МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

ВОЛЖСКИЙ ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ (ФИЛИАЛ)

ФЕДЕРАЛЬНОГО ГОСУДАРСТВЕННОГО БЮДЖЕТНОГО ОБРАЗОВАТЕЛЬНОГО

УЧРЕЖДЕНИЯ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«ВОЛГОГРАДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

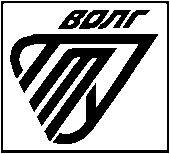
КАФЕДРА «ИНФОРМАТИКА И ТЕХНОЛОГИЯ ПРОГРАММИРОВАНИЯ»

О. Ф. Абрамова

**Применение языка функционального программирования AutoLISP к решению задачи автоматизированного построения типизированных чертежей**

**методические указания к контрольной работе**

Методические указания



Волгоград

2016

УДК 004

Рецензент:

канд. технич. наук доцент *О. В. Свиридова*

Издается по решению редакционно-издательского совета

Волгоградского государственного технического университета

Абрамова О. Ф. **Методические указания по выполнению контрольной работы по дисциплине «Программирование» [**Электронный ресурс**]**: методические указания / О. Ф. Абрамова// Сборник «Методические указания» Выпуск .-Электрон. текстовые дан.(1файл:141Kb) – Волжский:ВПИ(филиал)ВолгГТУ,2016.-Систем.требования:Windows 95 и выше; ПК с процессором 486+; CD-ROM.

Методические указания содержат детальную информацию о структуре и содержании контрольной работы по дисциплине «Программирование», а так же необходимые требования, согласно ГОСТ, на оформление пояснительной записки.

Предназначены для студентов бакалавриата, обучающихся по направлению подготовки 09.03.01 «Программная инженерия». CD-ROM

©Волгоградский

государственный технический

университет, 2016© Волжский

политехнический институт, 2016

1. ВВЕДЕНИЕ

Реализация контрольной работы в рамках изучения дисциплины «Программирование» обеспечивает в полной мере усвоение теоретической информации курса, а так же приобретение реальных практических навыков в области разработки программных продуктов в системе автоматизированного проектирования AutoCAD фирмы AutoDesk. Одной из основных функций программного инженера является проектирование объектов того или иного назначения или технологических процессов их изготовления. Задачи, возникающие перед современными техническими науками, вызывают необходимость создания все более сложных технологических объектов в сжатые сроки и с более высоким качеством. Для обеспечения эффективности выполнения таких задач создаются и используются *системы автоматизированного проектирования* (САПР).

САПР AutoCAD - это один из наиболее распространенных в нашей стране графических редакторов, при помощи которого конструктор создает чертежи проектируемого изделия. Для эффективной работы AutoCAD следует адаптировать под автоматизированное проектирование изделий определенного класса. Сделать это можно с помощью программирования каких-то типовых алгоритмов построения параметрических чертежей различных изделий. Тогда работа конструктора будет сведена к вводу параметров изделия, а всю черновую работу по отрисовке чертежа возьмет на себя программа, написанная на встроенном в AutoCAD языке программирования.

В качестве встроенного языка программирования в САПР AutoCAD была принята модификация функционального языка программирования LISP – AutoLIS, которая позволяет пользователям и разработчикам AutoCAD писать макропрограммы и функции на очень мощном языке высокого уровня, хорошо сочетаемом с прикладной графикой. LISP прост в изучении и очень гибок, обладает рядом значительных преимуществ.

Выполнение контрольной работы заключается в проектировании и программной реализации автоматизированного построения параметрического чертежа детали средней сложности согласно варианту задания.

Структура контрольной работы формируется в полном соответствии с основными разделами дисциплины «Программирование» и включает:

* исследование предметной области и анализ объекта программирования;
* полный цикл проектирования и разработки программы для автоматизированного построения параметрического чертежа с разработкой сопутствующей документации в соответствии со стандартами в области проектирования и разработки ПО;
* демонстрацию предложенного решения в форме видеоролика (скринкаста).

Результатами контрольной работы являются программная реализация автоматизированного построения параметрического чертежа, пояснительная записка, оформленная в соответствии с требованиями стандартов и задания на контрольную работу, видеоролик (скринкаст), демонстрирующий работу программы.

2. ОБЩИЕ ТРЕБОВАНИЯ К КОНТРОЛЬНОЙ РАБОТЕ

Общая тематика контрольной работы по дисциплине «Программирование»: ***Применение языка функционального программирования AutoLisp к решению задачи автоматизированного построения типизированных чертежей.***

Контрольные работы по дисциплине «Программирование» имеют однотипную структуру и различаются объектом реализации, в роли которого могут выступать чертежи нескольких видов некоторой детали.

Последовательность выполнения контрольной работы

Преподаватель выдает задание студенту до 15 марта. Контрольная работа выполняется в течение всего семестра до 25 мая.

Промежуточный контроль и решение всех возникающих у студентов вопросов производится преподавателем во время часов ОргСРС. Ответственность за выбор того или иного решения, правильность расчетов, оформление работы несет студент. Руководитель предостерегает его от ошибочных решений и характеризует достоинства и недостатки различных вариантов решений, при этом право окончательного выбора предоставляется студенту.

График защиты контрольных работ составляется преподавателем и доводится до сведения студентов. Защита контрольных работ осуществляется по графику после проверки ее работоспособности преподавателем.

Отчет выполненной и оформленной контрольной работы будет проходить в следующем порядке:

1. демонстрация работы реализованной программы построения параметрического чертежа;
2. предоставление преподавателю для ознакомления оформленной, в соответствии с требованиями, контрольной работы;
3. устный отчет по разработанной программе и вопросам преподавателя на темы, затронутые в представленной контрольной работе.

Структура контрольной работы

Пояснительная записка к контрольной работе по дисциплине «Программирование» должна в обязательном порядке содержать следующие разделы:

1. Теоретическая часть;

2. Практическая часть:

* постановка задачи;
* описание предлагаемого решения:
  1. перечень используемых дополнительно программных продуктов (графические редакторы, редакторы для построения диаграмм и т.п.)
  2. перечень реализованных функций в формате:
     + заголовок функции
     + входные и выходные параметры
     + общее описание работы функции
  3. перечень функций (алгоритмов, методов), используемых в программе сверх установленных требований к реализации
  4. подробное описание предлагаемых авторских решений
* общий алгоритм (диаграмма, блок-схема) работы программы;
* алгоритмы ключевых функций (классов, методов) программы
* распечатка программного кода с комментариями;
* скриншоты результатов работы разработанной программы.

Помимо текстового материала контрольная работа должна быть оформлена в электронном виде.

***Электронная версия отчета:***

* + Отчет в формате MsWord;
  + Программная реализация
  + Видеоролик, демонстрирующий результаты работы программы

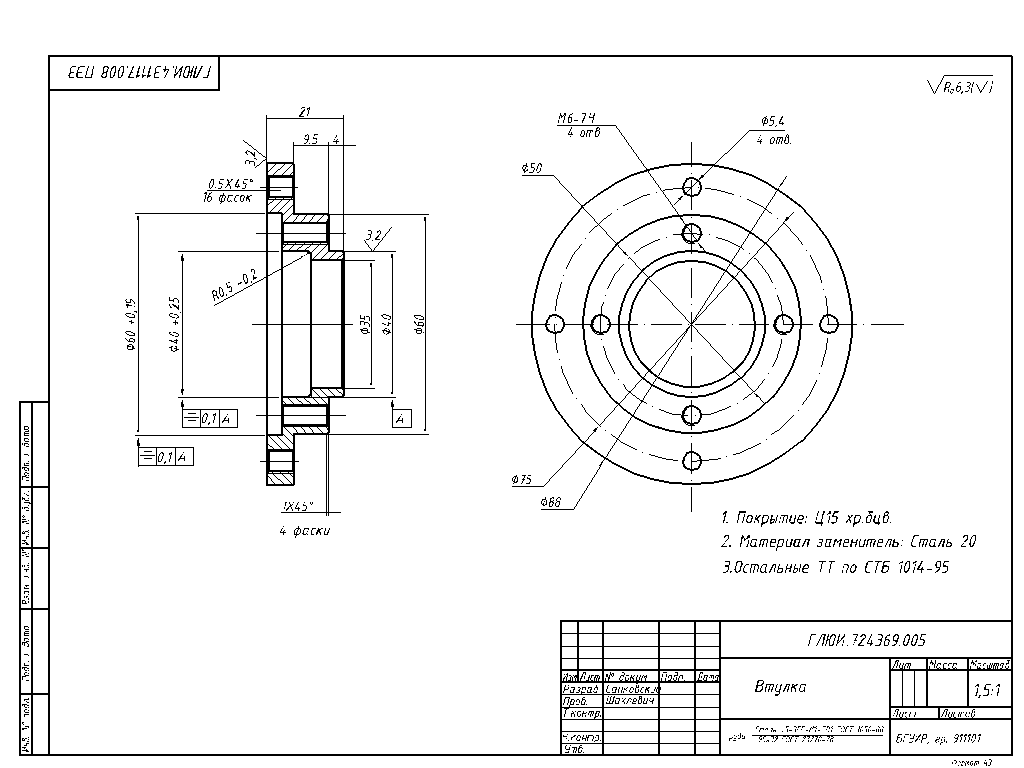
1. ВАРИАНТЫ КОНТРОЛЬНОЙ РАБОТЫ ПО ДИСЦИПЛИНЕ «ПРОГРАММИРОВАНИЕ»

