МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра механики, мехатроники и робототехники

ПРОГРАММИРОВАНИЕ ПРОМЫШЛЕННОГО РОБОТА KUKA KR AGILUS

методические указания по выполнению лабораторных работ для студентов направлений 15.03.06 «Мехатроника и робототехника», 15.04.06 «Мехатроника и робототехника» всех форм обучения

Курск 2020

УДК 621.865
Составители: С.Ф. Яцун, П.А. Безмен

Рецензент:

Кандидат технических наук, доцент Юго-Западного государственного университета *Е.Н. Политов*

Программирование промышленного робота KUKA KR AGILUS: методические указания по выполнению лабораторных работ для студентов направлений 15.03.06 «Мехатроника и робототехника», 15.04.06 «Мехатроника и робототехника» всех форм обучения / Юго-Зап. гос. ун-т; сост. С.Ф. Яцун, П.А. Безмен. Курск, 2020. – 62 с.

Методические указания содержат основные требования по выполнению и защите лабораторных работ, включают индивидуальные задания и справочный материал.

Работа над пособием выполнена в рамках проекта Erasmus+ APPLE (Applied curricula in space exploration and intelligent robotic systems).

Методические указания предназначены для студентов направлений 15.03.06 «Мехатроника и робототехника», 15.04.06 «Мехатроника и робототехника» всех форм обучения.

Текст печатается в авторской редакции

Подписано в печать . Формат 60x84 1/16. Усл.печ.л. Уч.-изд.л. Тираж экз. Заказ. Бесплатно.
Юго-Западный государственный университет.
305040 Курск, ул. 50 лет Октября, 94.

Содержание

1 Теоретические сведения

Цель лабораторных работ – изучение языка программирования промышленного робота KRL (англ. *KUKA Robot Language* – язык роботов KUKA) с использованием промышленного робота-манипулятора компании KUKA модели KR 10 R1100 sixx (далее KR AGILUS) для формирования у обучающихся знаний, умений и навыков управления промышленными роботами.

Одной из важнейших задач управления роботом является программирование его движения. В языке KRL есть много способов решения данной задачи: можно управлять как отдельно каждым сервоприводом, так и полностью всем манипулятором, задавая лишь координату инструмента (например, схвата манипулятора).

Лабораторные работы выполняются в аудитории, оснащенной промышленным роботом KR AGILUS.

Все отчеты по лабораторным работам выполняются на стандартных листах формата А4, скреплённых в тетрадь. Пример оформления титульного листа отчета по лабораторной работе приведен на рис. 1.

МИНОБРНАУКИРОССИИ

Федеральное государственное образовательное учреждение высшего образования

«Юго-Западный государственный университет» (ЮЗГУ)

Кафедра механики, мехатроники и робототехники

Отчет по лабораторной работе №1

Движение манипулятора от точки к точке

Вариант 1

Выполнил:	ст. гр. МТ-01	Иванов И.С.
Проверил:	к.т.н., доц.	Петров П.А.

Курск 2020

Рис. 1. Пример оформления титульного листа

Отчет по лабораторной работе должен быть достаточно кратким, без лишних подробных пояснений и теоретических выводов, имеющихся в учебниках и других учебных пособиях, но содержащим:

- титульный лист;
- текст задания на выполнение лабораторной работы в соответствие с вариантом задания студента;
- листинг написанной студентом программы с комментариями;
- графическое изображение траектории движения рабочего органа (инструмента) манипулятора с указанием координат опорных (узловых) её точек в прямоугольной системе координат с использованием координатной сетки;
- выводы о проделанной студентом лабораторной работе.

Отчет по лабораторной работе оформляется с применением ЭВМ в любом текстовом редакторе (Microsoft Word, Apache OpenOffice и др.), при этом желательно применение шрифтов 12 или 14 кегля. Отчет по лабораторной работе полностью печатается на принтере. После печати листы отчета скрепляются.

Изложение текстового материала отчета по лабораторной работе следует вести в безличной форме. Текст всего отчета должен быть выдержан в едином стиле.

Каждым студентом все отчеты по лабораторным работам должны выполняться и отдаваться преподавателю на проверку в сроки, предусмотренные графиком работы студентов в текущем семестре. После исправления всех ошибок, отмеченных преподавателем при проверке, результаты каждой лабораторной работы должны быть защищены студентом.

На защите лабораторной работы студенту преподавателем задаются контрольные вопросы по соответствующему разделу курса. Если студент дал ответы на все заданные вопросы, и у преподавателя нет никаких дополнительных замечаний по отчету, то защита лабораторной работы считается законченной.

1.1 Структура простейшей программы на языке KRL

Любая программа на языке KRL состоит из двух текстовых файлов с расширениями *.src и *.dat, где * - одно и то же имя. В файле *.src (файл кода) находится код программы, а в файле *.dat (файл данных) расположены описания переменных, точек, массивов и т.д. для сокращения размера файла *.src.

Файл *.src состоит из «главной» и «дополнительных» функций. «Главная» функция должна называться также как и сам (*.src) файл, в котором она находится.

Функция на языке KRL состоит из трёх основных разделов – рис 1.1.1.

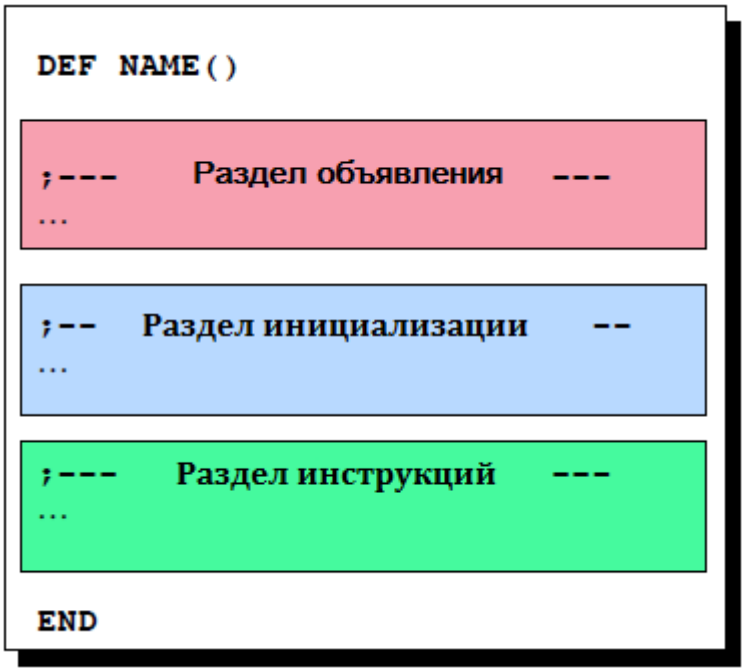


Рис 1.1.1 Разделы функции

Раздел объявления служит для объявления переменных, массивов, точек, структур и т.д.

Раздел инициализации нужен для инициализации переменных, массивов и т.д., объявленных в разделе объявления.

Раздел инструкций является основным разделом, в котором находится основной код программы.

Объявление функции начинается с ключевого слова «DEF» и заканчивается словом «END».

Например:

```
DEF TEST()
;Объявление
;Инициализация
;Основной код
END
```

В примере мы видим, что главная функция называется TEST – это значит, что файлы программы называются «TEST.SRC» и «TEST.DAT» соответственно.

Пример файла данных PROG_1.dat приведен на рис. 1.1.2.

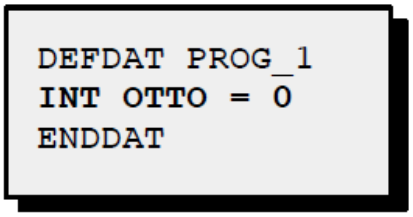


Рис 1.1.2 Файл данных

Круглые скобки после имени функции сообщают интерпретатору KRL, что ваша программа использует функцию. В разделе 1.10 «Функции и параметры» дана информация о формате записи функций и передаче им параметров (аргументов).

1.2 Переменные и типы данных

В языке KRL существуют как простые, так и структурированные данные.

Для объявления любой переменной желательно использовать ключевое слово «DECL». «DECL» резервирует память под заданную переменную. Использование данного слова в некоторых случаях не обязательно, но желательно: иначе могут появиться непреднамеренные сбои программы.

1.2.1 Простые типы данных

Под простыми типами данных, подразумевается ряд основных типов данных, которые доступны в большинстве языков программирования, они приведены в таблице 1.2.1.1.

Таблица 1.2.1.1 – Простые типы данных

Типы данных	Целое	С точкой	Логический	Символьный
Ключевое слово	**INT**	**REAL**	**BOOL**	**CHAR**
Диапазон значений	$-2^{31}...2^{31}-1$	±1.1E-38... ±3.4E+38	TRUE, FALSE	ASCII character

Объявление переменных почти ни чем не отличается от объявления в других языках программирования за исключением слова «DECL».

Добавим в предыдущий пример несколько переменных.VAR_1 и VAR_2:

```
DEF TEST()
;------------Объявление---------
DECL INT VAR_1
DECL REAL VAR_2
;------------------------------
;------------Инициализация------
;------------Основной код-------
END
```

Здесь переменная VAR_1 имеет тип INT, а VAR_2 тип REAL. Также этот язык поддерживает массивы данных, например:

```
DECL INT MASS[7]

DECL REAL MATRIX[5,4]
```

1.2.2 Структурные типы данных

Если разные типы данных должны быть объединены воедино, то нам на помощь приходит оператор STRUC, который позволяет создать новый составной тип данных.

Типичным примером использования составного типа данных является POS, объявленный, в файле $OPERATE.SRC.

Он состоит из шести переменных REAL и двух типа INT.

```
STRUC POS REAL X, Y, Z, A, B, C, INT S, T
```

Например, если будет использоваться переменная `POSITION` типа POS, то требуется записать:

```
DECL POS POSITION
```

Присвоить значения отдельным элементам структуры можно с помощью оператора точка «.», он позволяет обращаться отдельно к каждому элементу структуры:

```
POSITION.X = 34.4
POSITION.Y = -23.2
POSITION.Z = 100.0
POSITION.A = 90
POSITION.B = 29.5
POSITION.C = 3.5
POSITION.S = 2
POSITION.T = 6
```

или совместно при помощи «совокупности»:

```
POSITION={X 34.4,Y -23.2,Z 100.0,A 90,B 29.5,C 3.5,S 2,T
6}
```

Эти две записи выполняют одинаковую инициализацию.

Также тип данных может указываться в начале инициализации структуры.

Например:

```
POSITION={POS: X 230,Y 0.0,Z 342.5}
```

В языке KRL уже заранее определены основные структурные типы – табл. 1.2.2.1

Таблица 1.2.2.1 – Типы структур

```
STRUC AXIS    REAL  A1,A2,A3,A4,A5,A6
STRUC E6AXIS  REAL  A1,A2,A3,A4,A5,A6,E1,E2,E3,E4,E5,E6
STRUC FRAME   REAL  X,Y,Z,A,B,C
STRUC POS     REAL  X,Y,Z,A,B,C, INT S,T
STRUC E6POS   REAL  X,Y,Z,A,B,C,E1,E2,E3,E4,E5,E6, INT S,T
```

1.3 Арифметические операторы

Язык программирования KRL поддерживает все основные арифметические операции с типами данных INT и REAL – табл. 1.3.1

Таблица 1.3.1 - Арифметические операции

Оператор	Описание
+	Сложение
–	Вычитание
*	Умножение
/	Деление

В следующем примере демонстрируется выполнение различных арифметических операций:

```
DEF ARITH()
;------- Declaration section -------
INT A,B,C
REAL K,L,M
;---------- Initialization ---------
```

```
;Все переменные являются недействительными до
инициализации!
A = 2 ;A=2
B = 9.8 ;B=10
C = 7/4 ;C=1
K = 3.5 ;K=3.5
L = 0.1 E01 ;L=1.0
M = 3 ;M=3.0
;------------- Main section ------------
A = A * C ;A=2
B = B - 'HB';B=-1
C = C + K ;C=5
K = K * 10 ;K=35.0
L = 10 / 4 ;L=2.0
L = 10 / 4.0 ;L=2.5
L = 10 / 4. ;L=2.5
L = 10./4 ;L=2.5
C = 10./ 4. ;C=3
M = (10/3) * M ;M=9.0
END
```

1.4 Операторы сравнения

Операторы сравнения языка KRL, которые можно использовать, приведены в таблице 1.4.1.

Результат сравнения имеет тип данных BOOL, следовательно, результат будет иметь значение TRUE (истина) или FALSE (ложь).

Таблица 1.4.1 – Операторы сравнения

Оператор	Описание	Допустимые типы данных
==	равно	INT, REAL, CHAR, ENUM, BOOL
<>	не равно	INT, REAL, CHAR, ENUM, BOOL
>	больше чем	INT, REAL, CHAR, ENUM
<	меньше чем	INT, REAL, CHAR, ENUM
>=	больше либо равно	INT, REAL, CHAR, ENUM
<=	меньше либо равно	INT, REAL, CHAR, ENUM

Пример использования операторов сравнения:

```
DEF SL()

BOOL A,B

B = 10 < 3 ;B=FALSE
A = 10/3 == 3 ;A=TRUE
B = ((B == A) <> (10.00001 >= 10)) == TRUE ;B=TRUE
A = "F" < "Z" ;A=TRUE

END
```

1.5 Логические операторы

Логические операторы используются для выполнения логических операций над булевыми переменными, константами и простыми логическими выражениям, которые формируются с помощью операторов сравнения. Например, выражение

```
(A > 5) AND (A < 12)
```

имеет значение TRUE, только если значение в переменной **A** находится в диапазоне между 5 и 12. Логические операторы приведены в таблице 1.5.1.

