun	скасування перевизначення функції як зовнішнього додатка
vl-arx-import	імпорт додатка в простір імен іншого документа
vl-bb-ref	повернення значення змінної із недокументного простору імен
vl-bb-set	завдання значення змінної із недокументного простору імен
vl-catch-all-apply	виконання функції з кожним елементом списку
vl-catch-all-error-message	повернення повідомлення про помилку
vl-catch-all-error-p	перевірка, чи є аргумент помилковим
vl-cmdf	передача команди або опції в командний рядок
vl-consp	перевірка списку на nil
vl-directory-files	визначення списку імен файлів папки
vl-doc-export	експорт функції із простору імен VLX-додатка в поточний документ
vl-doc-import	імпорт останньої експортованої функції в простір імен VLX-додатка
vl-doc-ref	передача значення змінної із простору імен поточного документа
vl-doc-set	завдання значення змінної із простору імен поточного документа
vl-every	перевірка істинності умови з першим елементом кожного зі списків-аргументів

vl-exit-with-error	передача керування програмі *еггог*
vl-exit-with-value	повернення значення функції, що викликана іншим простором імен
vl-file-copy	копіювання або приєднання вмісту одного файлу до іншому
vl-file-delete	видалення файлу
vl-file-directory-p	перевірка, чи містить ім'я файлу шлях
vl-file-rename	перейменування файлу
vl-file-size	розмір файлу в байтах
vl-file-systime	дата останньої зміни файлу
vi-filename-base	виділення імені файлу без шляху й розширення
vl-filename-directory	виділення шляху, якщо він входить в ім'я файлу
vl-filename-extension	виділення розширення з імені файлу
vl-filename-mktemp	унікальне ім'я для тимчасового файлу
vl-get-resource	визначення вмісту файлу з розширенням txt, включеного в VLX-додаток
vl-list*	побудова списку
vl-list->string	побудова рядка по кодах буквено-цифрових знаків
vl-list-exported-functions	список експортованих функцій
vl-list-length	довжина списку
vl-list-loaded-vix	список всіх пов'язаних з даним документом VLX-додатків із власними просторами імен
vl-load-all	завантаження файлу в усі раніше відкриті документи й в усі документи, які будуть відкриті в даному сеансі AutoCAD
vl-load-com	завантаження додаткових функцій Visual LISP в AutoLISP
vl-load-react ors	завантаження функцій, що підтримують реактори
vl-member-if	перевірка істинності умови хоча б з одним елементом списку
vl-member-if-not	перевірка на nil умови хоча б з одним елементом списку
vl-position	номер символу як елемента даного списку
vl-prinl-to-string	вивід даних у рядок аналогічно функції prin1
vl-princ-to-string	вивід даних у рядок аналогічно функції princ
vl-propagate	передача значення змінної в усі раніше відкриті документи й в усі документи, які будуть відкриті в даному сеансі AutoCAD
vl-registry-delete	видалення значення з реєстру Windows
vl-registry-descendents	список ключів із зазначеного розділу реєстру Windows
vl-registry-read	читання даних з розділу реєстру Windows
vl-registry-write	створення розділу в реєстрі Windows
vl-remove	видалення елемента зі списку
vl-remove-if	виділення елементів списку, не задовольняють умову перевірки
vl-remove-if-not	виділення елементів списку, що задовольняють умову перевірки
vl-some	перевірка виконання умови, для елементів списків-аргументів
vl-sort	сортування елементів списку
vl-sort-i	сортування елементів списку з поверненням номерів у новому списку
vl-string->list	перетворення рядка буквено-цифрових знаків у список з кодами знаків
vl-string-elt	визначення коду по діючій таблиці для символу з номером у рядку
vl-string-left-trim	видалення символів з початку рядка
vl-string-mismatch	обчислення довжини загального префікса для двох рядків, починаючи з деякої позиції