## Варианты теоретической части

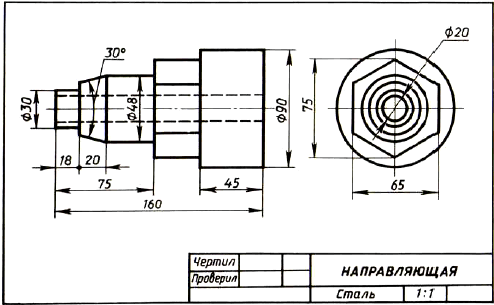
1. Обзор языков логического программирования.
2. Язык логического программирования Пролог.
3. Constraint-технология в логическом программировании.
4. Основные понятия логического программирования (основные конструкции, операционная и декларативная семантика, интерпретация, корректность).
5. Основные элементы языка Пролог.
6. Термы**.** Анализ структуры термов.
7. Программирование баз данных. Этапы проектирования.
8. Предикаты: определение, виды, способы применения.
9. Недетерминированное программирование: основные понятия и возможности, области применения.
10. Планирование как точная наука, или constraint-технологии на практике.
11. Применение логического программирования в задачах искусственного интеллекта.
12. Технический аспект создания искусственного интеллекта.
13. Различные подходы к созданию искусственного интеллекта.
14. Методы хранения и обработки нечетких данных.
15. Анализ подходов к обработке нечетких данных в аналитических задачах поддержки принятия решений.
16. Нечетко-логические модели и алгоритмы.
17. Нечеткие запросы к реляционным базам данных.
18. Неполные структуры данных в Пролог-программе.