Таблица 1.5.1 – Логические операторы

Оператор	Описание
NOT	Инверсия
AND	Логическое **И**
OR	Логическое **ИЛИ**
EXOR	Исключающее **ИЛИ**

Пример использования логических операторов:

```
DEF LOG()
```

```
BOOL A,B,C

A = TRUE ;A=TRUE
B = NOT A ;B=FALSE
C = (A AND B) OR NOT (B EXOR NOT A) ;C=TRUE
A = NOT NOT C ;A=TRUE


END
```

1.6 Комментарии

Комментарии являются важной частью любой компьютерной программы. Они позволяют сделать программы понятными для других. Комментарии не влияют на скорость выполнения вашей программы. Комментарии могут быть вставлены в любой точке программы. Им всегда предшествует точка с запятой "**;**".

Простой пример использования комментариев:

```
DEF COMM()

PTP P1 ;Motion to start point

;---------- Reset outputs ----------
FOR I = 1 TO 16
$OUT[I] = FALSE
ENDFOR

END
```

1.7 Скрытие разделов программы

В отличие от обычных редакторов, редактор KUKA позволяет отображать/скрывать заранее определённое содержимое текста программы.

Оператор ;FOLD позволяет скрывать (сворачивать) определённую часть кода.

Таким образом, обычный пользователь не видит этих разделов. Увидеть разделы можно только в режиме эксперта.

Для скрытия определенного блока кода нужно заключить его в пределы операторов «;FOLD» и «;ENDFOLD».

Например:

```
;FOLD RESETOUT
FOR I=1 TO 16
$OUT[I]=FALSE
ENDFOR
;ENDFOLD
```

В этом примере обычный пользователь в интерфейсе будет видеть только название скрытого блока **RESETOUT**, а всё остальное видит только эксперт.

1.8 Ветвления

Встречаются ситуации, когда программе нужно выбрать, какую операцию ей выполнить в зависимости от определенной ситуации.

1.8.1 Условный переход IF

Оператор IF служит для того, чтобы выполнить какую-либо операцию в том случае, когда условие является верным или наоборот.

Общая форма инструкции IF:

```
IF   Условие    THEN
     Инструкции, если условие равно TRUE
ELSE
     Инструкции если условие равно FALSE
ENDIF
```

Пример использования:

```
DEF UIF()
```

```
INT A,B

IF $IN[10]==FALSE THEN
   PTP HOME
ELSE
   IF A > B THEN
      $OUT[1] = TRUE
      LIN PUNKT1
   ENDIF
   A=A+1
   PTPHOME
ENDIF
END
```

Здесь мы видим, что если `$IN[10]` равно `FALSE` то `KUKA` перемещается в положение HOME, а если нет, то смотрит следующее условие `A>B`.

1.8.2 Условный переход SWITCH

Переход SWITCH позволяет осуществить выбор среди нескольких фрагментов кода, в зависимости от значения целочисленного выражения.

Пример использования:

```
DEF MAIN()

SIGNAL PROG_NR $IN[1] TO $IN[4]
;The desired program number is now stored in the
;INT variable PROG_NO by the PLC
SWITCH PROG_NO
   CASE 1 ;if PROG_NO=1
      PART_1()
   CASE 2 ;if PROG_NO=2
      PART_2()
      PART_2A()
   CASE 3,4,5;if PROG_NO=3, 4 or 5
      $OUT[3]=TRUE
      PART_345()
   DEFAULT ;if PROG_NO<>1,2,3,4,5
      ERROR_UP()
ENDSWITCH
```

```
END
```

1.9 Циклы

Цикл – это повторение выполнения одного и того же участка кода в программе. Последовательность действий, которые повторяются, называют телом цикла. Один проход цикла – это шаг или итерация. Переменные, изменяющиеся внутри цикла и влияющие на его окончание, называются параметрами цикла.

1.9.1 Оператор с предусловием WHILE

Оператор с предусловием работает следующим образом. Если условие истинно, то выполняются инструкции – тело цикла. В противном случае, цикл завершается.

Конструкция оператора имеет следующий вид:

```
WHILE Условие
        Инструкции
ENDWHILE
```

Пример использования:

```
DEF WHILE_PR()

INT X,W

X = 1
W = 1
WHILE W < 5
   X = X * W
   W = W + 1
ENDWHILE

W = 100
WHILE W < 100
   $OUT[15] = TRUE
   W = W + 1
```

```
ENDWHILE
END
```

1.9.2 Цикл с параметром FOR

Если известно требуемое количество действий (итераций) цикла, можно использовать цикл FOR. Конструкция этого цикла имеет следующий вид:

```
FOR   Счетчик = Начало TO Конец STEP Приращение
      Инструкции
ENDFOR
```

После каждого выполнения цикла переменная-счетчик увеличивается на значение приращения. Как только значение счетчика совпадет с конечным значением, цикл остановится. По умолчанию значение приращения равно 1. Следовательно, если инкремент должен быть равен 1, то секцию STEP можно не указывать. Также для приращения могут быть использованы отрицательные значения.

В следующем примере заполним двумерный массив целочисленными значениями используя циклы FOR:

```
DEF FOR_PROG()

INT I,J
INT ARRAY[10,6]

FOR I=1 TO 6
   $VEL_AXIS[I] = 100 ;Скорости всех осей 100%
ENDFOR

FOR I=1 TO 9 STEP 2
   FOR J=6 TO 1 STEP -1
      ARRAY[I,J]   = I*2 + J*J
      ARRAY[I+1,J] = I*2 + I*J
   ENDFOR
ENDFOR

END
```

После выполнения данного примера получим матрицу значений – см. табл. 1.9.2.1).

Таблица 1.9.2.1 - Заполненная матрица

Индекс		I =									
		1	2	3	4	5	6	7	8	9	10
J =	6	38	8	42	24	46	40	50	56	54	72
	5	27	7	31	21	35	35	39	49	43	63
	4	18	6	22	18	26	30	30	42	34	54
	3	11	5	15	15	19	25	23	35	27	45
	2	6	4	10	12	14	20	18	28	22	36
	1	3	3	7	9	11	15	15	21	19	27

1.9.3 Бесконечный цикл LOOP

Цикл LOOP не имеет счетчиков или условий, он выполняется бесконечное количество раз, пока вы не отключите питание робота или досрочно не завершите цикл инструкцией EXIT.

Конструкция выглядит так:

```
LOOP
    Инструкции
ENDLOOP
```

Пример использования бесконечного цикла и выхода из него:

```
DEF EXIT_PRO()

PTP HOME

LOOP ;Началоцикла
   PTP POS_1
   LIN POS_2

   IF $IN[1] == TRUE THEN
      EXIT ;Выход из цикла
   ENDIF
```

```
   CIRC HELP_1,POS_3
   PTP POS_4
ENDLOOP ;Конец бесконечного цикла

PTPHOME
END
```

1.10 Функции и параметры

Любая функция в языке KRL начинается с ключевого слова DEF (после которого идет имя функции) и заканчивается словом END.

Каждой функции можно передавать параметры. Параметры объявляются в скобках через запятую с ключевым словом :IN, например:

```
DEF SETNUMBER (NUMBER :IN)
```

В этом примере мы видим имя функции SETNUMBER с параметром NUMBER. Для того, чтобы использовать данный параметр, его нужно объявить. Объявляется он как обычная переменная. В данном случае объявление будет выглядеть так:

```
INT NUMBER
```

Ниже приведен пример программы, в которой используется функция с параметром. Функция выполняет разные действия в зависимости от значения параметра. Ветвление осуществляется с помощью оператора IF.

```
DEF test_5()

;-------Declarationsection -------
EXT BAS (BAS_COMMAND :IN,REAL :IN)
DECL AXIS HOME
```

```
;----------- Initialization ----------
BAS (#INITMOV,0)
HOME={AXIS: A1 0,A2 -90,A3 90,A4 0,A5 0,A6 0}
;----------- Main section ----------
$APO.CVEL = 5
$VEL.CP = 0.1
$ACC.CP = 0.1

$VEL_AXIS[1]=10
$VEL_AXIS[2]=10
$VEL_AXIS[3]=10
$VEL_AXIS[4]=10
$VEL_AXIS[5]=10
$VEL_AXIS[6]=10
$ACC_AXIS[1]=10
$ACC_AXIS[2]=10
$ACC_AXIS[3]=10
$ACC_AXIS[4]=10
$ACC_AXIS[5]=10
$ACC_AXIS[6]=10
PTP HOME ;BCO run
SETNUMBER(8) ;Вызов функции
END
;------Новая функция------------------------------
DEF SETNUMBER (NUMBER :IN)
INT NUMBER ;Объявлениепараметра
DECL FRAME POS_1 ;Структурная переменная
INTI
POS_1={FRAME: X 500,Y 300,Z 500,A 0,B 90,C 0}
```

```
        IF NUMBER == 8 THEN ;Еслипараметрравен 8


         PTP {X 500,Y 300,Z 700,A 0,B 90,C 0}
LOOP
         LIN {X 500,Y 200,Z 800,A 0,B 90,C 0} C_VEL

         CIRC  {X  500,Y  300,Z  1173},{X  500,Y  400,Z
800}, CA 300 C_VEL

         LIN {X 500,Y 200,Z 600,A 0,B 90,C 0} C_VEL

           CIRC {X 500,Y 300,Z 227},{X 500,Y 400,Z 600},
CA 300 C_VEL

         LIN {X 500,Y 300,Z 700,A 0,B 90,C 0}C_VEL
ENDLOOP
        ENDIF

        IF NUMBER == 4 THEN   ;Еслипараметрравен 4
            VAL_1 = 100
            FOR I = 1 TO 5
                POS_1.X = POS_1.X + VAL_1
                PTP POS_1
                PTP POS_2
                PTP POS_3
            ENDFOR
        ENDIF
END
```

В данном примере вызывается функция SETNUMBER с параметром 8. После выполнения кода рабочий орган манипулятора «рисует» цифру «8» в плоскости YOZ – рис. 1.10.1.

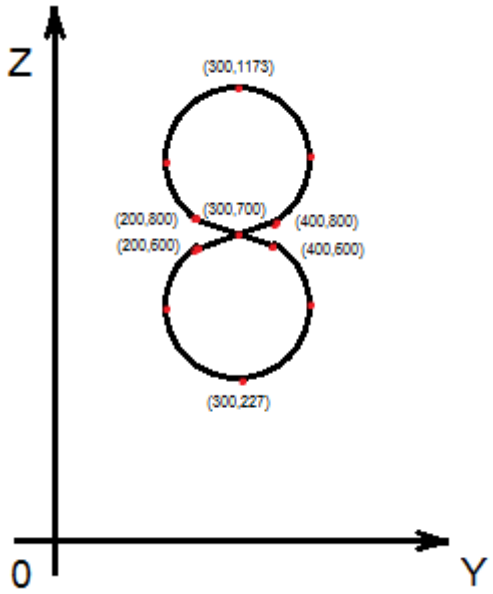


Рис. 1.10.1 Траектория движения рабочего органа манипулятора

1.11 Порты ввода/вывода

В языке KRL присутствует 1026 портов ввода и 1026 портов вывода. В стандартном блоке управления KUKA в разъеме X11 с 1-го по 16-й контакты находится ввод, а с 17-го по 20-й – вывод.

Порты ввода-вывода могут быть прочитаны/записаны с помощью системных массивов $IN[*No*] и $OUT[*No*], где: массив $IN – массив, связанный с портами ввода, массив $OUT – массив, связанный с портами вывода, *No* – номер порта (индекс массива).

В следующем примере показано, как обращаться к портам ввода/вывода:

```
DEF PVPU()

IF $IN[2] == TRUE THEN
   $OUT_C[10] = TRUE
ENDIF

$OUT_C[11] = FALSE
$OUT_C[12] = TRUE
END
```

2 Лабораторные работы

2.1 Движение манипулятора от точки к точке

Цель лабораторной работы – написать программу на языке KRL, задающую движение манипулятора и использующую функции PTP.

Для перемещения робота из одной точки пространства в другую или для поворота определенной оси манипулятора на заданный угол используют функцию PTP. Скорости и ускорения при этом задаются с помощью массивов $VEL_AXIS и $ACC_AXIS. Для перемещения какой-либо оси робота на нужный угол в качестве параметра PTP можно записать номер нужного сервопривода и значение угла в градусах, на который выходное звено привода должно повернуться. Номера сервоприводов обозначаются от A1 до An.

Например, переместим ось A3 манипулятора на угол 90°:

```
DEFPTP_AXIS()

$VEL_AXIS[3]=50 ;Скорость оси A3 равна 50%
$ACC_AXIS[3]=50 ;Ускорение оси A3 равно 50%

PTP {AXIS: A1 0,A2-90,A3 90,A4 0,A5 0,A6 0}

END
```

Ключевое слово AXIS используется здесь для указания типа структурных данных.

После выполнения данного примера робот примет положение, показанное на рис 2.1.1.

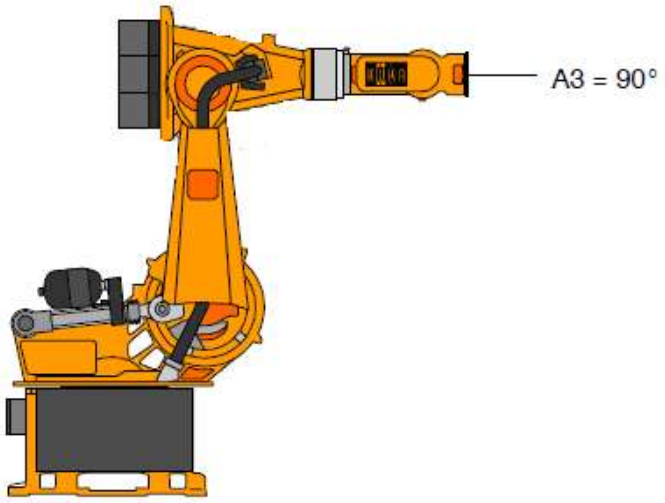


Рис. 2.1.1 Положение манипулятора при
A1 = 0, A2 = -90, A3 = 90, A4 = 0, A5 = 0, A6 = 0

В следующем примере переместим рабочий орган манипулятора на заданные координаты X, Y, Z, A, B, C. Для этого применим структуру FRAME.

```
DEF PTP_POS()

$BASE = $WORLD ;Устанавливаем систему координат

$VEL_AXIS[1]=100
$VEL_AXIS[2]=100
$VEL_AXIS[3]=100
$VEL_AXIS[4]=100
$VEL_AXIS[5]=100
$VEL_AXIS[6]=100

$ACC_AXIS[1]=100
$ACC_AXIS[2]=100
$ACC_AXIS[3]=100
$ACC_AXIS[4]=100
$ACC_AXIS[5]=100
$ACC_AXIS[6]=100
```

```
PTP {FRAME:X500,Y 800,Z 700,A 0,B 90,C 0}

END
```

В данном примере значения координат указаны в миллиметрах от основания робота KUKA. A, B, C – это углы ориентации рабочего органа манипулятора (например, схвата) - см. рис. 2.1.2.