vl-string-position	пошук символу із заданим кодом у рядку
vl-string-right-trim	видалення символів з кінця рядка
vl-string-search	пошук заданого зразка усередині рядка
vl-string-subst	заміна усередині рядка одного шаблона на інший
vl-string-translate	заміна символів у рядку за законом
vl-string-trim	видалення символів з початку й кінця рядка
vl-symbol-name	повернення імені символу як рядка
vl-symbol-value	повернення значення символу
vl-symbolp	перевірка, чи є об'єкт символом
vl-unload-vix	вивантаження VLX-додатка, завантаженого у простір користувача
vl-vbaload	завантаження проекту VBA
vl-vbarun	виконання VBA-макро
vl-vlx-loaded-p	перевірка, чи завантажена VLX-додаток
vlax-3D-point	створення точки-варіанта
vlax-add-cmd	додавання команди AutoCAD
vlax-create-object	створення VLA-об'єкта
vlax-curve-getArea	площа VLA-об’єкта
vlax-curve-getDistAtParam	довжина кривої від початкової точки до точки, заданої параметром
vlax-curve-getDistAtPoint	довжина кривої від початкової до зазначеної точки
vlax-curve-getEndParam	значення параметра кінцевої точки кривої
vlax-curve-getEndPoint	обчислення кінцевої точки кривої
vlax-curve-getParamAtDist	значення параметра на заданій відстані від початкової точки кривої
vlax-curve-getParamAtPoint	значення параметра в точці кривої
vlax-curve-getPointAtDist	обчислення точки, що перебуває на заданій відстані від початкової точки кривої
vlax-curve-getPointAtParam	обчислення точки, що лежить на кривій, із заданим значенням параметра
vlax-curve-getStart Pa ram	початкове значення параметра на кривій
vlax-curve-getStartPoint	обчислення початкової точки кривої
vlax-curve-isClosed	перевірка замкнутості кривої
vlax-curve- isPeriodic	перевірка періодичності кривої як функції параметра
vlax-curve-isPlanar	перевірка планарності кривої
vlax-curve-getClosestPointTo	обчислення точки, що лежить на кривій і найближчої до заданої точки
vlax-curve-getClosestPointoProjection	обчислення найближчої точки на кривої після проектування кривої на площину
vlax-curve-getFirstDeriv	обчислення першої похідної в заданому місці кривої
vlax-curve-getSecondDeriv	обчислення другої похідної в заданому місці кривої
vlax-dump-obj ect	вивід властивостей об'єкта, а також методів, до нього застосованих
vlax-ename->vla-object	перетворення примітива в VLA-об'єкт
vlax-erased-p	перевірка, чи вилучена VLA-об'єкт
vlax-for	обчислення вираження з кожним VLA-об'єктом із групи (collection)
vlax-get-acad-object	відновлення об'єкта lAcadApplication верхнього рівня
vlax-get-object	обчислення поточного входження об'єкта додатка

vlax-get-or-create-object	обчислення поточного входження об'єкта додатка або створення нового
vlax-get-property	визначення властивості VLA-об'єкт
vlax-import-type-library	імпорт інформації з бібліотеки типів
vlax-invoke-method	виклик методу Active
vlax-ldata-delete	видалення LISP-даних зі словника
vlax-ldata-get	визначення LISP-даних зі словника або об'єкта
vlax-ldata-list	вивід LISP-даних зі словника
vlax-ldata-put	запис LISP-даних у словник або об'єкт
vlax-ldata-test	перевірка можливості збереження LISP-даних
vlax-make-safearray	створення безпечного масиву
vlax-make-variant	створення варіанта
vlax-map-collection	застосування функції до всіх об'єктів групи
vlax-method-applicable-p	перевірка підтримки об'єктом методу
vlax-object-released-p	перевірка, чи звільнена об'єкт у графічній базі
vlax-product-key	шлях до розділу AutoCAD у системному реєстрі Windows
vlax-property-available-p	перевірка наявності в об'єкта необхідної властивості
vlax-put-property	присвоєння властивості VLA-об'єкту
vlax-read-enabled-p	перевірка читаності об'єкта
vlax-release-object	звільнення об'єкта
vlax-remove-cmd	видалення команди або групи команд
vlax-safearray-fill	збереження даних у безпечному масиві
vlax-safear ray-get-dim	визначення розмірності масиву
vlax-safear ray-get-element	читання елемента масиву
vlax-safearray-get-l-bound	нижня границя масиву
vlax-safearray-get-u-bound	верхня границя масиву
vlax-safearray-put-element	запис елемента в масив
vlax-safearray-type	визначення типу масиву
vlax-safearray->list	вивід масиву в список
vlax-tmatrix	представлення матриці перетворення типу 4X4, використовуваної в операціях Active
vlax-typeinfo-available-p	перевірка наявності інформації TypeLib
vlax-variant-change-type	значення варіанта після зміни типу даних
vlax-variant-type	тип даних варіанта
vlax-variant-value	значення варіанта
vlax-vla-object->ename	перетворення VLA-об'єкта в примітив AutoCAD
vlax-write-enabled-p	перевірка можливості зміни об'єкта
vlisp-compile	компіляція LSP-файлу в FAS-файл
vlr-acdb-re actor	створення реактора операцій додавання, зміни й видалення об'єктів
vlr-add	відновлення відключеного реактора
vlr-added-p	перевірка включення реактора
vlr-beep-reaction	видача звукового сигналу
vlr-command-reactor	створення реактора команд
vlr-current-reaction-name	ім'я поточної події, викликаного реактором
vlr-data	отримання даних додатків, пов'язаних з реактором
vlr-data-set	перезапис даних додатків, пов'язаних з реактором
vlr-deepclone-reactor	створення реактора для операцій розмноження об'єктів