## Варианты практической части:

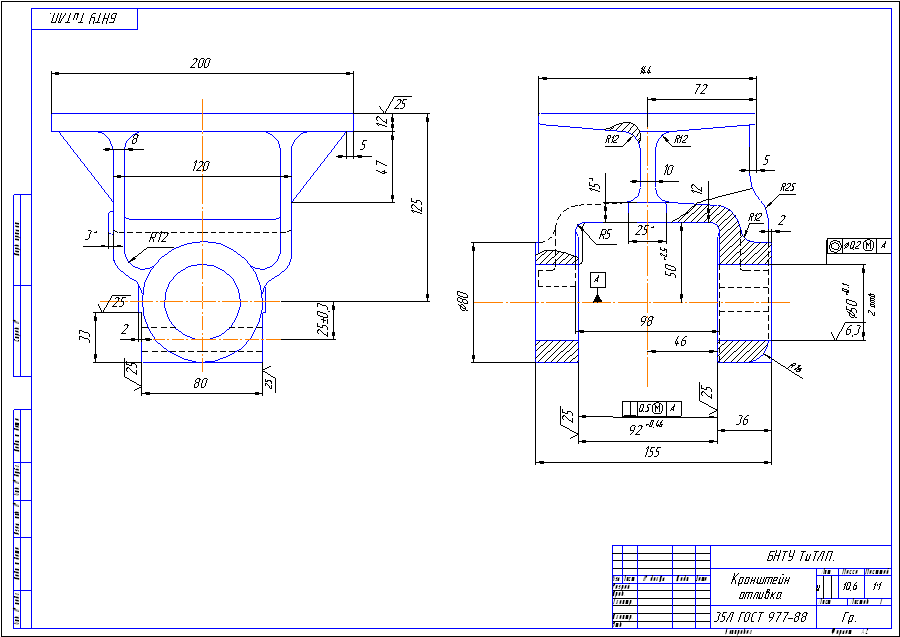
Вариант 1



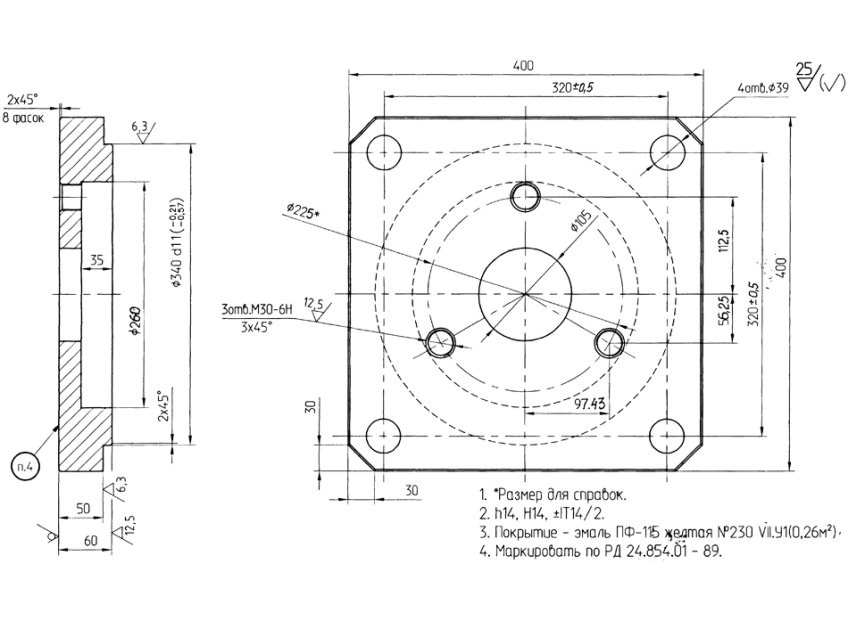
Вариант 2



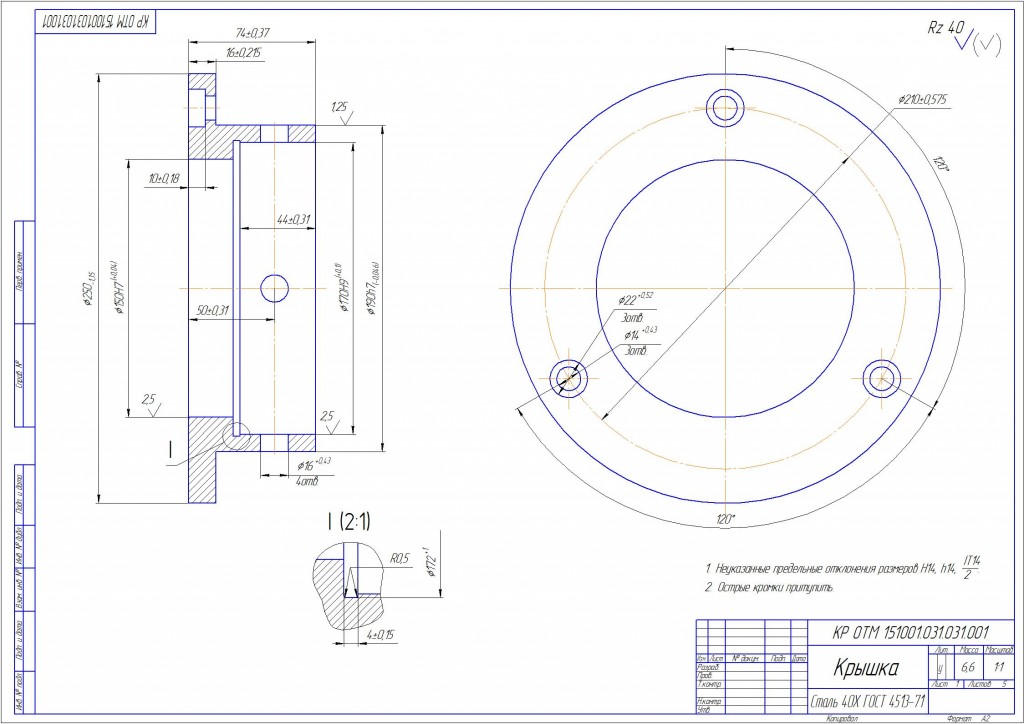
Вариант 3



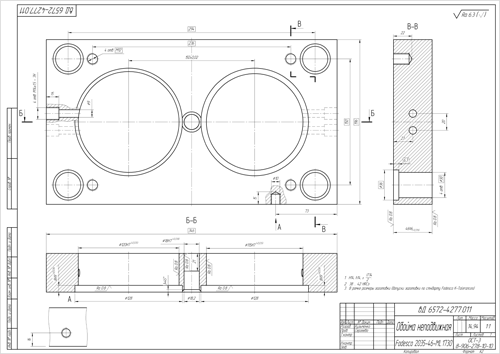
Вариант 4



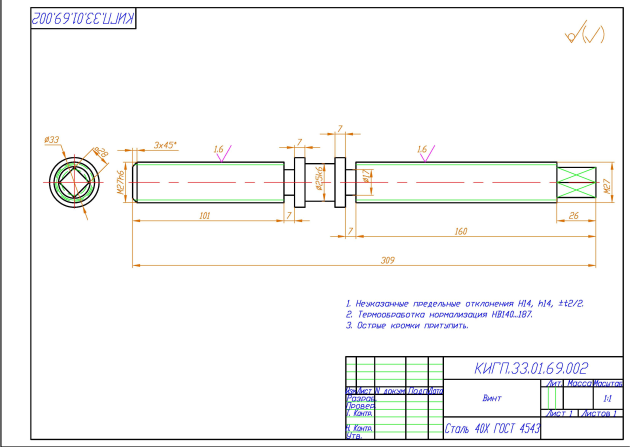
Вариант 5



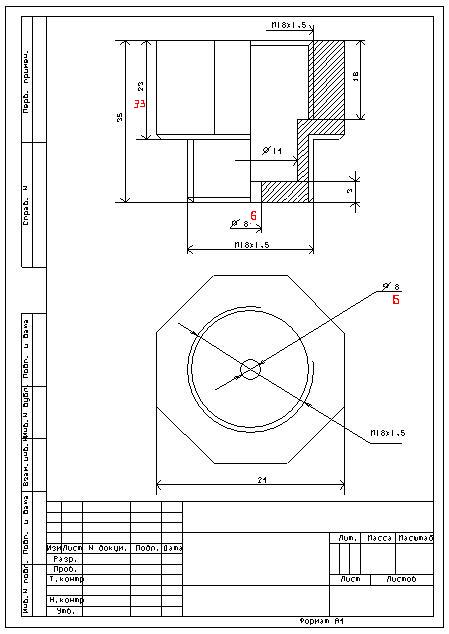
Вариант 6



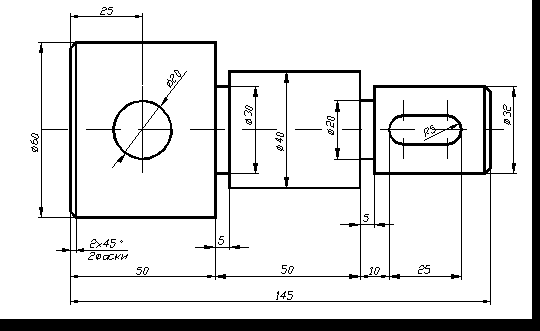
Вариант 7



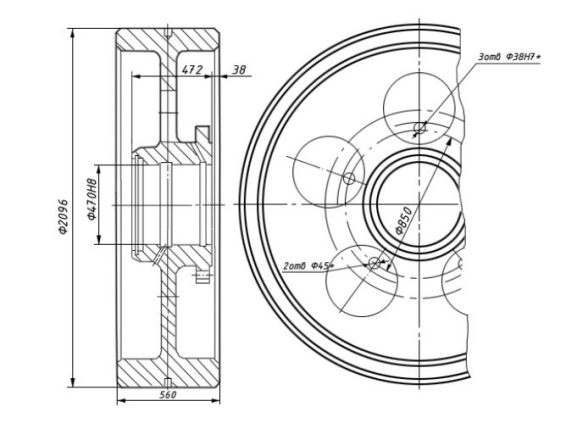
Вариант 8



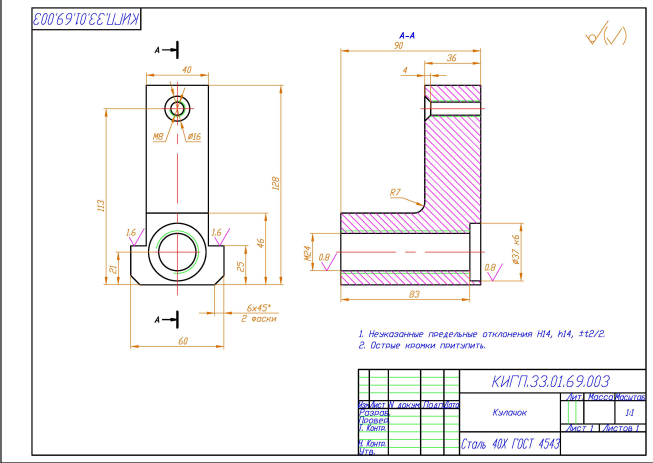
Вариант 9



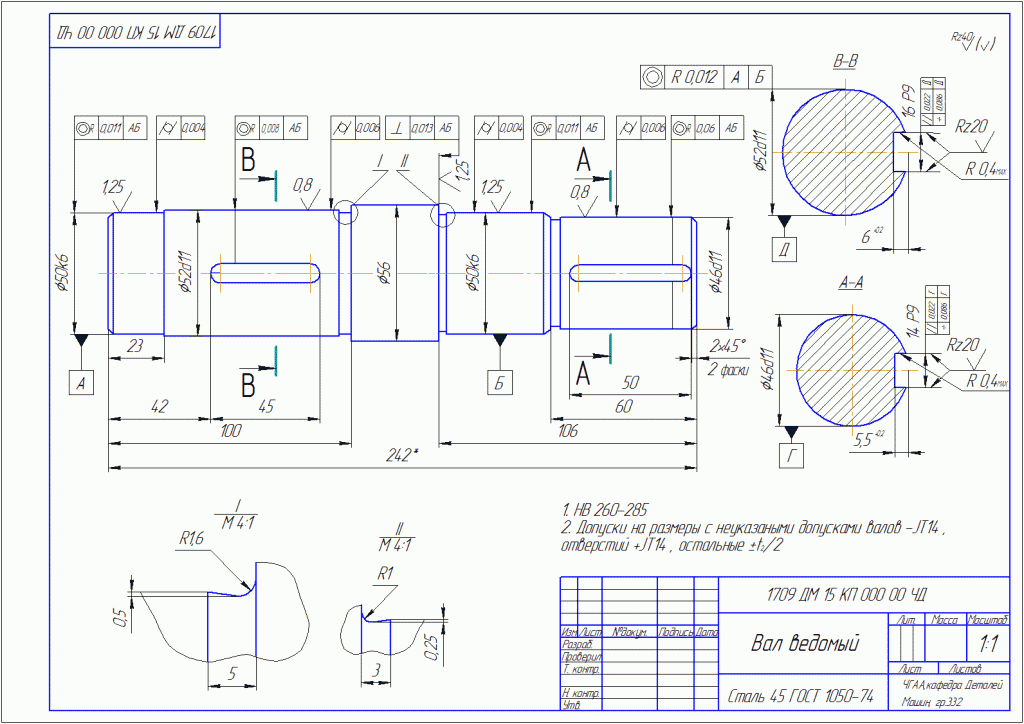
Вариант 10



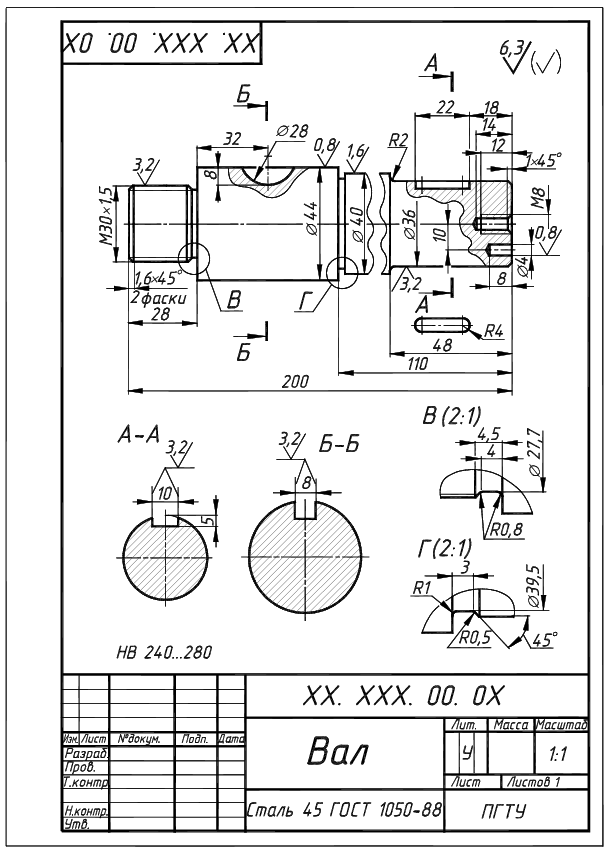
Вариант 11



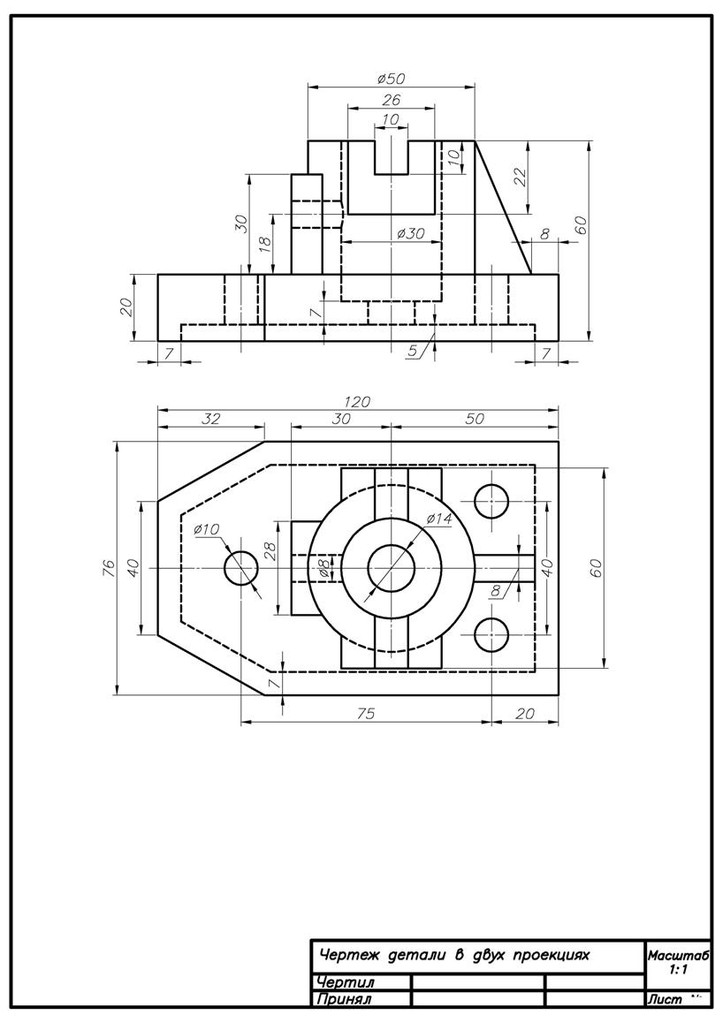
Вариант 12



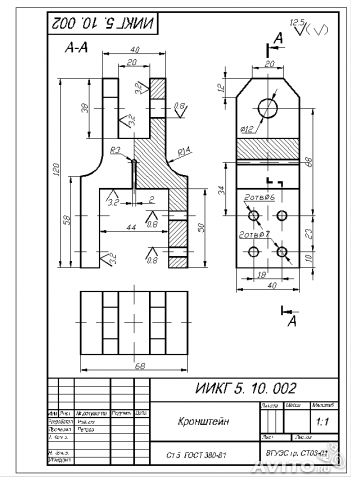
Вариант 13



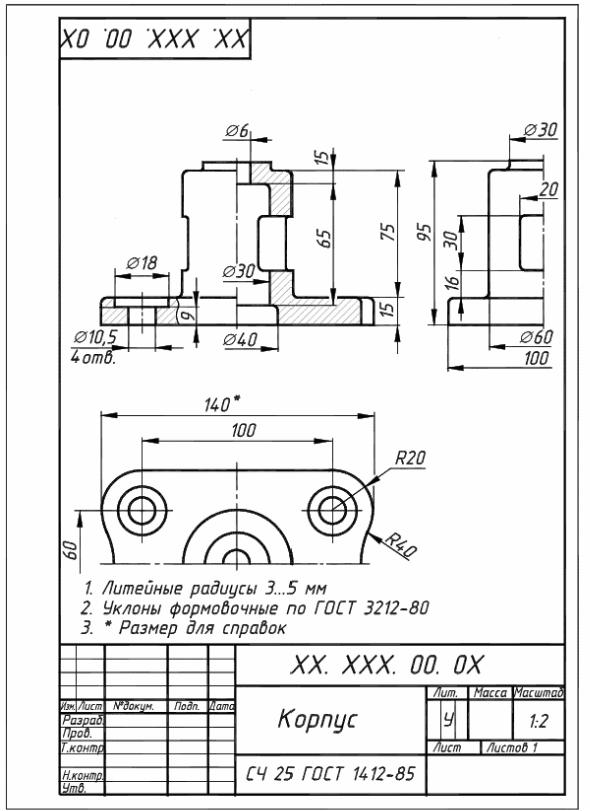
Вариант 14



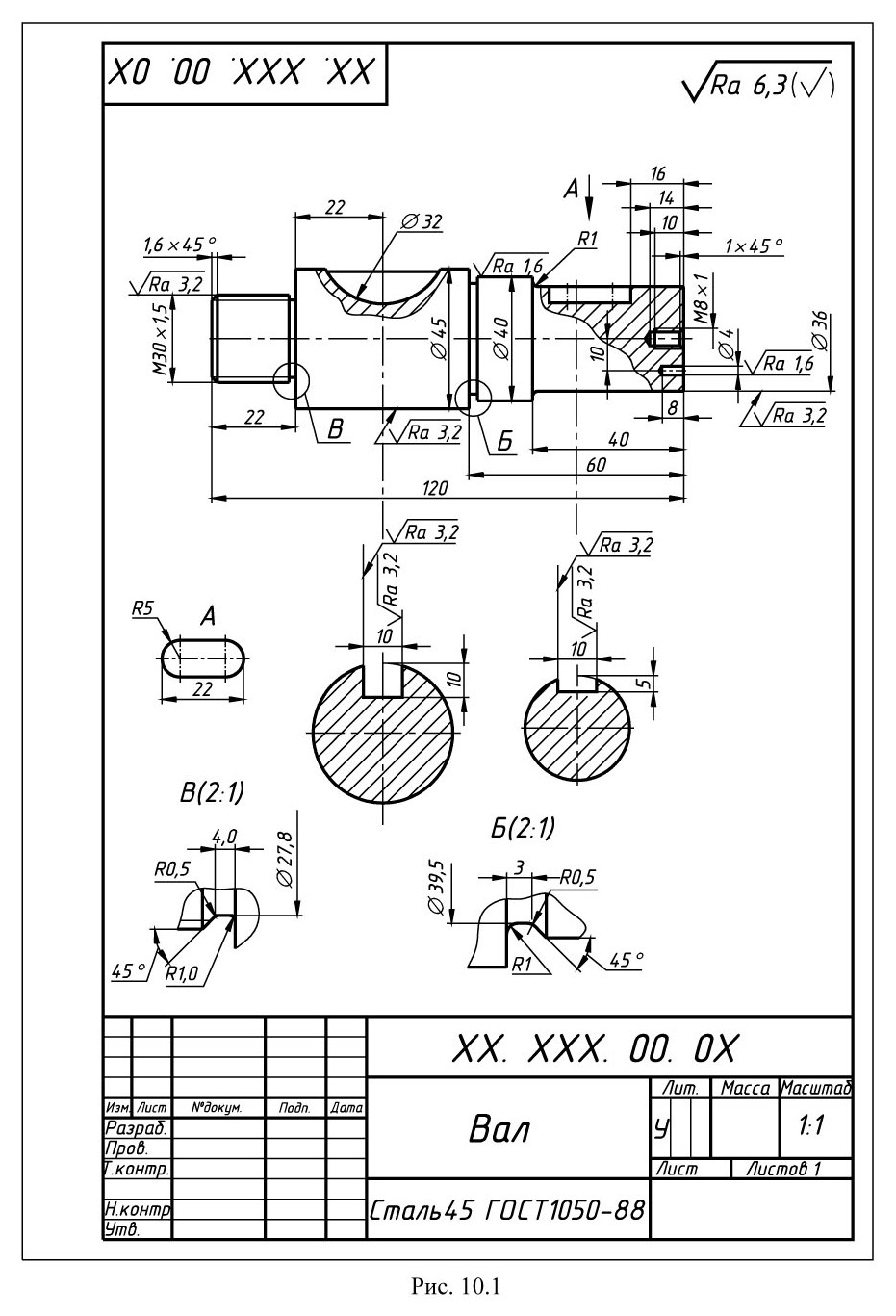
Вариант 15



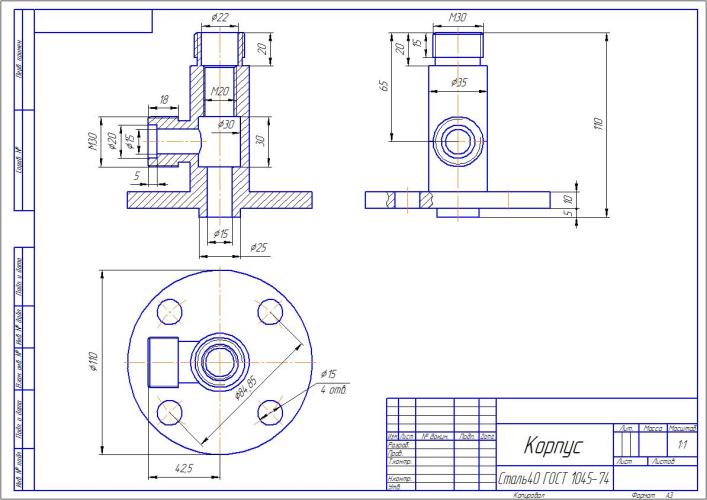
Вариант 16



Вариант 17



Вариант 18



# ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Одной из основных функций инженера является проектирование объектов того или иного назначения или технологических процессов их изготовления. Задачи, возникающие перед современными техническими науками, вызывают необходимость создания все более сложных технологических объектов в сжатые сроки и с более высоким качеством. для обеспечения эффективности выполнения задач, встающих перед проектировщиком, создаются и используются *системы автоматизированного проектирования* (САПР).

САПР AutoCAD - это один из наиболее распространенных в нашей стране графических редакторов, при помощи которого конструктор создает чертежи проектируемого изделия. Для эффективной работы AutoCAD следует адаптировать под проектирование изделий определенного класса. Сделать это можно с