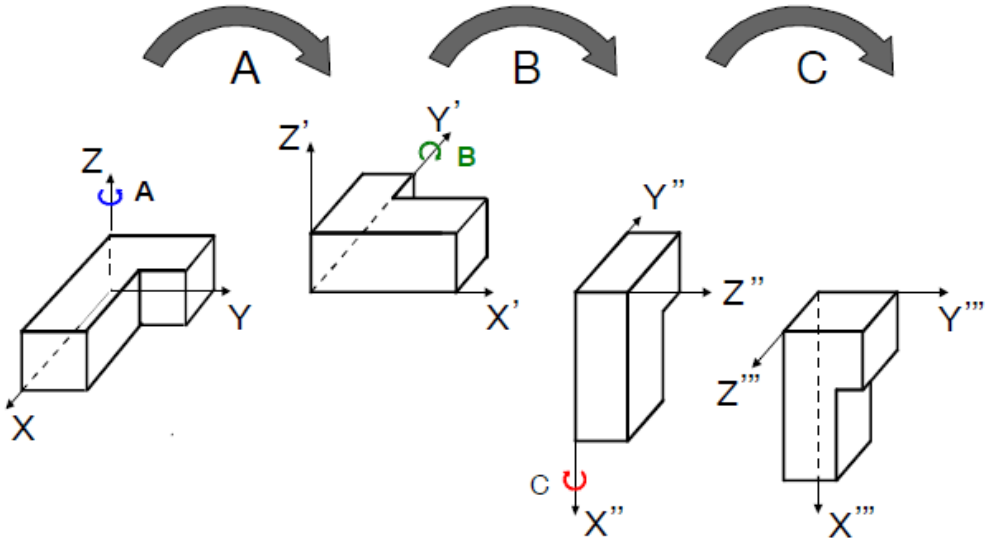


Рис. 2.1.2 Углы ориентации рабочего органа манипулятора

Приведём ещё один пример, в котором манипулятор будет перемещаться по трем разным точкам с паузой между точками в 2 секунды.

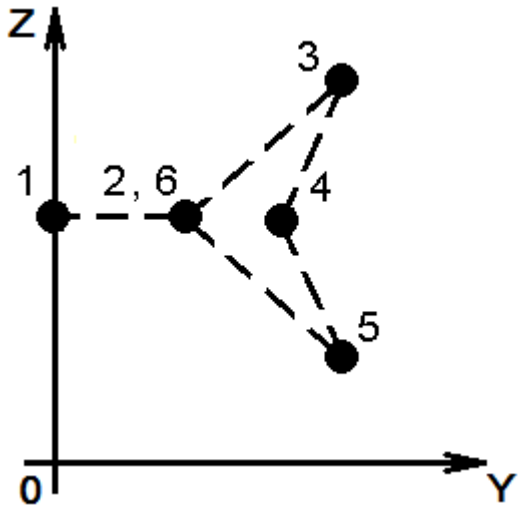
Каждый раз при вызове PTP робот начинает движение от текущего положения в заданное.

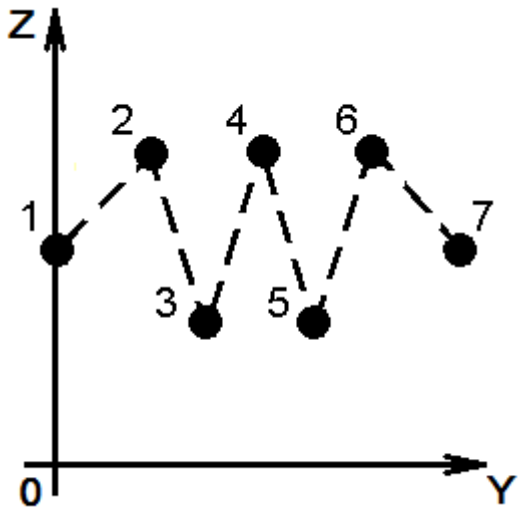
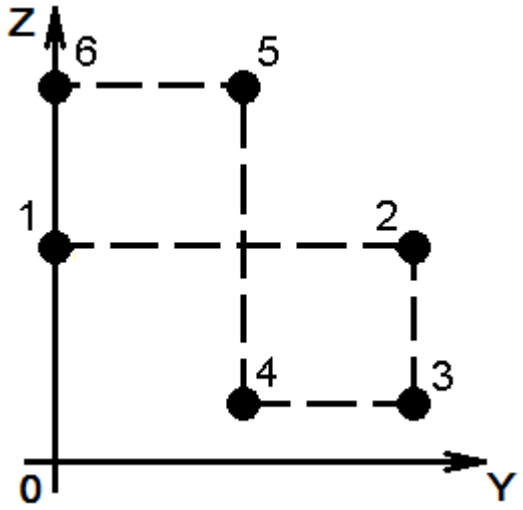
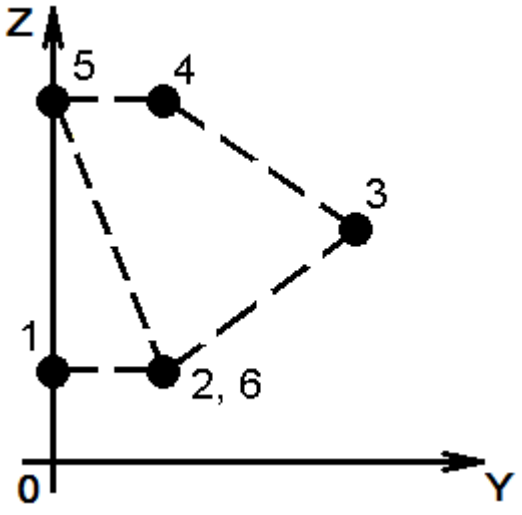
```
DEF kukatest()

$BASE = $WORLD
$VEL_AXIS[1]=1
$VEL_AXIS[2]=1
$VEL_AXIS[3]=1
$VEL_AXIS[4]=1
$VEL_AXIS[5]=1
$VEL_AXIS[6]=1
$ACC_AXIS[1]=1
```

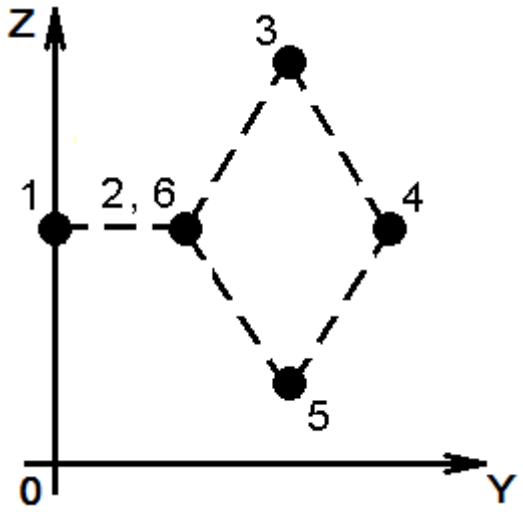
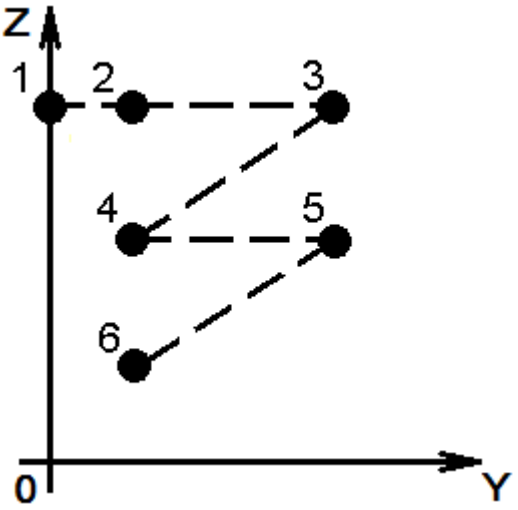
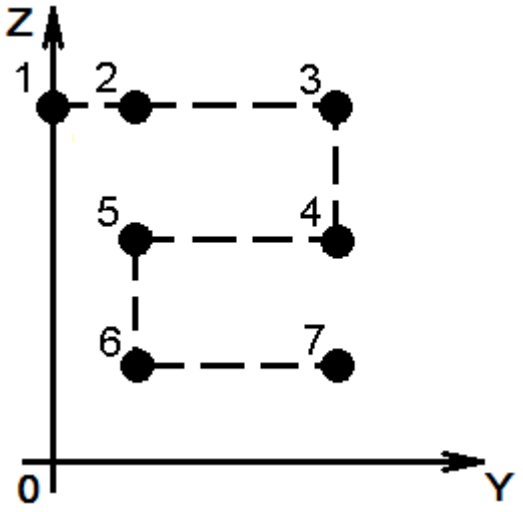
```
$ACC_AXIS[2]=1
$ACC_AXIS[3]=1
$ACC_AXIS[4]=1
$ACC_AXIS[5]=1
$ACC_AXIS[6]=1

PTP {A1 0,A2 -90,A3 90,A4 0,A5 30,A6 0}
WAIT SEC 2   ;Пауза 2 секунды
PTP {A1 45,A2 -100,A3 70,A4 20,A5 90,A6 100}
WAIT SEC 2   ;Пауза 2 секунды
PTP {A1 0,A2 -90,A3 90,A4 0,A5 30,A6 0}
WAIT SEC 2   ;Пауза 2 секунды

END
```

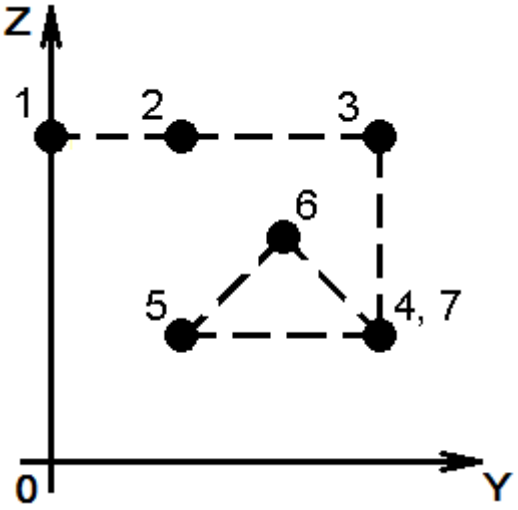
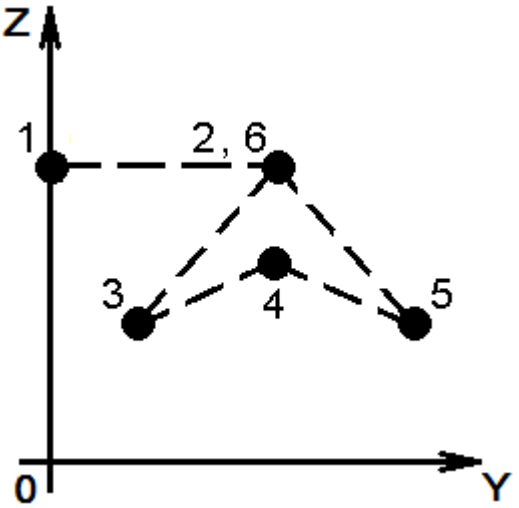
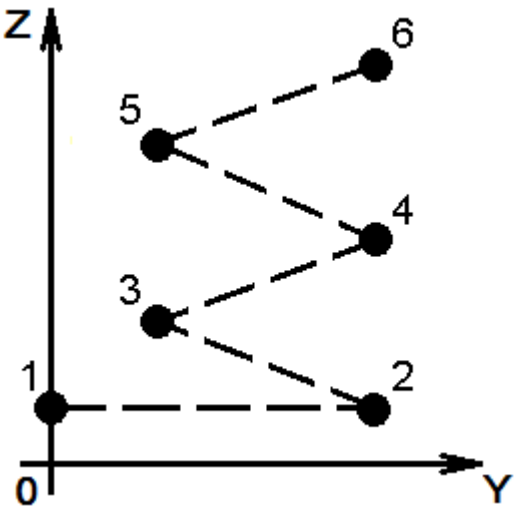
Задание: написать программу на языке KRL, задающую движение рабочего органа манипулятора по траектории в соответствии с номером варианта задания (таблица 2.1.1) и использующую вызовы функции PTP. При составлении программы рекомендуется решить обратную задачу кинематики, учитывая длины звеньев манипулятора.