vlr-docmanager-reactor	створення реактора малюнків як документів
vlr-dwg-reactor	створення реактора для подій, пов'язаних зі збереженням, відкриттям, закриттям малюнків
vlr-dxf-reactor	створення реактора, пов'язаного із читанням або записом DXF-файлів
vlr-editor-reactor	створення реактора редагування
vlr-insert-reactor	створення реактора операцій вставки блоків
vlr-linker-reactor	створення реактора операцій завантаження/вивантаження ARX-додатків
vlr-lisp-reactor	створення реактора уведення LISP-виражень
vlr-miscellaneous-reactor	створення реактора інших подій
vlr-mouse-reactor	створення реактора операцій з мишею
vlr-notification	перевірка активності реактора залежно від простору імен
vlr-object-reactor	створення реактора об'єктів
vlr-owner-add	додавання об'єкта до списку власників реактора об'єктів
vlr-owner-remove	видалення об'єкта зі списку власників реактора об'єктів
vlr-owners	отримання списку власників реактора об'єктів
vlr-pers	переклад реактора в постійний режим
vlr-pers-list	отримання списку постійних реакторів
vlr-pers-p	перевірка постійності режиму реактора
vlr-pers-re lease	переведення реактора в змінний режим
vlr-reaction-names	список всіх можливих подій для даного типу реакторів
vlr-reaction-set	додавання функцій до реактора
vlr-re act ions	список точкових пар подій і дій
vlr-reactors	список існуючих реакторів
vlr-remove	відключення реактора
vlr-remove-all	відключення реакторів певного типу
vlr-set-notification	перевірка роботи реактора, якщо простір імен неактивний
vlr-sysvar-reactor	створення реактора, пов'язаного зі зміною значень системних змінних
vlr-toolbar-reactor	створення реактора, пов'язаного зі зміною піктограм кнопок панелей інструментів
vlr-trace-reaction	печатка у вікні Трасування (Trace)
vlr-type	тип реактора
vlr-types	список всіх типів реакторів
vlr-undo-react or	створення реактора, пов'язаного зі скасуванням дій
vlr-wblock-reactor	створення реактора, пов'язаного з операцією WBLOCK
vlr-window-reactor	створення реактора, для операцій зміни розмірів вікна AutoCAD
vlr-xref-reactor	створення реактора, пов'язаного з операціями над зовнішніми посиланнями
vports	параметри поточної конфігурації видових екранів
wcmatch	порівняння зі зразком
while	цикл за умовою
write-char	вивід символу
write-iine	вивід рядка
xdroom	визначення розміру вільної пам'яті зони розширених даних
xdsize	визначення розміру пам'яті, для додавання розширених даних
zerop	перевірка на нуль