помощью программирования каких-то типовых алгоритмов, используемых в проектировании. Тогда работа конструктора будет сведена к вводу параметров изделия, а всю черновую работу по отрисовке чертежа возьмет на себя программа, написанная на встроенном в AutoCAD языке программирования.

В качестве встроенного языка программирования в САПР AutoCAD была принята модификация языка программирования LISP - AutoLISP который позволяет пользователям и разработчикам AutoCAD писать макропрограммы и функции на очень мощном языке высокого уровня, который хорошо сочетается с прикладной графикой. LISP прост в изучении и очень гибок. К достоинствам языка, повлиявшем на выбор его фирмой AutoDesk в качестве встроенного в AutoCAD, можно отнести следующее:

1. Основное понятие языка - ***список*** - перечень атомов или списков, отделеных друг от друга пробелами и заключенных в круглые скобки. Список - это оптимальный способ представления графической информации.

2. Высокая компактность языка (ядро интерпретатора занимает 4 .. 10 Кб).

3. Легкость реализации и чрезвычайно высокая надежность программ.

4. Функциональный стиль программирования.

# Основные понятия AutoLISP

## Особенности языка AutoLISP

LISP - это язык программирования созданный в конце 50-х гг., для решения задач искусственного интеллекта. Название языка переводится как "обработка списков" (**LIS**t **P**rocessing). На основе синтаксиса и семантики этого языка был разработан язык AutoLISP.

Основным понятием языка AutoLISP, как уже было сказано, является ***список*** - *перечень атомов и списков, отделенных друг от друга пробелами и заключенных в круглые скобки.*

*Атом* - это неделимая часть языка (число, символ, строковая константа). Атом не может быть вложенным, в отличии от списка.