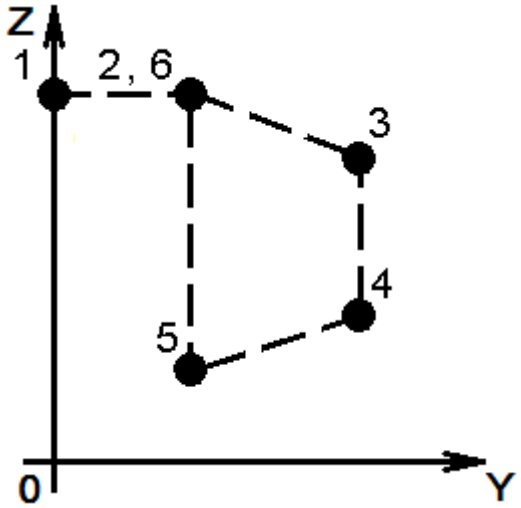
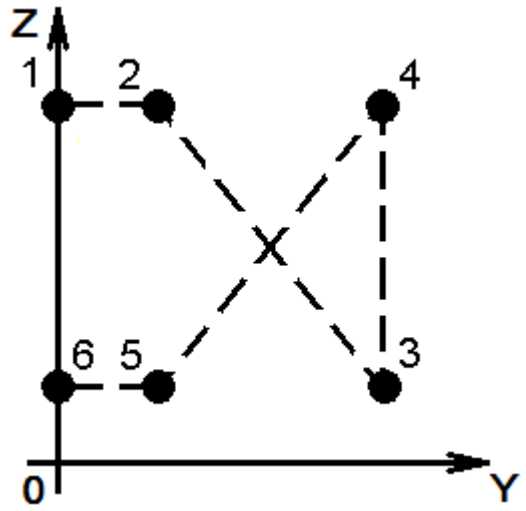
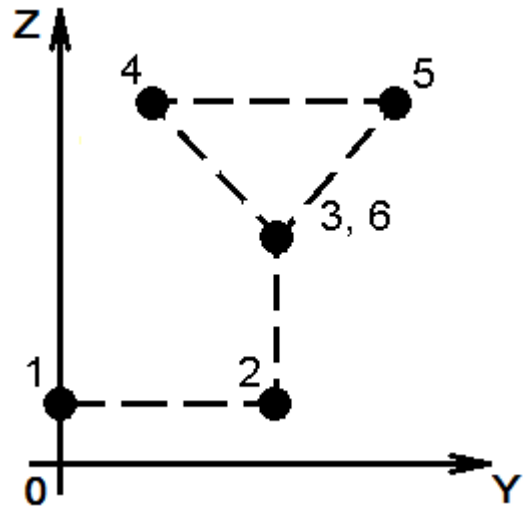
Таблица 2.1.1 – Варианты заданий

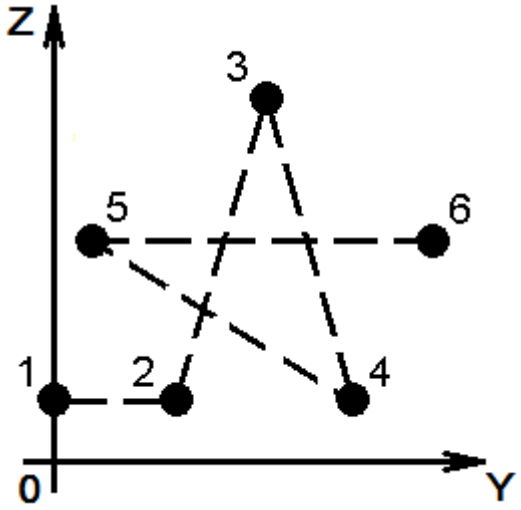
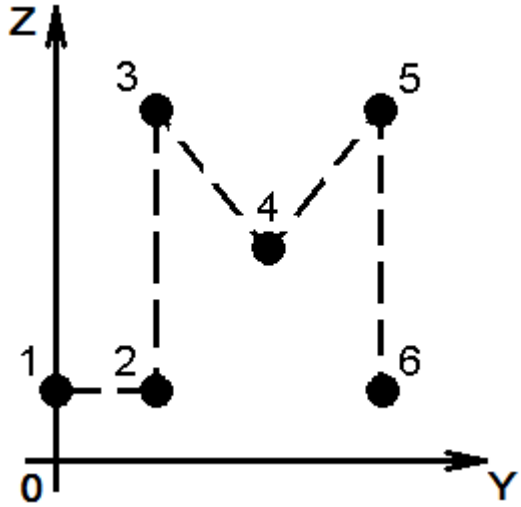
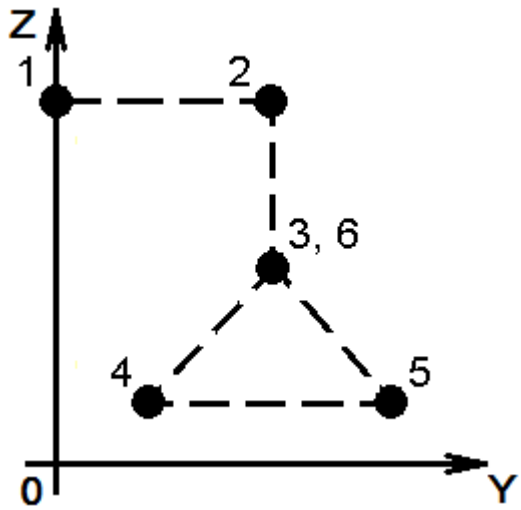
№ вар.	Форма траектории	Параметры
1		значения скоростей всех осей равны 5%, значения ускорений всех осей равны 5%, движение по траектории выполняется однократно

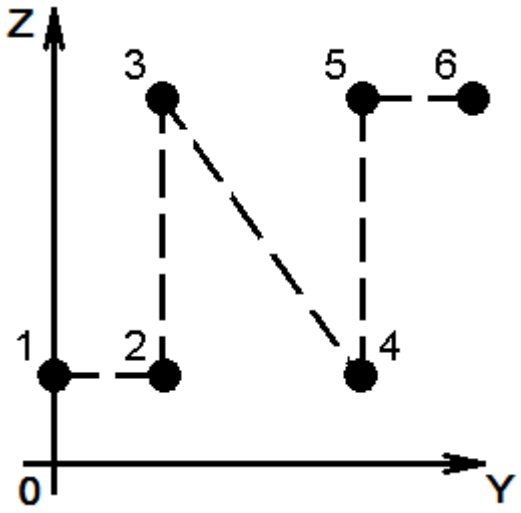
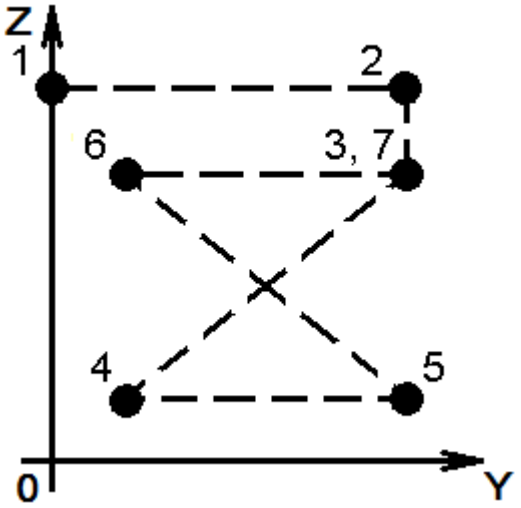
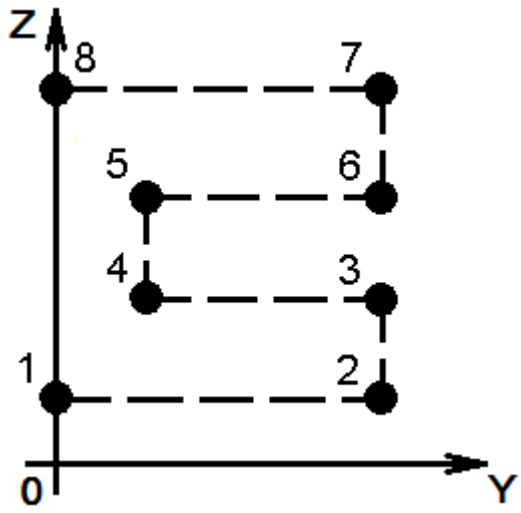
№ вар.	Форма траектории	Параметры
2		значения скоростей всех осей равны 1%, значения ускорений всех осей равны 1%, движение по траектории выполняется однократно
3		значения скоростей всех осей равны 2%, значения ускорений всех осей равны 5%, движение по траектории выполняется однократно
4		значения скоростей всех осей равны 4%, значения ускорений всех осей равны 1%, движение по траектории выполняется однократно

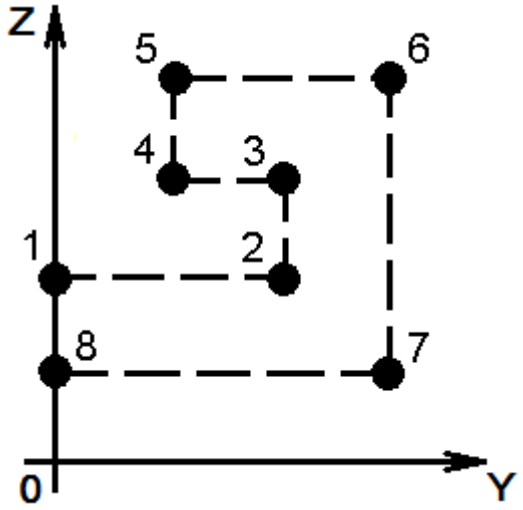
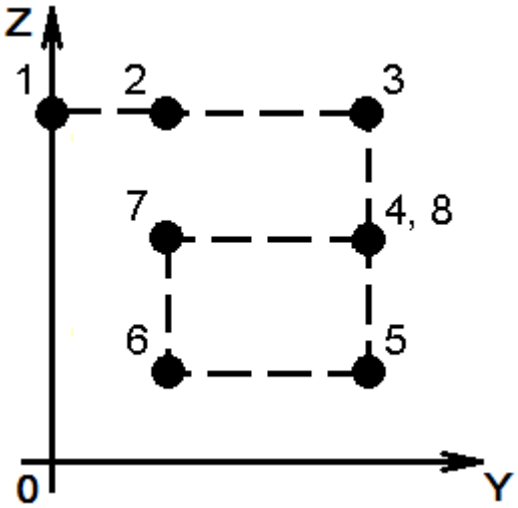
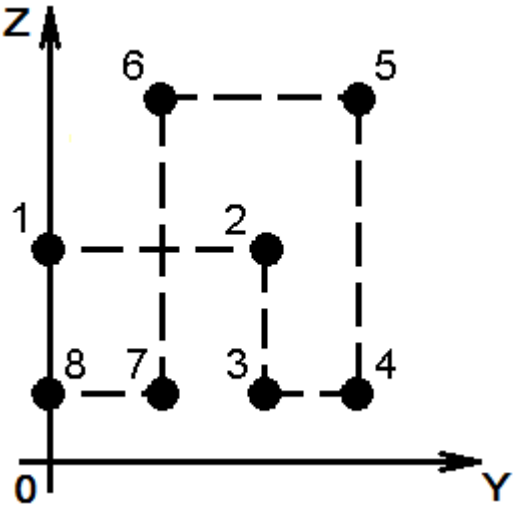
№ вар.	Форма траектории	Параметры
5		значения скоростей всех осей равны 1%, значения ускорений всех осей равны 1%, движение по траектории выполняется однократно
6		значения скоростей всех осей равны 4%, значения ускорений всех осей равны 1%, движение по траектории выполняется однократно
7		значения скоростей всех осей равны 2%, значения ускорений всех осей равны 2%, движение по траектории выполняется однократно

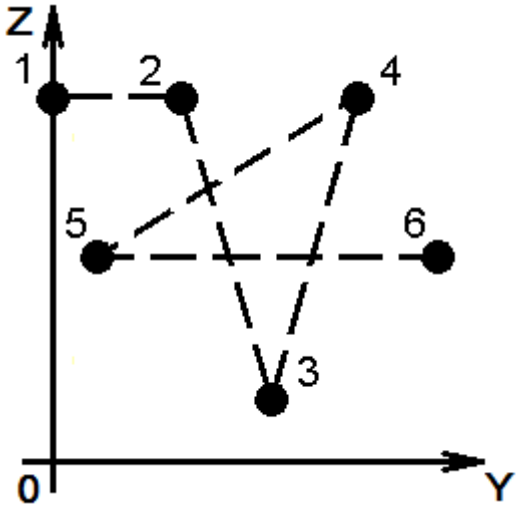
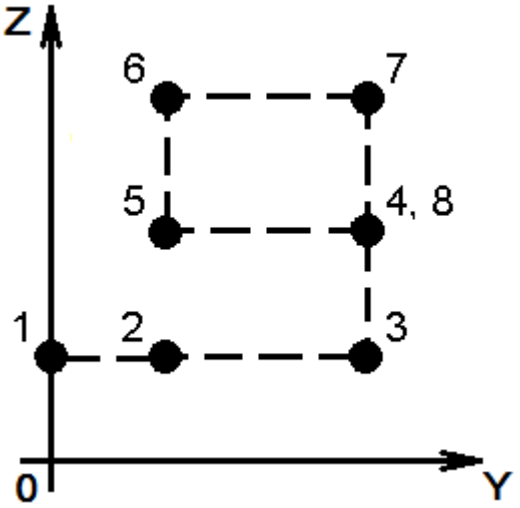
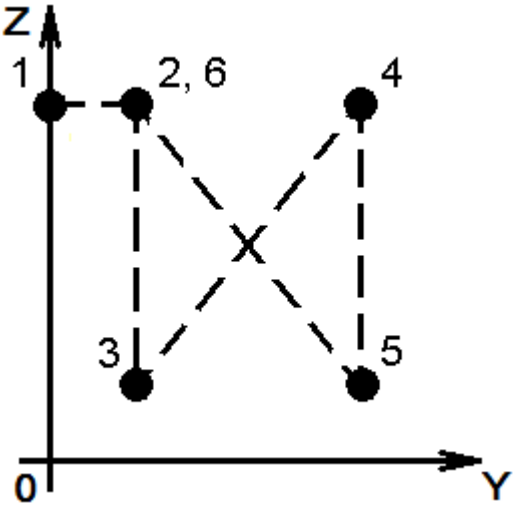
№ вар.	Форма траектории	Параметры
8		значения скоростей всех осей равны 5%, значения ускорений всех осей равны 1%, движение по траектории выполняется однократно
9		значения скоростей всех осей равны 3%, значения ускорений всех осей равны 1%, движение по траектории выполняется однократно
10		значения скоростей всех осей равны 5%, значения ускорений всех осей равны 5%, движение по траектории выполняется однократно