Додаток 3. Повідомлення про помилки

autocad reject function Аргументи, що передаються для AvtoCAD, були невірними або сама функція є невірною у поточному контексті;
bad argument type Функція передала невірний тип аргументу;
bad association list Список *ASSOC* не складається з підсписків *key value*;
bad function Перший елемент у списку не є ім'ям функції;
bad list Функції передано невірно оформлений список;
bad node Невірний тип елемента, виявлений функцією *TYPE*;
bad node tipe in list Невірний тип елемента, виявлений функцією *FOREACH*;
bad point argument Функції передається невірно визначена точка;
bad point value Функції передається невірно визначена точка;
can't evluate expression Невірне місцезнаходження десяткової крапки або інші невірно відформатовані вирази;
console breack Користувач увів *CTRL-C*, коли оброблювалася функція;
divide by zero Ділення на нуль не дозволяється;
extra right paren Виявлено зайву дужку;
file not open Дескриптор файлу для операції введення-виведення не є дескриптором відкритого файлу;
function canceled Користувач увів *CTRL-C* у відповідь на запрошення введення;
function undefined for argument Аргумент, що передається до *LOG* або *SQRT* виходить за межі допустимих значень;
function undefined for real Дійсне число було передане як аргумент тієї функції, для котрої було потрібне ціле число;
improper argument Аргумент для *GCD* є від’ємним або дорівнює нулю;
incorrect number of argument to a function Кількість переданих аргументів не відповідає кількості формальних аргументів, заданих у *DEFUN*;
insufficient node space Функція не вміщується в області динамічної пам'яті;
insufficient string space Текст не вміщується в області динамічної пам'яті;
invalid argument Невірний тип аргументу;
invalid character У виразі розміщується невірний символ;
invalid dotted pare Точкові пари є списками з двома елементами, відокремленими конструкцією "пробіл – крапка – пробіл";
LISPSTACK overflow Перевищено простір стекової області пам'яті;
misplased dot Дійсне число починається з десяткової крапки;
null function Була спроба вказати на функцію, яка має нульове визначення;
too few arguments Для функції було передано мало аргументів;
too meny arguments Для функції було передано багато аргументів.

Додаток 4. Структура графічної бази даних

Код Значення
0 ARC (Дуга)
10 Центральна точка
40 Радіус
50 Почачковий кут
51 Кінцевий кут

0 CIRCLE (Коло)
10 Центр
40 Радіус

0 POLYLINE (Полілінія)
10 Початкова точка
40 Початкова ширина
41 Кінцева ширина

66 Атрибути (напр. вертекси)
70 Флаги полілінії:
1 - Замкнута відрізком,
2 - Замкнута кривою

0 LINE (Відрізок)
10 Початкова точка
11 Кінцева точка

0 VERTEX(Вершина полилиниї)
10 Координати вершини
40 Початкова ширина
41 Кінцева ширина
42 Кривизна(якщо зглажувалась)
50 Дотична (якщо зглажувалась)

70 Флаги вертексів:
1 - Додаткова вершина,ство-
рена при зглажуванні,
2 - Відзначена дотичною

Код Значення
0 DIMENSION (Розмір)
1 Розмірний текст
2 Им`я системного блока
10 Початкова точка
11 Центральна точка тексту
12 Точка продовження
13 Перша точка відрізка або кута
14 Друга точка відрізка або кута
15 Діаметр,радіус або точка кута
16 Точка дуги кутового розміру
40 Довж ина зноски
50 Кут (в радіанах)
70 Тип розміру:
0-Горизонтальний
(Вертикальний)
1 - Зрівняний, 2 - Кутовий,
3 - Діаметр,4 – Радіус

0 TEXT (Текст)
1 Текстова строка
10 Початкова точка
11 Точка зрівняння (якщо є)
40 Висота тексту
41 Коефіцієнт розтягнення тексту
50 Кут обертання
51 Флаги атрибута тексту:
1 - Прихований,2 - Постійний,
3 – Контролюємий
71 Флаги генерації тексту:
1- Дзеркальний,2-
Перевернутий
72 Код виравнювання тексту:
0 - Ліве;1 - Центр;2 - Праве;
3 - Зрівняне;4 - "M" Середина;
5 - "F" Вписаний

Наприклад:

накреслений відрізок

((-1 . <Entity name: 60000018>)

(0 . "LINE") ; тип примітиву
(8 . "0") ; шар
(10 2.0 2.0) ; початкова точка
(11 8.0 6.0)) ; кінцева точка
Накреслене коло
((-1 . <Entity name: 600000a8>)
(0 . "CIRCLE") ; тип примітиву
(8 . "0") ; шар
(10 5.0 5.0) ; центр
(40 . 3.0)) ; радіус
Написаний текстовий примітив
((-1 . <Entity name: 60000018>)
(0 . "TEXT") ; тип примітиву
(8 . "pr") ; шар
(10 1.0 3.0) ; початкова точка
(40 . 2.0) ; висота тексту
(1 . "Select object") ; текст
(50 . 0.0) ; кут повороту
41 . 1.0) ; коефіцієнт розтягнення тексту
(51 . 0.0) ; кут нахилу
(7 . "STANDARD") ; тип шрифту
(71 . 0) ; прапорець генерації тексту
(72 . 0) ; вирівювання тексту
(11 0.0 0.0)) ; точка вирівнювання

ПОКАЖЧИК ВИЗНАЧЕНЬ ФУНКЦІЙ