Самое важное, что необходимо запомнить в начале изучения языка AutoLISP, это то, что ***любая информация*** на нем, будь то действие, данные или последовательность действий, ***записывается в круглых скобках через пробел***. То есть в ***AutoLISP нет различия между программой и данными***, и то и другое есть список. Любой список интерпретатор языка сразу пытается выполнить, поэтому при записи данных необходимо явно отключать восприятие списка как руководства к действию. Это делается при помощи специальных встроенных функций, о которых мы с Вами поговорим чуть позже.

LISP, а так же, соответственно и AutoLISP, относятся к языкам функционального стиля программирования. Основная идея этого стиля заключается в том, что любое действие представляется в виде некоторой функции, ***на входе*** у которой может быть неограниченное количество аргументов (вектор), а ***на выходе всегда*** *есть один результат -* ***скаляр***. В общем виде такую запись можно представить формулой: . Где P (входной вектор аргументов) может быть и пустым множеством, а вот ***результат у функции*** должен быть обязательно и ***должен быть*** ***единственным***.

Реализация этой идеи в языке AutoLISP осуществляется с помощью ***обратной польской нотации*** - записи действий (функций) в определенной форме, общий вид которой

**(<идентификатор\_функции> apr1 apr2 ... aprN)**

*Идентификатор\_функции* - это либо имя стандартной функции AutoLISP, либо имя пользовательской функции, придуманной программистом.

*apr1...aprN* - это ***входные параметры*** (аргументы) функций, количество которых зависит от конкретной функции. Значением аргументов могут быть числа, символы, символьные строки и другие функции. То есть аргументами функции могут быть как атомы, так и списки.

Например, запись простейшего арифметического действия 3 + 4 на языке AutoLISP будет выглядеть следующим образом: (+ 3 4), где "+" - это имя стандартной арифметической функции AutoLISP, а "3" и "4" - значения ее аргументов.

Так как такая запись достаточно сильно отличается от привычных нам записей математических действий, важно понимать следующие ***правила записи любого выражения, действия или данных AutoLISP:***

- любая информация записывается в круглых скобках;

- сразу после открывающейся скобки ВСЕГДА записывается **идентификатор** функции;

- между именем функции и аргументами ВСЕГДА ставится **пробел** (несколько пробелов воспринимаются интерпретатором языка как один), являющийся разделителем элементов списка;

- между последним аргументом и закрывающейся скобкой списка пробел не нужен.

Рассмотрим теперь более сложный пример. Попытаемся записать выражение:

Сложение и вычитание обозначаются в AutoLISP привычными нам знаками этих операций, деление - знаком слэш ( / ), а степень вычисляет стандартная функция expt. Но, что бы не отвлекаться пока на объяснение ее формы записи ( список математических функций будет приведен позднее), квадрат в выражении мы запишем просто перемножив скобку два раза. Тогда общая польская запись предложенного выражения (после недолгих размышлений) будет выглядеть так:

( / (\* (+ 3 4) (+ 3 4)) ( - (+ 4 5 6) (- 35 21))).

Теперь, обобщив все сказанное ранее, мы можем определить ***основные особенности языка AutoLISP:***

1) в AutoLISP нет различия между программами и данными;

2) любая информация записывается в виде списка;

3) действия в AutoLISP - это некая функция (функциональный стиль программирования);

4) запись функций осуществляется с помощью обратной польской нотации.

## Типы данных в AutoLISP

AutoLISP поддерживает несколько различных типов данных:

- списки;

- символы;

-строковые константы - строковые константы могут быть любой длинны, память для них распределяется динамически. Однако существует максимальная длинна для строковых констант, равная 100 символам;

- действительные числа - представлены как числа с плавающей запятой двойной точности;

- целые числа - целые 16-битовые числа в пределах от -32768 до +32767;

- файловый дескриптор;

- "имена" примитивов AutoCAD;

- наборы AutoCAD;

- встроенные функции.

AutoLISP содержит несколько встроенных функций для обеспечения основ программирования 2-х мерных и 3-х мерных графических программ. Когда вы имеете дело с координатами, соблюдаются следующие соглашения:

* 2-х мерные точки выражаются в виде списков из двух действительных чисел (X Y), как: (3.400000 7.520000). Первое значение - координата X, второе - Y.
* 3-х мерные точки выражаются в виде списков из трех действительных чисел (X Y Z), как: (3.400000 7.520000 1.000000). Первое координата X, второе - Y, третье - Z.

При любом запросе AutoCAD некоторого значения (например, точки или масштабного фактора) для достижения необходимого результата могут быть использованы выражения AutoLISP этого типа или функции AutoLISP, возвращающие результат этого типа.

**Соглашение по лексике**

Ввод в AutoLISP может принимать несколько форм:

- с клавиатуры в сеансе AutoCADа;

- считываться с ASCII-файла;

- считываться из строковой константы.

Во всех случаях должны соблюдаться следующие соглашения:

* имена (идентификаторы) атомов и функций могут состоять из любой последовательности печатных знаков, за исключением: скобок **( )**, точки, апострофа ( **'** ), двойных ковычек ( **"** ) и точки с запятой ( **;** );
* следующие символы вызывают прерывание идентификатора или числовой константы: **( ) ' " ; (пробел) (конец файла)**;
* **T** и **NIL** - зарезервированы и не должны использоваться в имени атома или функции;
* имена атомов и функций в AutoLISP безразличны к регистру, на котором они набраны;
* идентификаторы не могут начинаться с цифры;
* выражения могут занимать несколько строк;
* несколько пробелов между символами равны одному пробелу. Хотя отступы в строках и не требуются, их можно использовать для повышения наглядности структурирования функций;
* целые константы могут начинаться с необязательных символов "+" или "-";
* действительные константы состоят из одной или более цифры, за которыми следует десятичная точка, за которй, в свою очередь следует одна или более цифр; т.о. "4" не является действительным, правильным будет "0.4". Точно также и "5." не является действительным правильно будет "5.0". Действительные числа могут выражаться в экспоненциальной форме, т.е. иметь необязательный параметр "е" или "Е", за которым следует степень числа 10;
* символьные строковые константы - это последовательность знаков, взятая в кавычки. Внутри взятых в кавычки строковых констант можно ввести управляющие последовательности, пользуясь знаком обратной черты ( \ ).

Возможно использование следующих ***управляющих последовательностей:***

- \\ означает знак \ ;

- \\**е** означает символ escape-последовательности;

- \**n** означает новую строку (newline);

- \**r** означает возврат каретки (return);

- \**t** означает табуляцию (tab);

- **\nnn** означает знак, восьмеричный код которого равен nnn.

Например, следующая запись выводит подсказку на новую строку:

(prompt "\n Введите первую точку: " )

* Знак апострофа может быть использован как короткая запись функции QUOTE. Например: ' ( 10 20 30 ) эквивалентно ( quote 10 20 30 ).
* Текст пользовательской функции, как и значение любой переменной AutoLISP, можно вывести на экран при помощи восклицательного знака ( ! ).
* В AutoLISP-программы могут быть включены комментарии, которые начинаются с символа " ; "и продолжаются до конца строки. Например: ; это строка, состоящая только из комментария ( setq a (\* pi r r ); Вычисление площади круга

**Создание собственных функций**

Как вы помните, все действия на языке AutoLISP записываются с помощью функций. Существует большой выбор встроенных функций AutoLISP, но они, естественно, не могут удовлетворить все потребности программиста (конструктора), задумавшего воплотить чертеж некоторой детали на экране, а затем и на бумаге. Поэтому существует возможность создавать свои собственные функции, выполняющие только вам необходимые действия. Для этого, собственно, автор и составлял это пособие - ознакомить вас с основными правилами синтаксиса и семантики функций, написанной на языке AutoLISP. Начнем с основного: общих правил построения собственной функции на AutoLISP.

***Создание пользовательской функции*** заключается в использование встроенной функции **DEFUN**, общая форма записи которой:

**(DEFUN <name> (**

**(выражение 1)**

**(выражение 2)**

**...**

**(выражение k)**

**)**

где: *name* - имя пользовательской функции;

- список ***входных*** параметров (***аргументов***) функции;

**-** список ***локальных*** переменных функции, т.е. переменных, которые используются и видны только внутри данной функции. В начале работы пользовательской функции локальные переменные получают значения, по окончании работы этой функции локальные переменные удаляются из памяти;

(выражение 1) ... (выражение k) - запись действий, которые должнавыполнить данная пользовательская функция.

**Действие функции DEFUN:** *создает* в памяти пользовательскую функцию с именем name и списком аргументов - . При этом сама функция еще *не выполняется*, а только помещается в память. Пользовательская функция может не иметь локальных переменных и/или аргументов. Но, в любом случае, круглые скобки в заголовке функции DEFUN сохраняются.

Например:

( defun ttt ( d f) ...) - функция ttt имеет 2 аргумента и не имеет локальных переменных;

( defun ttt ( / d f) ...) - функция ttt имеет 2 локальных переменных и не имеет входных параметров;

( defun ttt ( ) ...) - функция ttt ни содержит аргументов, ни локальных переменных.

***Для запуска на выполнение*** пользовательская функция должна быть:

1***) загружена в AutoCAD*** с помощью команды **LOAD**

- либо указанной в командной строке виде:

command> "load файла (без расширения!)>"

- либо через управляющее меню AutoCAD:

Tools / LoadApplication / выбор файла / Load ;

2) ***явно вызвана***, согласно правилам вызова любой функции AutoLISP (напоминаю: в круглых скобках, с указанием имени и фактических значений аргументов).