№ вар.	Форма траектории	Параметры
11		значения скоростей всех осей равны 1%, значения ускорений всех осей равны 2%, движение по траектории выполняется однократно
12		значения скоростей всех осей равны 5%, значения ускорений всех осей равны 1%, движение по траектории выполняется однократно
13		значения скоростей всех осей равны 1%, значения ускорений всех осей равны 2%, движение по траектории выполняется однократно

№ вар.	Форма траектории	Параметры
14		значения скоростей всех осей равны 10%, значения ускорений всех осей равны 5%, движение по траектории выполняется однократно
15		значения скоростей всех осей равны 1%, значения ускорений всех осей равны 2%, движение по траектории выполняется однократно
16		значения скоростей всех осей равны 1%, значения ускорений всех осей равны 5%, движение по траектории выполняется однократно

№ вар.	Форма траектории	Параметры
17		значения скоростей всех осей равны 1%, значения ускорений всех осей равны 2%, движение по траектории выполняется однократно
18		значения скоростей всех осей равны 10%, значения ускорений всех осей равны 10%, движение по траектории выполняется однократно
19		значения скоростей всех осей равны 2%, значения ускорений всех осей равны 2%, движение по траектории выполняется однократно

№ вар.	Форма траектории	Параметры
20		значения скоростей всех осей равны 5%, значения ускорений всех осей равны 1%, движение по траектории выполняется однократно
21		значения скоростей всех осей равны 5%, значения ускорений всех осей равны 5%, движение по траектории выполняется однократно
22		значения скоростей всех осей равны 10%, значения ускорений всех осей равны 10%, движение по траектории выполняется однократно

№ вар.	Форма траектории	Параметры
23		значения скоростей всех осей равны 2%, значения ускорений всех осей равны 2%, движение по траектории выполняется однократно
24		значения скоростей всех осей равны 2%, значения ускорений всех осей равны 1%, движение по траектории выполняется однократно
25		значения скоростей всех осей равны 1%, значения ускорений всех осей равны 1%, движение по траектории выполняется однократно

2.2 Линейное движение манипулятора

Цель лабораторной работы – написать программу на языке KRL, задающую движение манипулятора и использующую функции LIN.

Функция LIN похожа на функцию PTP, отличие в том, что в первом случае конечное звено движется по прямой линии. Ограничением функции LIN и функции CIRC является то, что нельзя провести линию из одной четверти системы координат в другую т.к. меняются знаки, а это недопустимо – см. рис. 2.2.1.

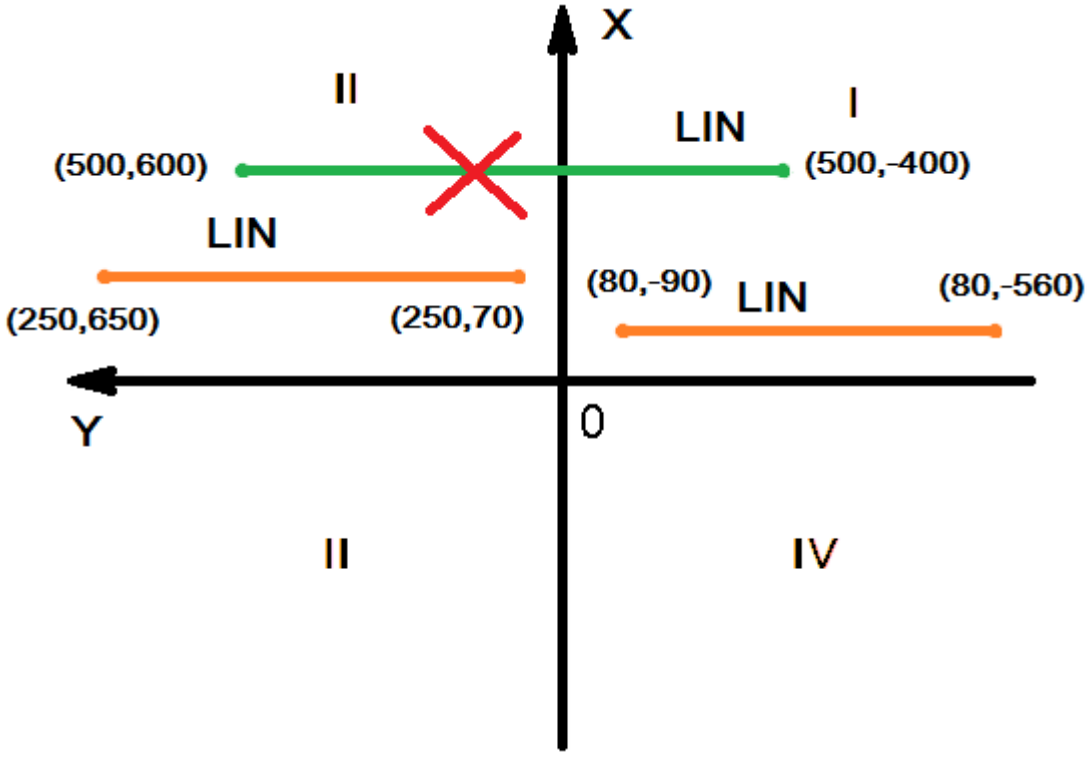


Рис. 2.2.1 Допустимые перемещения рабочего органа манипулятора в системе координат XOY

Для того, чтобы решить данную проблему, нужно использовать PTP после каждого перехода из одной четверти системы координат в другую.

Рассмотрим пример программы, в которой инструмент манипулятора циклично «рисует» квадрат. После того, как робот совершит 6 повторений одной и той же траектории движения, произойдет выход из цикла.

Перед тем как «рисовать» квадрат, манипулятор примет положение HOME. HOME – это структурная переменная типа AXIS**,** которая хранит в себе углы поворота каждой оси. В дальнейшем это положение будем считать начальным. Инициализация скоростей и ускорений производится функцией BAS(•), в качестве параметра используется #INITMOV – это означает, что все скорости и ускорения примут максимальные значения.

```
DEF TEST_1()
   EXT BAS (BAS_COMMAND :IN,REAL :IN) ;Описание BAS()
   DECL AXIS HOME
   INT I
   INT FL

BAS (#INITMOV,0) ;Инициализацияскоростей

HOME={AXIS: A1 0,A2 -90,A3 90,A4 0,A5 30,A6 0}
FL = 1
PTP HOME ;Перемещаемся в начальное положение

WAIT SEC 2

   LOOP
      LIN {X 500,Y -100,Z 500,A 0,B 0,C 0}
      WAIT SEC 1
      LIN {X 500,Y 100,Z 500,A 0,B 0,C 0}
      WAIT SEC 1
      LIN {X 500,Y 100,Z 700,A 0,B 0,C 0}
      WAIT SEC 1
      LIN {X 500,Y -100,Z 700,A 0,B 0,C 0}
      WAIT SEC 1
      IF FL > 6 THEN
         EXIT
```

```
      ELSE
         FL = FL + 1 ;Увеличиваем флаг на 1
      ENDIF
   ENDLOOP
END
```

При линейном движении скорость задается с помощью системной переменной $VEL.CP, а ускорение – $ACC.CP.

Скорость указывается в м/с (стандартное значение 2.0000), а ускорение в м/с^2 (стандартное значение 2.300000).

В следующем примере рабочий орган манипулятора «рисует» синусоиду в плоскости YOZ (скорость движения манипулятора 0.2 м/с).

```
DEF test_3()
   EXT BAS (BAS_COMMAND :IN,REAL :IN)
   DECL AXIS HOME
   BAS (#INITMOV,0)
   HOME={AXIS: A1 0,A2 -90,A3 90,A4 0,A5 0,A6 0}
   $APO.CVEL=100
   $VEL.CP=0.2 ;Скорость линейного движения
   PTP HOME ;BCO run
   PTP {X 500,Y 0,Z 700,A 0,B 90,C 0}
   LIN {X 500,Y 100,Z 800} C_VEL
   LIN {X 500,Y 300,Z 600} C_VEL
   LIN {X 500,Y 500,Z 800} C_VEL
   LIN {X 500,Y 700,Z 600} C_VEL
   LIN {X 500,Y 900,Z 800} C_VEL
   PTP HOME
END
```

После выполнения данного кода получим траекторию движения, напоминающую график функции синус – см. рис. 2.2.2.

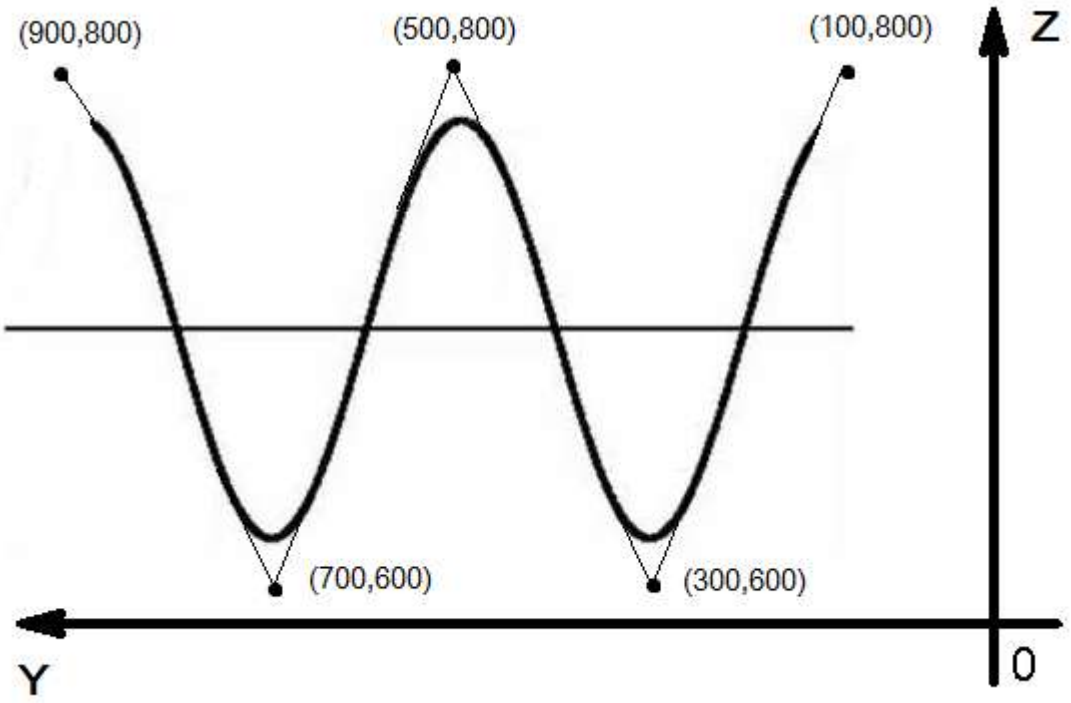


Рис. 2.2.2 Траектория движения рабочего органа манипулятора

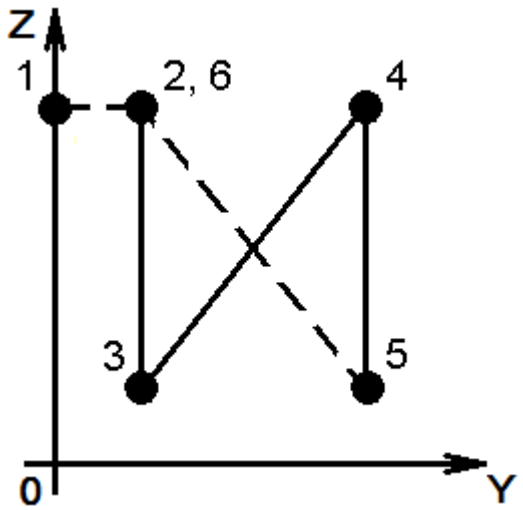
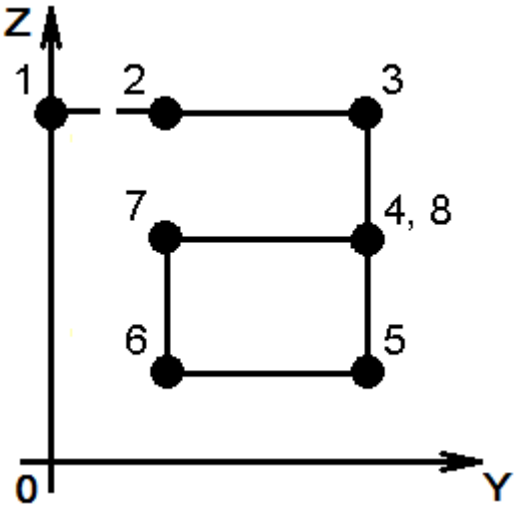
В данном примере использовался параметр C_VEL. C_VEL позволяет скруглить линию, идущую от одной точки к другой. Максимальное значение этого параметра равно 100. Для его задания нужно обратиться к элементу CVEL системной структуры $APO:

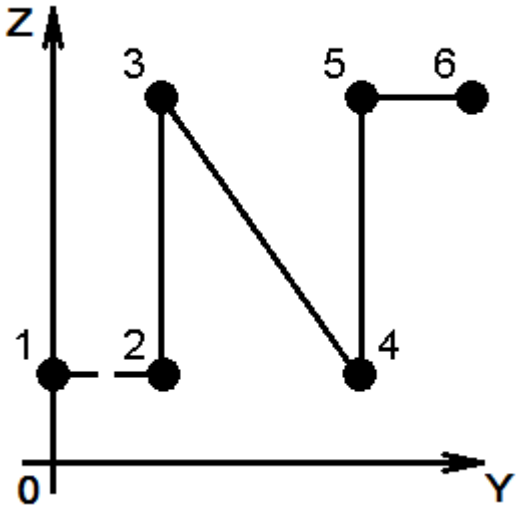
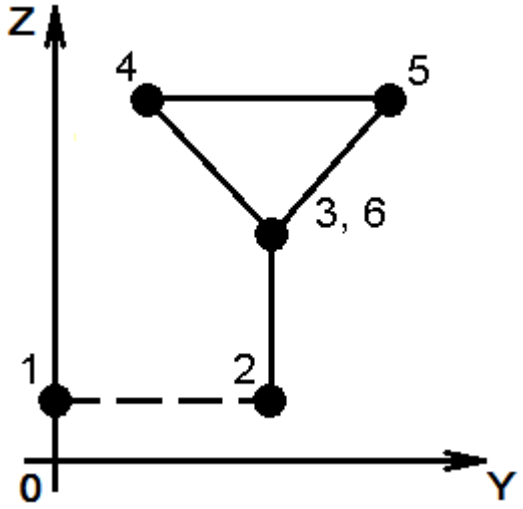
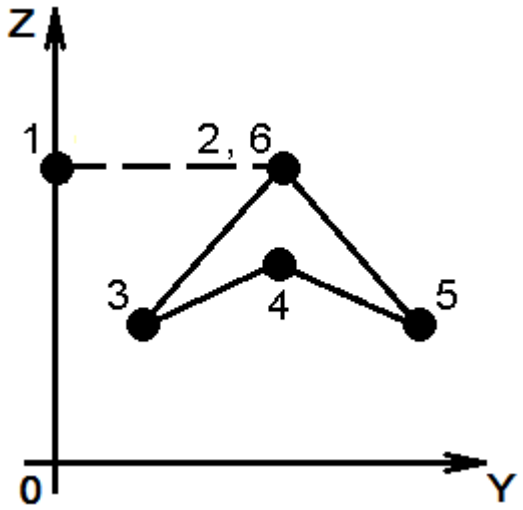
```
$APO.CVEL=100
```

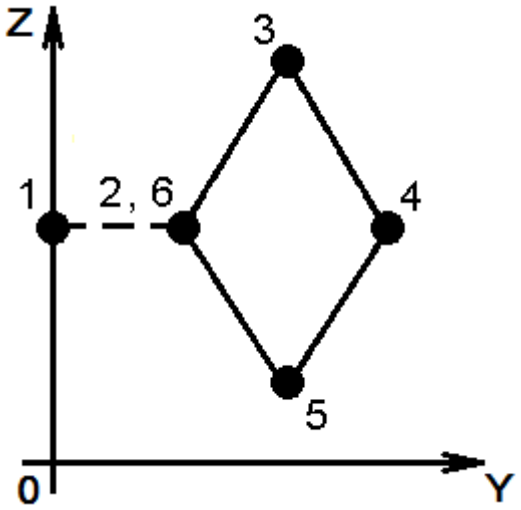
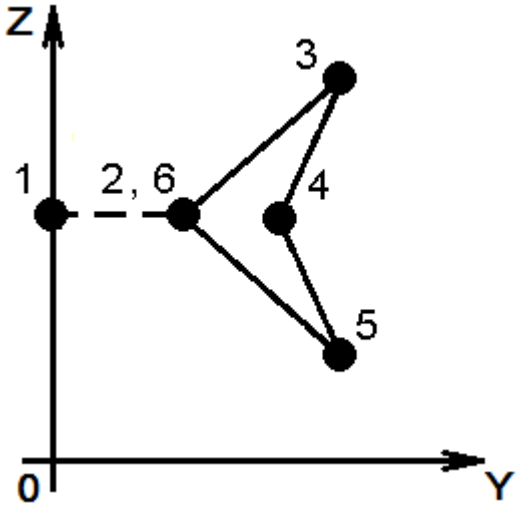
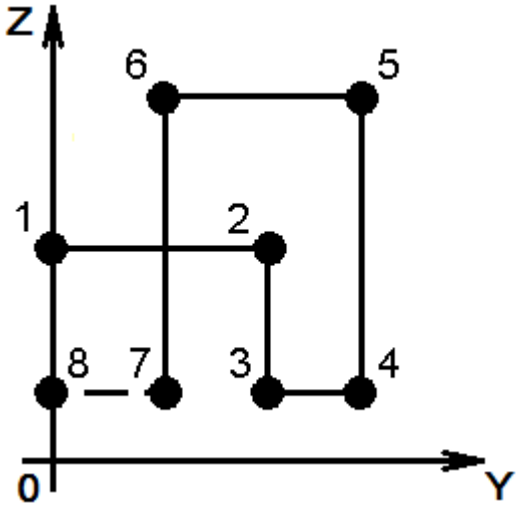
Задание: написать программу на языке KRL, задающую движение рабочего органа манипулятора по траектории в соответствии с номером варианта задания (таблица 2.2.1) и использующую вызовы функций PTP и LIN. Штриховыми линиями на рисунках траекторий заданий обозначены вызовы функций PTP,

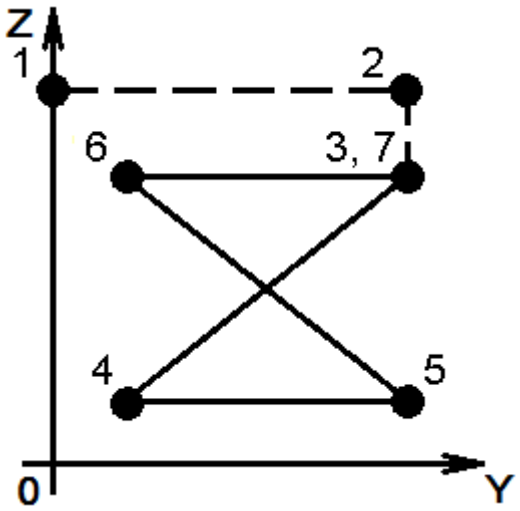
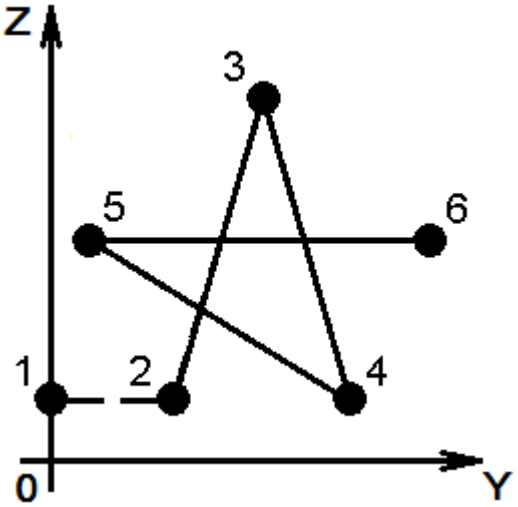
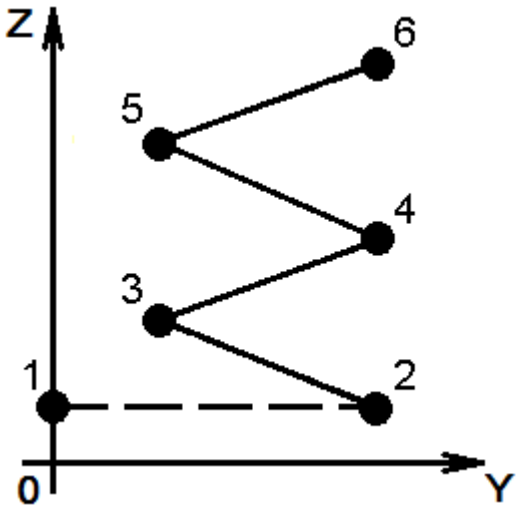
сплошными линиями – вызовы функций LIN. При цикличном движении рабочего органа по траектории после достижения её последней точки выполняется вызов функции WAIT, затем происходит переход в первую точку, используя вызов функции PTP. Цикличное движение реализуется циклом с параметром FOR.

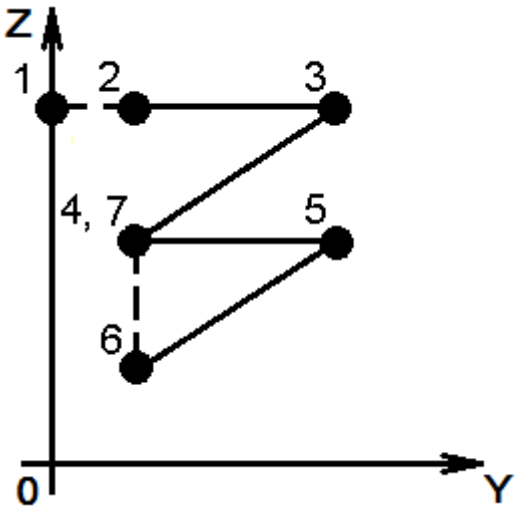
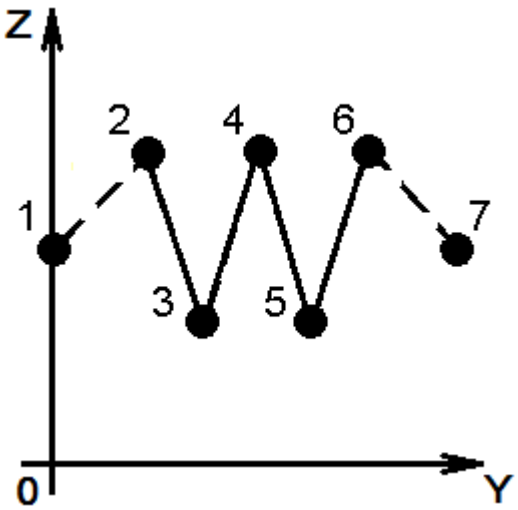
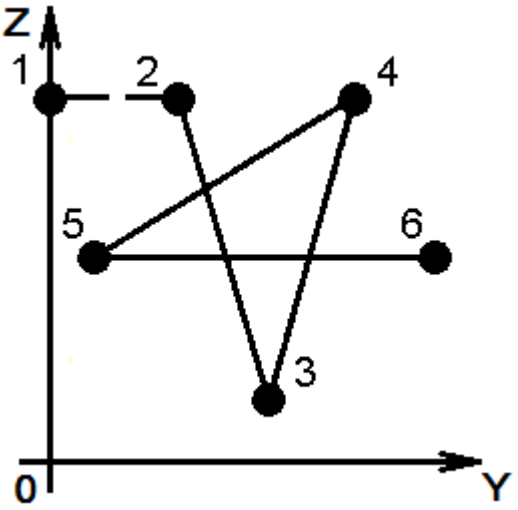
Таблица 2.2.1 – Варианты заданий

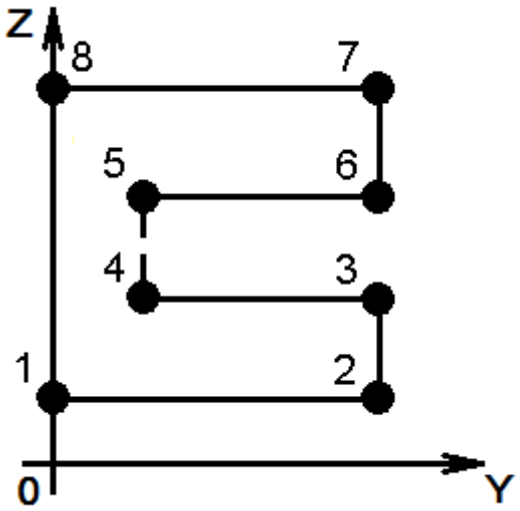
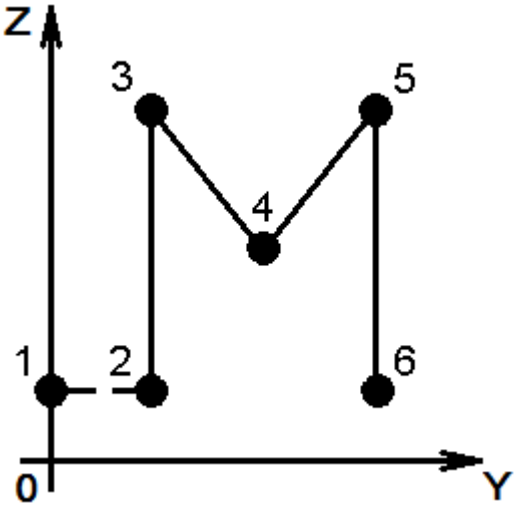
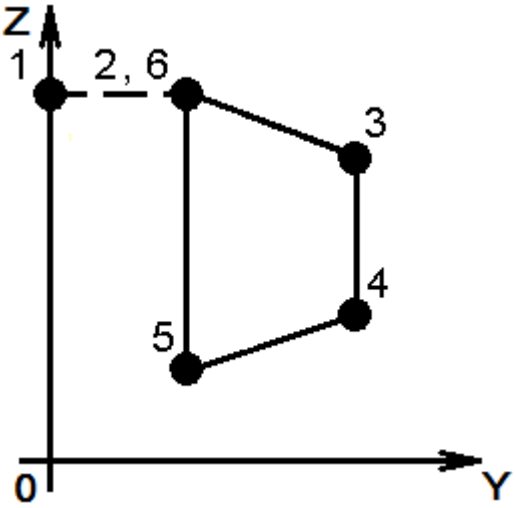
№ вар.	Форма траектории	Параметры
1		значения скоростей всех осей равны 1%, значения ускорений всех осей равны 1%, движение по траектории выполняется циклично 3 раза, с паузами по 2 сек. между циклами
2		значения скоростей всех осей равны 1%, значения ускорений всех осей равны 10%, движение по траектории выполняется циклично 4 раза, с паузами по 1 сек. между циклами