***При вызове*** некоторой ***пользовательской функции ей передаются необходимые значения*** *(фактические параметры)* ***для всех аргументов,*** указанных при определении этой функции. Пользовательские функции не могут иметь произвольное число фактических значений аргументов. Переданные при вызове функции значения должны совпадать с объявленными при определении функции аргументами: *по типу; по количеству; по порядку следования.*

***Вызов пользовательских функций с неверным числом аргументов приведет к ошибке.***

Пользовательская функция, как и любая другая в AutoLISP, возвращает некоторое значение (результат работы данной функции).

***Возвращаемым значением (результатом)*** всей пользовательской функции ***является результат выполнения последнего выражения***, записанного в функции Defun. Для определения этого значения используются следующие правила:

1) найти предпоследнюю закрывающую скобку;

2) найти парную ей открывающую;

3) значение, возвращаемое функцией, записанной после этой открывающей скобки, и есть результат *всей* пользовательской функции.

*Например*: написать функцию, вычисляющую tg угла.

Для определения tg (x) необходимо знать значение аргумента x, т.е. входной параметр в вашей функции будет один.Для вычисления тангенса некоторого угла x необходимо выполнить следующее действие: tg (x) = sin (x) / cos (x), т.е. для получения нужного результата нам достаточно одного действия, поэтому никаких локальных переменных мы вводить не будем, и сделаем функцию деления результирующей (т.е. возвращающей результат выполнения пользовательской функции).

(Defun tan (a)

( / (sin a) (cos a))

)

Загрузка данного списка приведет к появлению в памяти переменной tan типа "определенная пользователем функция", значением которой является список-программа вычисления tg. Для вызова на выполнение этой функции необходимо в комндной строке AutoCAD в скобках набрать имя функции и фактическое значение ее аргумента, например:

Command> (tan 34)

**Преобразование пользовательской функции в команду AutoCAD**

Если в заголовке пользовательской функции перед именем поставить "С:", то вся функция после загрузки будет восприниматься AutoCAD как собственная команда. Для запуска такой функции необходимо будет только набрать ее имя в командной строке и нажать <enter>, т.е. произвести ее запуск на выполнение так же, как и запуск любой встроенной в AutoCAD команды, и не тратить время на постановку скобок. Но данным приемом можно пользоваться только для функций, *не имеющих аргументов*.

**Область видимости переменных**

Переменные AutoLISP делятся на два типа:

***Глобальные*** - постоянно находятся в оперативной памяти и, следовательно, доступны из любой функции. Глобальные переменные создаются автоматически при присваивании переменной значения.

***Локальные*** - явно описанные в заголовке функции DEFUN и видны только внутри соответствующей пользовательской функции.

Имена локальных и глобальных переменных могут совпадать, т.к. во время работы некоторой функции локальное имя "перекрывает" глобальное. Но, во избежание путаницы, желательно локальным и глобальным переменным давать разные имена.

Аргументы пользовательских функций также являются локальными переменными.

*Использование глобальных переменных не желательно*, так как они занимают память, вызывая необходимость отслеживания имени переменных.

**Данные следует хранить в локальных переменных и передавать через память функций.**

Если глобальные переменные все же используются, то, как только необходимость в них отпадает, следует освобождать занимаемое ими место, присваивая значение NIL.

## Присваивание значений в AutoLISP

Так как значение, полученное от пользователя (т.е. введенное с клавиатуры) либо в результате каких-либо вычислений, не может просто "зависнуть в воздухе" и должно быть где-то сохранено, в AutoLISP существуют специальные функции, осуществляющие указанные действия. Известно, что с данными можно работать как непосредственно, так и по адресу, т.е. используя указатель на области памяти, в которой эти данные были записаны.

Особенности организации памяти в AutoLISP приводят к тому, что следует различать два понятия:

1) ***переменная*** - *указатель* на область динамической памяти, имеющая имя;

2) **значение переменной** - *данные*, записанные в динамической памяти, начиная с адреса, записанного в переменную.

Изменить можно как переменную, так и ее значение. В первом случае та же переменная будет указывать на другой участок динамической памяти, во втором - в том же участке динамической памяти будет помещено новое значение.

Для присваивания значений переменным в AutoLISP имеются две функции:

1. **Функция (SETQ <переменная> <значение> ...) - меняет значение переменной.**

Аргументами функции является перечень параметров (<переменная> <выражение>). Функция присваивает в <переменную1> значение <выражения1>, затем присваивает в <переменную2> значение <выражения2> и т.д. В качестве <выражения> можно использовать как атомы, так и функции, возвращающие некое значение простого типа. Порядок выполнения нескольких присваиваний определяется слева направо.

Функция ***возвращает результат последнего присваивания***.

*Например*:

(Setq a 10) - данная функция присваивает переменной **а** значение 10. Возвращаемым значением функции будет число 10 (как значение переменной **а)**.

(Setq a 10 b (+ a 10)) - данная функция сначала присваивает переменной **а** значение 10, затем присваивает переменной **b** сумму значения переменной **а** и числа 10 (результат функции суммирования), т.е. 20. и результатом, который вернет вся функция после отработки, будет число 20 - значение переменной **b**.

Изменение значения переменной ***NIL*** освобождает занимаемую ее значениемобласти памяти.

(SETQ a NIL) - сохраняется сама переменная **а**, но для ее значения память не выделяется.

2. ***Функция (SET <переменная> <выражение> ...) - работает с переменными.***

То есть записывает в <переменную> не значение, а адрес (указатель) другой переменной, иначе говоря, заставляет ссылаться <переменную> нату же область памяти, что и второй аргумент функции Set.

*Например*:

(SET a b) заставляет переменную **а** ссылаться на ту же область что и переменная **b**.

## Дополнительные функции

**Создание и обработка списка данных**

Очень часто возникают ситуации, когда в одну переменную необходимо записать список значений (например, координаты точки, размерные параметры). Но интерпретатор языка LISP воспринимает любой список как руководство к действию и пытается сразу же его вычислить. Как же объяснить интерпретатору, что данный список содержит данные для вычислений, а не руководство к действию?

Для этого в AutoLISP существует специальная **функция, отключающая восприятие списка как функции:**

**(QUOTE <выражение>)**

***возвращает <выражение>, не выполняя его***. В качестве альтернативной записи данной функции может использоваться знак апострофа ( ' ).

*Например*:

(Quote 10 20 30) вернет ( 10 20 30)

(Quote "ASD" 2 5) вернет ( "ASD" 2 5)

' (+ 10 34.25) вернет (+ 10 34.25)

Т.е. для записи в одну переменную нескольких значений мы можем использовать функцию Quote, например так:

(Set a (Quote 10 20 30) ).

**Функция**

**(EVAL <выражение>)**

возвращает результат вычисления <выражения>, где <выражение> - запись любой функции, значения, выражения языка AutoLISP, т.е. выполняет прямо противоположные функции QUOTE действия:

*Например*, дано:

(setq a 123)

(setq b 'a)

тогда:

|  |  |
| --- | --- |
| (eval 4.0)  (eval (abs -10) )  (eval a)  (eval b)  (eval (+ 20 10 5) ) | возвращает 4.000000  возвращает 10  возвращает 123  возвращает 123  возвращает 35 |

Использование функции EVAL позволяет в ходе выполнения программы сформировать список каких-то функций (действий), а затем в нужный момент исполнить его.

## Функции для обработки списков

В процессе работы над созданием программы-параметризатора очень часто появляется необходимость объединять несколько значений в список (например, координаты точки, результаты пользовательских функций).

***Для объединения нескольких значений в список*** используется стандартная функция AutoLISP

***(LIST <аргумент 1> <аргумент 2> ...<аргумент n>)***

*Например*:

(list a 10 b) - вернет список из трех значений (a 10 b)

(list a (list 10 b) b) - вернет список из трех значений, одно из которых - тоже является списком (a (10 b) b).

Каждое значение списка имеет индекс, причем нумерация *начинается с* *нуля*.

Почти так же часто возникает обратная ситуация, когда из уже сформированного списка, записанного в некоторую (одну!) переменную, необходимо выделить какой-то один элемент, или все элементы и записать их каждый в отдельную переменную (или произвести над ними какие-то действия). Для решения такой проблемы в AutoLISP существует большое количество встроенных функций. Мы же с вами рассмотрим только одну, наиболее общую из них.

Функция

**(Nth <имя списка> <индекс элемента>)**

***возвращает значение под номером <индекс элемента> из списка <имя списка>.*** Возвращаемое значение может либо сразу использоваться в каком-либо выражении, либо сохраняется в дополнительной переменной с помощью функции присваивания Setq.

*Например*: если было дано

(Setq a (list 10 20 30)

(Setq b (list 10 (list 20) 30) )

то

(Nth a 1) - вернет 20;

(Nth b 1) - вернет (20).

Многие функции AutoLISP не умеют работать со списком, им необходимо просто значение, поэтому чтобы в предыдущем примере из списка **b** выделить значение 20, а не список, необходимо записать следующее: (Nth (Nth b 1) 0) - т.е., сначала выделяем из списка **b** второе значение, а потом из полученного списка выделяем первое (и единственное).

Такой "многоступенчатый" доступ нужен, например, для выделения какой-то одной координаты, из списка координат некоторой точки. Как уже говорилось, имеются в AutoLISP и более изящные функции доступа к нужному элементу списка, которые вы всегда можете посмотреть в специальной литературе (если вам это стало интересно ☺).