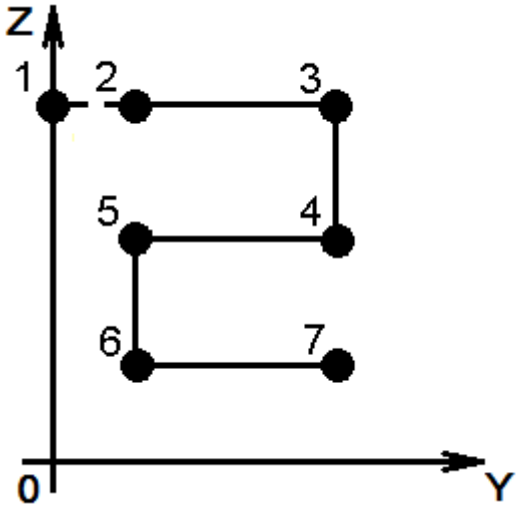
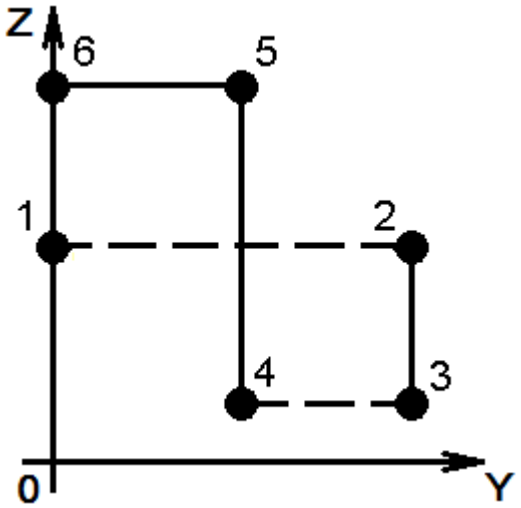
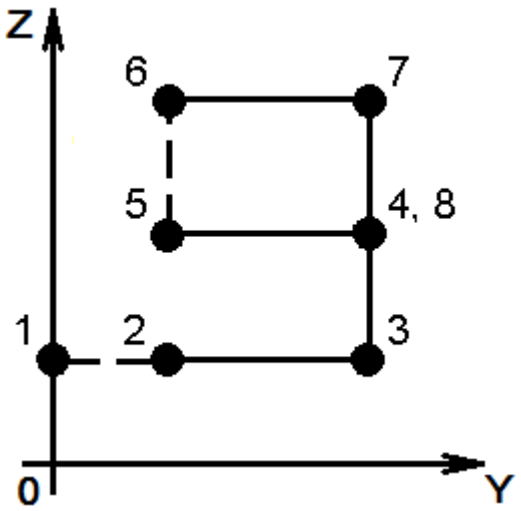
№ вар.	Форма траектории	Параметры
3		значения скоростей всех осей равны 1%, значения ускорений всех осей равны 2%, движение по траектории выполняется циклично 4 раза, с паузами по 2 сек. между циклами
4		значения скоростей всех осей равны 5%, значения ускорений всех осей равны 1%, движение по траектории выполняется циклично 4 раза, с паузами по 3 сек. между циклами
5		значения скоростей всех осей равны 1%, значения ускорений всех осей равны 1%, движение по траектории выполняется циклично 3 раза, с паузами по 1 сек. между циклами

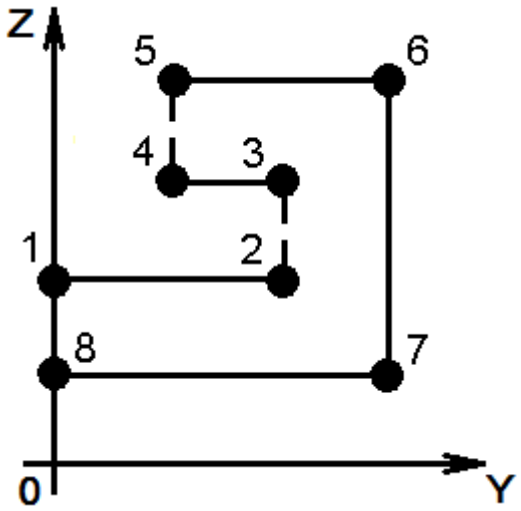
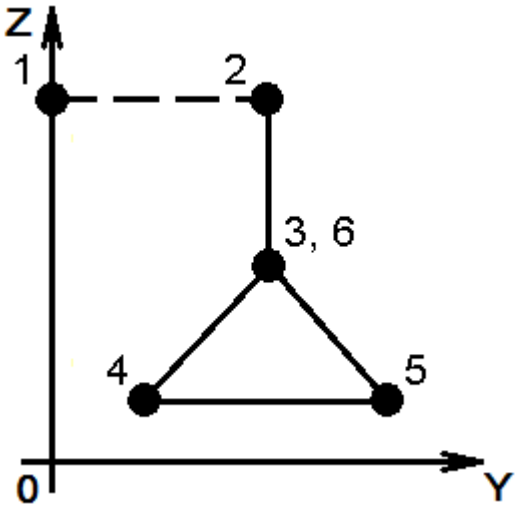
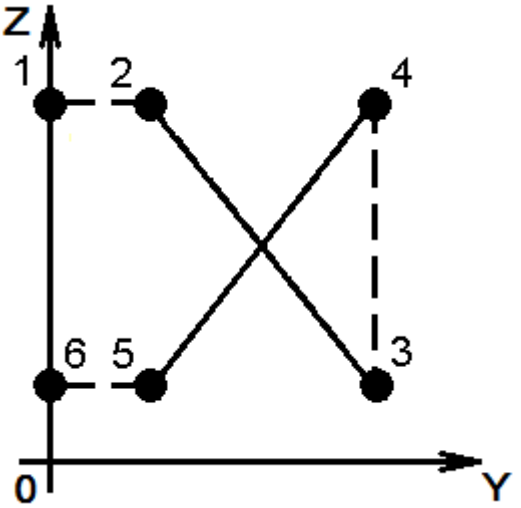
№ вар.	Форма траектории	Параметры
6		значения скоростей всех осей равны 5%, значения ускорений всех осей равны 5%, движение по траектории выполняется циклично 2 раза, с паузами по 1 сек. между циклами
7		значения скоростей всех осей равны 2%, значения ускорений всех осей равны 2%, движение по траектории выполняется циклично 3 раза, с паузами по 2 сек. между циклами
8		значения скоростей всех осей равны 5%, значения ускорений всех осей равны 5%, движение по траектории выполняется циклично 2 раза, с паузами по 2 сек. между циклами

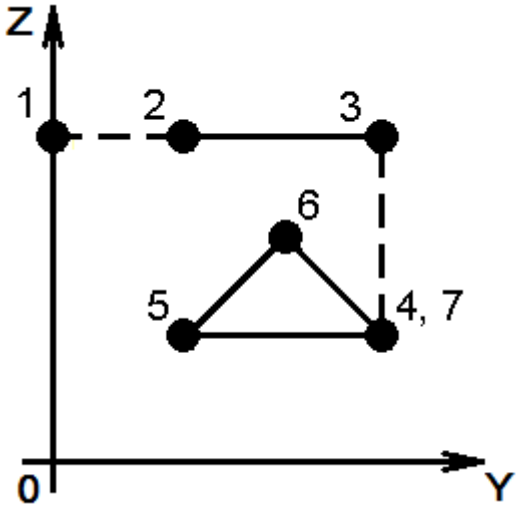
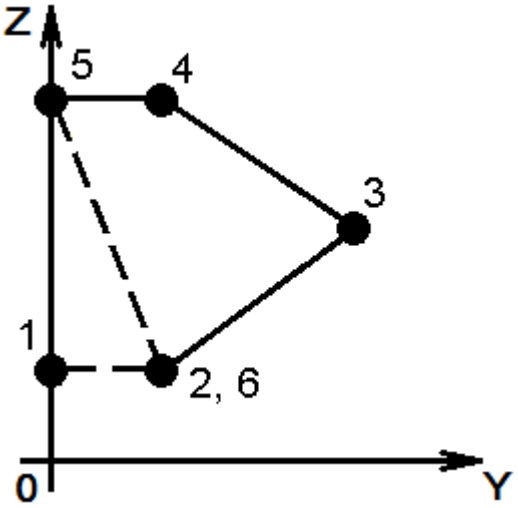
№ вар.	Форма траектории	Параметры
9		значения скоростей всех осей равны 1%, значения ускорений всех осей равны 1%, движение по траектории выполняется циклично 3 раза, с паузами по 1 сек. между циклами
10		значения скоростей всех осей равны 1%, значения ускорений всех осей равны 1%, движение по траектории выполняется циклично 2 раза, с паузами по 2 сек. между циклами
11		значения скоростей всех осей равны 2%, значения ускорений всех осей равны 2%, движение по траектории выполняется циклично 4 раза, с паузами по 1 сек. между циклами

№ вар.	Форма траектории	Параметры
12		значения скоростей всех осей равны 1%, значения ускорений всех осей равны 1%, движение по траектории выполняется циклично 4 раза, с паузами по 3 сек. между циклами
13		значения скоростей всех осей равны 2%, значения ускорений всех осей равны 1%, движение по траектории выполняется циклично 3 раза, с паузами по 5 сек. между циклами
14		значения скоростей всех осей равны 1%, значения ускорений всех осей равны 1%, движение по траектории выполняется циклично 2 раза, с паузами по 1 сек. между циклами

№ вар.	Форма траектории	Параметры
15		значения скоростей всех осей равны 2%, значения ускорений всех осей равны 2%, движение по траектории выполняется циклично 3 раза, с паузами по 1 сек. между циклами
16		значения скоростей всех осей равны 1%, значения ускорений всех осей равны 1%, движение по траектории выполняется циклично 4 раза, с паузами по 3 сек. между циклами
17		значения скоростей всех осей равны 1%, значения ускорений всех осей равны 1%, движение по траектории выполняется циклично 3 раза, с паузами по 3 сек. между циклами

№ вар.	Форма траектории	Параметры
18		значения скоростей всех осей равны 10%, значения ускорений всех осей равны 10%, движение по траектории выполняется циклично 4 раза, с паузами по 1 сек. между циклами
19		значения скоростей всех осей равны 2%, значения ускорений всех осей равны 2%, движение по траектории выполняется циклично 3 раза, с паузами по 2 сек. между циклами
20		значения скоростей всех осей равны 1%, значения ускорений всех осей равны 1%, движение по траектории выполняется циклично 4 раза, с паузами по 3 сек. между циклами

№ вар.	Форма траектории	Параметры
21		значения скоростей всех осей равны 1%, значения ускорений всех осей равны 1%, движение по траектории выполняется циклично 3 раза, с паузами по 2 сек. между циклами
22		значения скоростей всех осей равны 10%, значения ускорений всех осей равны 10%, движение по траектории выполняется циклично 4 раза, с паузами по 1 сек. между циклами
23		значения скоростей всех осей равны 2%, значения ускорений всех осей равны 2%, движение по траектории выполняется циклично 4 раза, с паузами по 1 сек. между циклами

№ вар.	Форма траектории	Параметры
24		значения скоростей всех осей равны 1%, значения ускорений всех осей равны 1%, движение по траектории выполняется циклично 4 раза, с паузами по 3 сек. между циклами
25		значения скоростей всех осей равны 1%, значения ускорений всех осей равны 1%, движение по траектории выполняется циклично 3 раза, с паузами по 1 сек. между циклами

2.3 Круговое движение манипулятора

Цель лабораторной работы – написать программу на языке KRL, задающую движение манипулятора и использующую функции CIRC.

Чтобы выполнить движение рабочего органа манипулятора по окружности, эллипсу или дуге, используют функцию CIRC**.** Значения скорости и ускорения задаются аналогично функции LIN**.**

Вызов функции CIRC немного схож с вызовами функций PTP и LIN за исключением того, что в функцию передаются координаты двух точек. Первая точка «вспомогательная», а вторая «конечная». Также, после этих двух точек, задается параметр CA**,** который характеризует угол дуги (см. рис. 2.3.1).

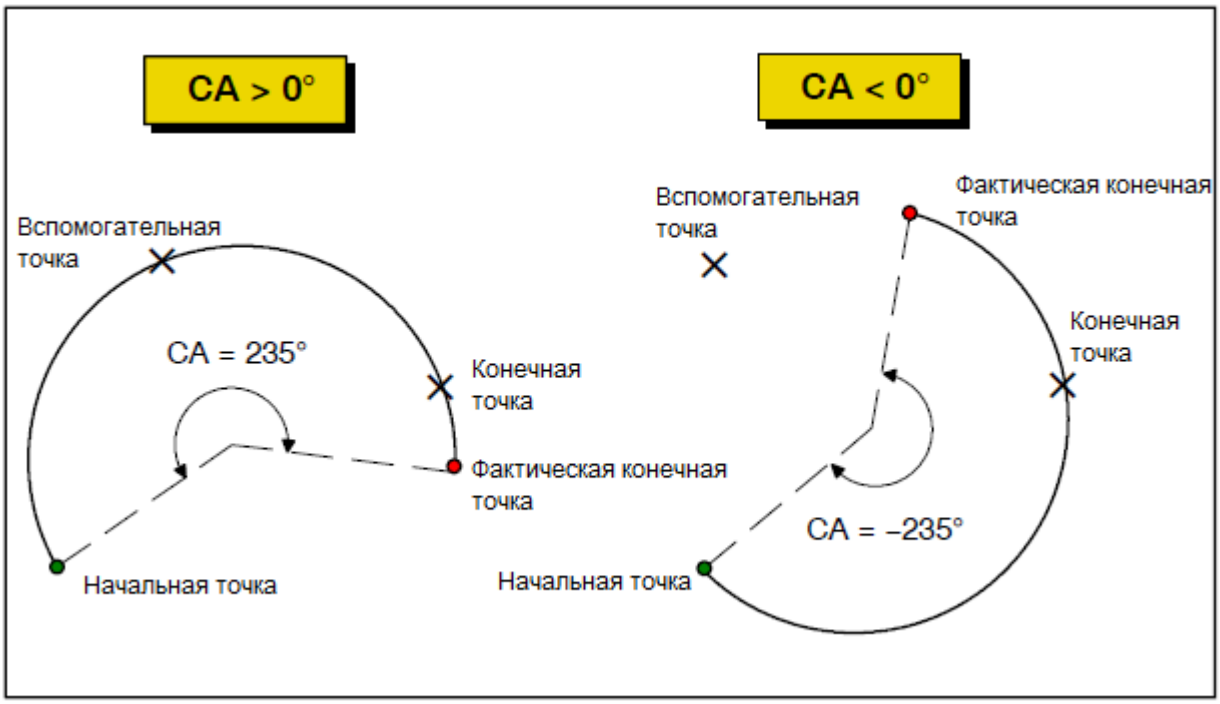


Рис. 2.3.1 Задание длины дуги окружности с помощью параметра CA

Начальная точка дуги задается выполнением функции LIN или PTP, а остальные точки – функцией CIRC**.**

Например, «нарисуем» окружность диаметром 500 в плоскости YOZ рабочим органом манипулятора:

```
DEF test_2 ()

   EXT BAS (BAS_COMMAND :IN,REAL :IN)
   DECL AXIS HOME
   INT I
   $BASE = $WORLD
   $TOOL = $NULLFRAME
   BAS (#INITMOV,0)
   HOME={AXIS: A1 0,A2 -90,A3 90,A4 0,A5 0,A6 0}

   FOR I=1 TO 6
      $VEL_AXIS[I] = 100 ;Скорости всех осей 100%
   ENDFOR
   $VEL.CP=0.5
   PTP HOME
   PTP {X 500,Y 250,Z 1000,A 0,B 90,C 0} ;Начальная точка

   LOOP
     CIRC {X 500,Y 0,Z 750},{X 500, Y 250, Z 500}, CA 360
   ENDLOOP

END
```

При выполнении данного кода мы получим траекторию движения рабочего органа в виде окружности, изображенную на рис. 2.3.2.