AutoLISP содержит большой набор математических функций, некоторые из которых представлены в таблице1.

**Список наиболее часто используемых математических функций AutoLISP**

Таблица 1

|  |  |  |
| --- | --- | --- |
| Имя функции | Аргументы | Значение |
| + | а1, а2... | а1 + а2... |
| - | а1, а2... | а1 - а2... |
| \* | а1, а2... | а1 \* а2\*... |
| / | а1, а2... | а1 / а2 /... |
| 1+ | а | a + 1 |
| 1 - | а | a - 1 |
| abs | а | | a | |
| sqrt | а |  |
| exp | а |  |
| expt | a b |  |
| GCD | a b | HOD a b |
| log | a | In (a) |
| min | а1, а2... | Минимальное значение из а1, а2... |
| max | а1, а2... | Максимальное значение из а1, а2... |
| REM | a b | a/b |
| Sin | a | Sin a |
| Cos | a | Cos a |
| Atan | a | Arctg a |
| pi |  | число Пи |
| ! | a | Значение переменной а |

## Выполнение функций AutoLISP в AutoCAD

***Предусмотрено три способа:***

1. Запись функции AutoLISP ***в командной строке AutoCAD***, при этом результат выводится в следующей строке (т.к. AutoCAD имеет командный интерфейс: каждая команда выполняется сразу же после нажатия Enter).

2. Программу, т.е. пользовательскую функцию AutoLISP, можно набрать в ***любом текстовом редакторе*** и сохранить в файле в расширении lsp. Затем в AutoCAD загрузить данный файл либо через меню (**Tools → load application**), либо записать имя файла в командной строке AutoCAD. При загрузке файла через командную строку необходимо использовать команду ( **load "имя файла"**). Эта функция при ошибки загрузки возвращает имя файла, а при нормальной загрузке возвращает ссылку на последнюю функцию в файле (выводится имя функции). Имя *файла является переменной типа "Текст"*, либо выражением вычисление которого дает результат типа "Текст".Если имя задается явно, то оно заключается в *двойные кавычки*, расширение и точку перед ним не указывают. Если указывается полный путь доступа к файлу, знак " \ ". А затем, после загрузки, функцию можно вызвать на выполнение первым способом (через командную строку).

3. Пользовательскую функцию AutoLISP можно также набрать во ***встроенном в AutoCAD текстовом редакторе***. При этом файл автоматически будет сохранен с расширением .lsp в каталоге AutoCAD и загружать его дополнительно не требуется. А запуск функции на выполнение осуществляется опять-таки через командную строку AutoCAD/

Если вы внимательно прочитали все три способа, то, наверное, заметили, что запуск на выполнение функции AutoLISP в САПР AutoCAD на самом деле один - через командную строку. Это связано с тем, что, как уже говорилось, AutoCAD имеет ***командный интерфейс***, который отличается следующим:

- ***в командной строке*** можно написать ***только одну функцию*** AutoLISP (в виде списка!);

- после нажатия <Enter> ***функция немедленно выполняется***;

- ***результат*** работы функции печатается ***в следующей строке*** AutoCAD;

- изменить уже выполненную функцию через список последних выполненных команд (расположенных над командной строкой) ***нельзя!***

- ***для исправления*** функцию необходимо заново ***ввести в командную строку*** (можно, конечно, каждый раз это проделывать вручную... но не переживайте разработчики AutoCAD подумали о вас! Для возврата функций, выполненных за сеанс работы, в командную строку, можно использовать клавиши перемещения курсора <↑> и <↓>. После появления нужной функции выполняете исправления и заново запускаете ее на выполнение нажав клавишу <Enter>).

Последнее правило можно применить и к записи действий в виде пользовательских функций в отдельном файле.

***При внесении изменений в уже созданный файл необходимо сохранить файл и загрузить заново.***

И еще несколько полезных дополнений:

* Чтобы *посмотреть значение* переменной необходимо в командной строке набрать *восклицательный знак*.
* При не правильном вводе выражения (неверном указании скобок, неправильном написании имени функции и т.д.) приглашение в командной строке меняется на n> (где n - число число закрытых скобок). Но вводить в такую командную строку функцию заново не рекомендуется. Желательно с помощью клавиши <ESC> вернуть строку в стандартный вид и только потом записывать функцию AutoLISP заново.

ПРИМЕР.

Записать в обратной польской последовательности следующую функцию и выполнить ее в AutoCAD с произвольно заданными значениями:

перевод, с помощью правил обратной польской нотации, математического выражения в функцию AutoLISP мы выполним сначала на отдельном листочке (☺), чтобы максимально удобно для себя посчитать все скобки в получившейся "бесконечной" функции. Можно упростить перевод, "переведя на AutoLISP" отдельно числитель и знаменатель выражения (рисунок 1).

|  |
| --- |
| 1 шаг - запишем числитель с помощью обратной польской нотации:  (+ (expt (+ (\* 3 a) (expt x (+ a 1 ))) 2) (\* 2.3 (expt a 2)))  2 шаг - теперь знаменатель:  (- (\* 2 (+ 4 (\* 6 x))) (expt 9 -3.5 (\* 2 (expt a (= x 1)))) (- a x)))  3 шаг - а теперь объединим в функции деления:  (/ (+ (expt (+ (\* 3 a) (expt x (+ a 1 ))) 2) (\* 2.3 (expt a 2))) (- (\* 2 (+ 4 (\* 6 x))) (expt 9 -3.5 (\* 2 (expt a (= x 1)))) (- a x))) ) |

Рисунок 1. Отдельный листочек

Проверив несколько раз количество открывающихся и закрывающихся скобок, а также их правильное расположение в выражении, можно приступать к запуску получившейся функции в AutoCAD. Для этого сначала присвоим в переменные, используемые в выражении, произвольные значения, а потом запишем полученное выражение в AutoLISP в командную строку.

В итоге, последовательность действий (функций) в AutoCAD будет выглядеть так:

Command> (setq A 1.5 X 3)

3

Command> (Setq y (/ (+ (expt (+ (\* 3 a) (expt x (+ a 1 ))) 2) (\* 2.3 (expt a 2))) (- (\* 2 (+ 4 (\* 6 x))) (expt 9 -3.5 (\* 2 (expt a (= x 1)))) (- a x))) ) )

0.681012

Число, распечатанное в AutoCAD в последней строке, говорит о том, что ниши построения были правильными, и AutoCAD просчитал значение выражения с заданными исходными данными. Для проверки можно вызвать в командную строку заново первую функцию и поменять значения **а** и **х**, а затем вызвать вторую функцию и запустить ее заново.

## Организация диалога с пользователем

Экран AutoCAD может работать в двух режимах: текстовом (TEXTSCR) и графическом (GRAPHSCR).

При работе в текством режиме для организации диалога с пользователем, т.е. для вывода некоторой информации в командную строку AutoCAD и ввода необходимых для работы программы данных, используется ряд функций AutoLISP.

**Ввод данных с клавиатуры**

В случае необходимости получения некоторой информации (данных) от пользователя во время работы программы, можно использовать следующие встроенные функции AutoLISP:

* **(Getint "текст подсказки")** - считывает значение целого типа;
* **(Getreal "текст подсказки")** - считывает значения вещественного типа;
* **(Getstring "текст подсказки")** - считывает значение строкого типа (символ, строку символов); аргумент <флаг> устанавливается для указания действия над возможными пробелами в вводимой строке: если <флаг>=NIL (и по умолчанию), то текст считывается до первого пробела; если <флаг>=Т, то текстсчитывается до конца строки (нажатия Enter).

Все *функции ввода*, так называемые Get-функции, хотя и считывают они каждая свой тип данных, *объединяют следующие моменты*:

- аргумент <подсказка> - факультативный аргумент, для высвечивания в командной строке AutoCAD текста в виде подсказки для пользователя. В тексте подсказки можно использовать управляющие последовательности (таблица 2);

Таблица 2

|  |  |
| --- | --- |
| \ n | Переход на новую строку |
| \ r | Возврат каретки |
| \ t | Табуляция ( 8 пробелов) |
| \ nnn | Вывод символов с восьмеричным кодом |

- все ***GET-функции возвращают введенные с клавиатуры значения*** или NIL, если имел место пустой ввод. Поэтому, для сохранения этого значения его следует записать в переменную, т.е. осуществлять вызов GET-функций в функции присваивания SETQ.

*Например*:

(setq x (getreal "\nВведите значение координаты Х:") )

В результате выполнения указанных йункций в командной строке AutoCAD появится текстовая строка

Введите значение координаты Х:

И после ввода пользователем некоторого значения оно будет считано функцией *getreal*, а затем функцией setq присвоено в переменную х.

Для автоматического исключения наиболее очевидных ошибочных ситуаций предназначена функция

**(INITGET <сумма кодов>)**

действие которой распространяется *только на одну,* *следующую за ней*, *GET-функцию*.

*Основные коды:*

1 - запретить пустой ввод;

2 - запретить ввод с нуля;

4 - запретить ввод отрицательных чисел.

**Вывод информации на экран**

Для вывода информации на экран ( в командную строку AutoCAD) также используется некоторый набор функций AutoLISP, применение которых зависит от типа выводимой информации.

*Для вывода числовых значений* используются следующие функции:

* **(Princ <переменная (выражение)>) -** вывод **значения** без учета управляющих кодов.
* **(Prinl <переменная (выражение)>)**  - вывод  **значения**  с учетом управляющих кодов (которые должны быть указаны как текстовое значение, т.е. в двойных кавычках).
* **(Print <переменная (выражение)>) -** вывод **значения** с учетом управляющих кодов (которые должны быть указаны как текстовое значение, т.е. в двойных кавычках), с новой строки и с пробелом в конце.

*Для вывода текстовых значений:*

* **(Prompt "текст") -** вывод **<текста> на экран с учетом управляющих кодов.**
* **(Terpri )** - вывод на экран  **пустой строки**.

***Возвращаемым значением*** каждой из вышеперечисленных функций ***является то значение, которой она печатает в командной строке*** AutoCAD. И только *функция* ***Prinl*** *не возвращает никакого значения,* поэтому она удобна для тихого завершения программы (пользовательской функции), т.е. используется в том случае, когда некая пользовательская функция Prinl записывается в качестве последнего выражения пользовательской функции.

# Параметрическое проектирование

Изделия, отличающиеся друг от друга только размерами, но не формой - называются ***конструктивно однородными изделиями***. Яркий пример таких изделий - технологическая оснастка: кондукторы, материальный инструмент, пресс-формы и др. (рисунок 2). Поэтому, при отрисовке чертежей таких деталей изменяются только какие-то размеры, а последовательность действий сохраняется неизменной.

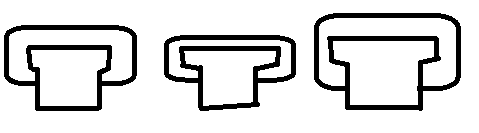


Рисунок 2. Пример конструктивно однородных изделий

Следовательно, возможно создание программы, которая выполняет эти действия по заданному алгоритму, подставляя введенные пользователем данные в нужный момент. Такая программа называется ***программой-параметром,*** а подход - ***параметрическим проектированием***.

К достоинствам параметрического проектирования можно отнести:

- упрощение процесса проектирования конструктивно однородных изделий, что ведет к существенному сокращению временных, трудовых и др. затрат для достижения результата;

- возможность контролирования в процессе проектирования.

Для каждого класса однородных изделий набор размерных параметров выбирается исходя из конфигурации. Для сложных деталей возможны различные наборы, программист-конструктор выбирает нужный, руководствуясь технологическими соображениями. Учитывается при этом и то, что проекция детали - это векторное изображения, состоящее из множества базовых геометрических элементов, предлагаемых САПР ( в данном случае AutoCAD).

Для построения каждого геометрического элемента необходим определенный набор точек - так называемых ***базовых точек***  - по которым определяется его положение на плоскости чертежа. Каждый элемент имеет свой стандартный набор базовых точек, выбор которого зависит от способа построения ( например, для отрезка необходимы две точки: координаты начала и конца отрезка, для дуги - координаты трех точек: начало, центр и конец дуги ( возможны и другие варианты построения дуги, с другим набором базовых точек) т т.д.).

Программа-парамезатор должна, основываясь на введенных пользователем размерных параметрах, рассчитывать координаты базовых точек детали выбранной конфигурации необходимо задаться  ***точкой привязки*** - опорной точкой, определяющей положение проекции на листе чертежа.

## Использование команд AutoCAD из программы на AutoLISP

А теперь, прежде чем перейти к следующему этапу формирования нашей программы, необходимо пополнить наши знания об языке AutoLISP и AutoCAD.

Вам, конечно, известно, что AutoCAD имеет встроенный набор команд, которые вводятся с клавиатуры в ответ на приглашение

>> Command :

и в большинстве своем запрашивают значения некоторых параметров, последовательность ввода которых определяется форматом команды. После выполнения действие команды должно быть прервано, для этого необходимо нажать два раза клавишу <Enter>.

Для выполнения многих команд AutoCAD возникает необходимость ввода координат одной или нескольких точек. При этом следует учитывать, что координаты можно задать в виде:

1) (х у) - абсолютные координаты точек;

2) @dx dy. - приращение по осям координат от текущей точки;

3) @ r<a - перемещение от текущей точки под углом "а" на растояние "r".

Также значение параметров, задаваемых пользователем для выбранной команды, могут быть следующих типов:

- строка (имя файла, название команд);

- выбор объекта;

- число (номер цвета);

- ключевые слова;

- прерывание выполнения команды.

Так как мы изучаем составление программ на языке AutoLISP, основной задачей которых есть отрисовка чертежа некоторой детали именно в AutoCAD, то, конечно же, существуют специальные приемы (функции), с помощью которых можно из программы на AutoLISP обратиться к нужной команде AutoCAD. При этом необходимо конечно же знать эти самые команды в AutoCAD, их формат ввода. Это можно посмотреть в специальной литературе или вручную, с клавиатуры выполнить чертеж в AutoCAD и запомнить все вводимые данные, т.к. автор данной работы не ставил своей целью ознакомление читателей с командами AutoCAD. Хотя...перечень некоторых команд AutoCAD автор все-таки приводит в Приложении 2, (ну, чтоб совсем вам не отвлекаться на посторонние поиски ☺).

***Вызов*** большинства ***команд AutoCAD из программы на AutoLISP*** можно осуществить при помощи функции **COMMAND:**

**(COMMAND "name" "")**

где

- "**name"**  - имя вызываемой команды AutoCAD, записанное в  *двойных кавычках*;

-  **-** параметры вызываемой команды, записанные через пробел.

При этом следует учитывать следующее:

* координаты точек указываются в виде *списка* из двух или трех чисел:
  + - с использованием функции quote;
    - или текстовой строкой;
    - или с помощью функции List.
* имена команд, строковые значения и установки, т.е. все константы, являющиеся *параметрами функции command*, задаются как  *текстовые строки* (в двойных кавычках);
* численные значения записываются явно, без дополнительного оформления;
  + - "" - прерывание команды (вызов функции "command" без аргументов.

Главной особенностью функции COMMAND является возможность подстановки в качестве параметров результатов выполнения других функций, т.е. любой параметр функции COMMAND можно заменить имя переменной или выражение AutoLISP, и при этом данный параметр примет значение этой переменной или результат вычисления выражения.

* В качестве аргумента функции COMMAND можно использовать дополнительный параметр PAUSE, позволяющий ввести следующее значение параметра вручную.

*Внутри функции "COMMAND" нельзя вызывать функции ввода данных,*  а так же из программы на AutoLISP нельзя вызывать следующие команды AutoCAD: ДТЕКСТ; ЭСКИЗ; ПЕЧАТАЙ; ЧЕРТИ; ПАКЕТ. А также команды, определенные при помощи функции (DEFUNC:).

**Системные переменные в AutoCAD**

***Системная переменная -***  *это ячейка памяти, содержащая определенное значение и имеющая неизменное имя.* Значение системных переменных задают различные режимы работы команд AutoCAD, единицы измерения, форматы отрисовок стандартных элементов чертежа и т.п. Их значения сохраняются в файле чертежа.

*Переменная AutoLISP и системная переменная AutoCAD - это абсолютно разные вещи.* К системным переменным AutoCAD нельзя обращаться на прямую, как к обычным переменным AutoLISP.

***Для доступа к системным переменным AutoCAD из программы на AutoLISP***  имеются две функции:

1. ***Функция* (GETVAR "имя") *возвращает значение системной переменной*** с именем "имя", заданным как текстовая строка.

Например, системная переменна Lastpoint содержит координаты текущей точки. Для использования в программе значение этой системной переменной следует использовать функцию Getvar:

(GETVAR "Lastpoint")

2. ***Функция* (SETVAR "имя" <значение>)  *устанавливает в системную переменную***  с именем "имя" ***заданное*** <Значение>. Данной функцией следует пользоваться очень осторожно, т.к. изменение значения системной переменной на неправильное может привести к непредсказуемым последствиям.

Оглавление

[1. ВВЕДЕНИЕ 3](#_Toc444205226)

[2. ОБЩИЕ ТРЕБОВАНИЯ К КОНТРОЛЬНОЙ РАБОТЕ 5](#_Toc444205227)

[Последовательность выполнения контрольной работы 5](#_Toc444205228)

[Структура контрольной работы 6](#_Toc444205229)

[4. ВАРИАНТЫ КОНТРОЛЬНОЙ РАБОТЫ ПО ДИСЦИПЛИНЕ «ПРОГРАММИРОВАНИЕ» 8](#_Toc444205230)

[Варианты теоретической части 8](#_Toc444205231)

[Варианты практической части: 9](#_Toc444205232)

[5. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ 22](#_Toc444205233)

[Основные понятия AutoLISP 23](#_Toc444205234)

[Особенности языка AutoLISP 23](#_Toc444205235)

[Типы данных в AutoLISP 26](#_Toc444205236)

[Присваивание значений в AutoLISP 33](#_Toc444205237)

[Дополнительные функции 34](#_Toc444205238)

[Функции для обработки списков 36](#_Toc444205239)

[Выполнение функций AutoLISP в AutoCAD 38](#_Toc444205240)

[Организация диалога с пользователем 41](#_Toc444205241)

[Параметрическое проектирование 44](#_Toc444205242)

[Использование команд AutoCAD из программы на AutoLISP 45](#_Toc444205243)

Учебное издание

Оксана Федоровна **Абрамова**

Дмитрий Николаевич **Лясин**

*Применение языка функционального программирования AutoLISP к решению задачи автоматизированного построения типизированных чертежей*

*методические указания к контрольной работе*

План электронных изданий 2016 г. Поз. № 16В

Подписано на « Выпуск в свет» .Уч-изд. л. .

На магнитоносителе.

Волгоградский государственный технический университет.

400131, г. Волгоград, пр. Ленина, 28, корп. 1.