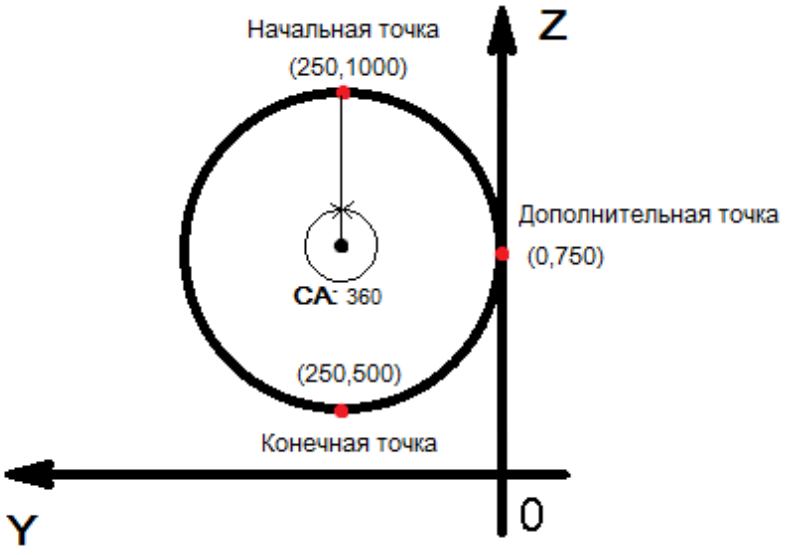


Рис. 2.3.2 Траектория движения рабочего органа манипулятора

Чтобы «нарисовать» рабочим органом манипулятора овал Кассини, нужно использовать C_VEL. Следующий пример демонстрирует использование параметра C_VEL совместно с функцией CIRC – рис. 2.3.3.

```
DEF test_4()
EXT BAS (BAS_COMMAND :IN,REAL :IN)
DECL AXIS HOME
BAS (#INITMOV,0) ;Initialization of velocities
HOME={AXIS: A1 0,A2 -90,A3 90,A4 0,A5 0,A6 0}
$APO.CVEL=100
PTP HOME ;BCO run
PTP {X 500,Y 0,Z 800,A 0,B 90,C 0}

LOOP
   CIRC {X 500,Y 100,Z 700},{X 500,Y 200,Z 800}, CA 180
    C_VEL
   CIRC {X 500,Y 300,Z 700},{X 500,Y 400,Z 800}, CA 180
```

```
   CIRC {X 500,Y 300,Z 900},{X 500,Y 200,Z 800}, CA 180
    C_VEL
   CIRC {X 500,Y 100,Z 900},{X 500,Y 0,Z 800}, CA 180
   WAIT SEC 4
ENDLOOP
END
```

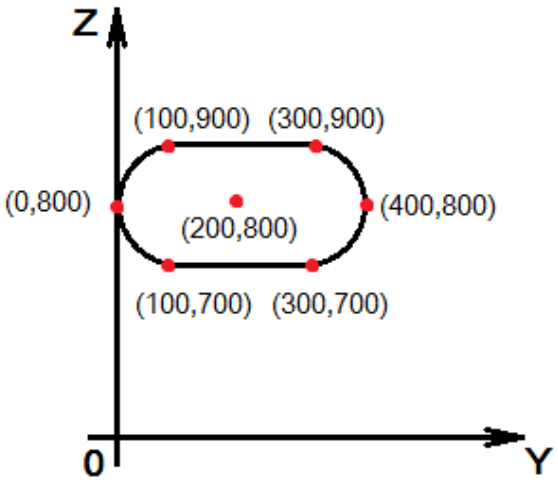
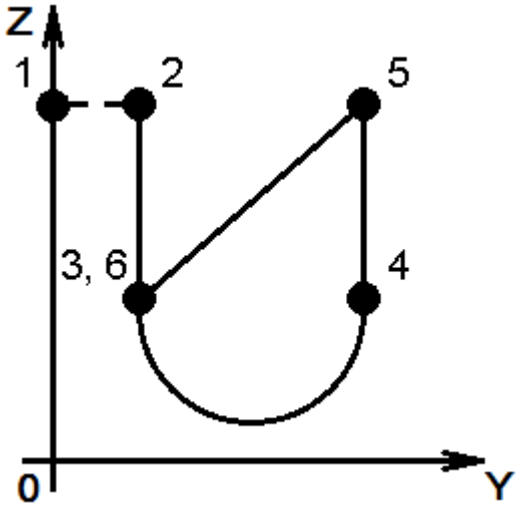
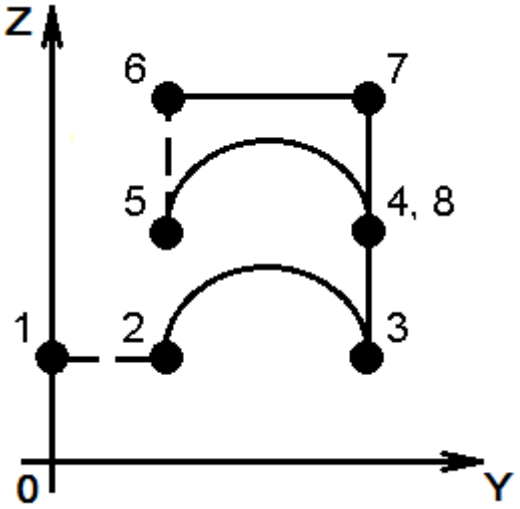
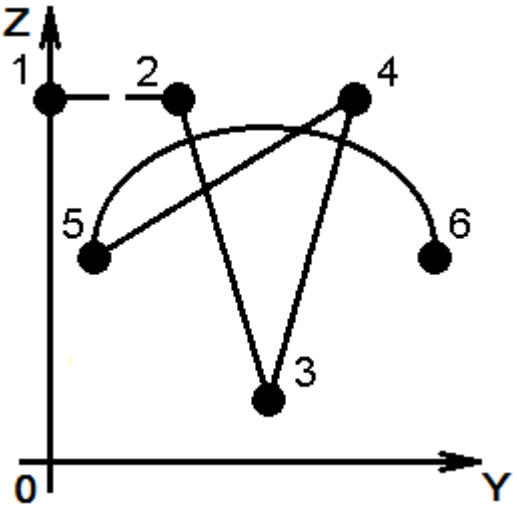


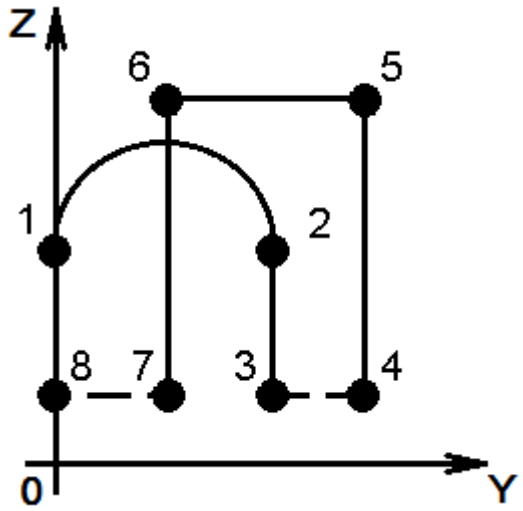
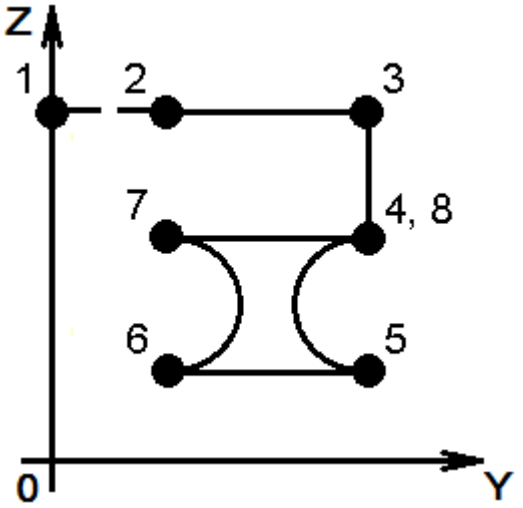
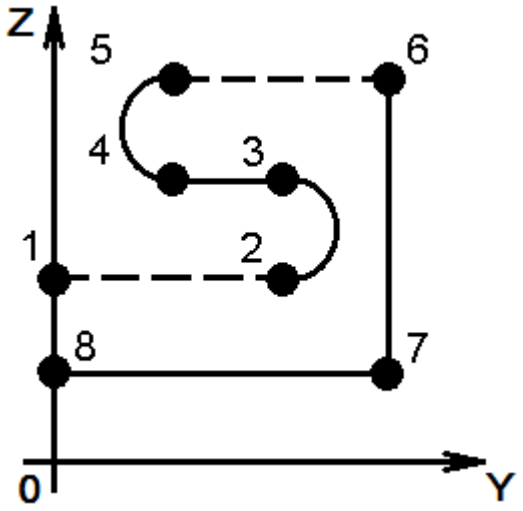
Рис 2.3.3 Траектория движения рабочего органа

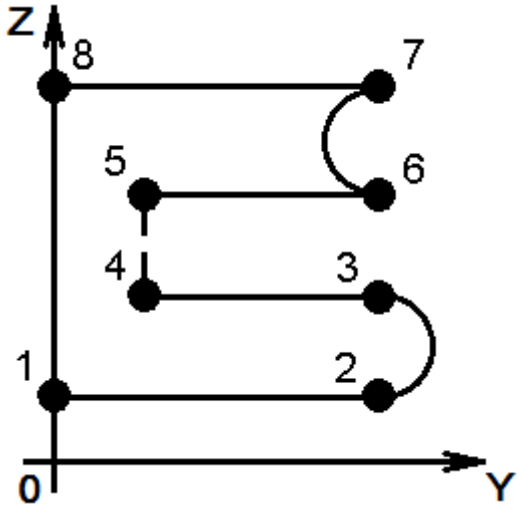
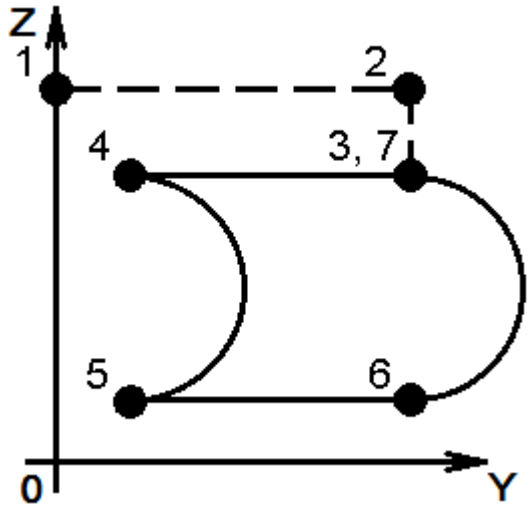
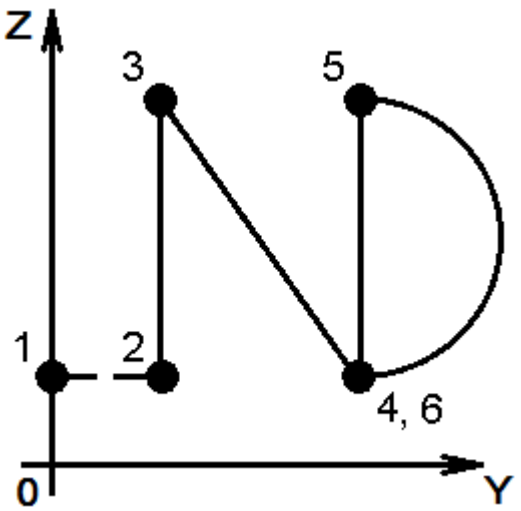
Чтобы провести прямые линии между двумя окружностями, был использован параметр `C_VEL`.

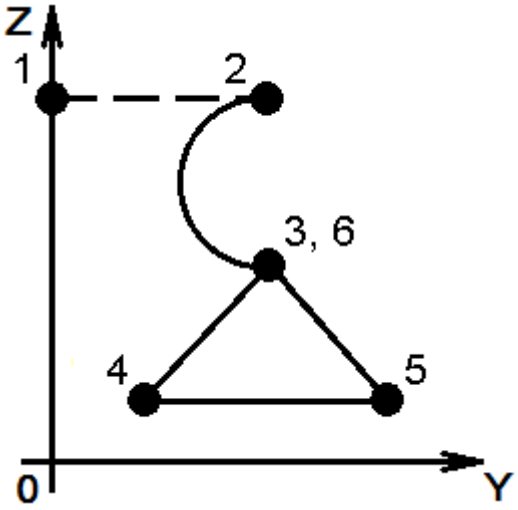
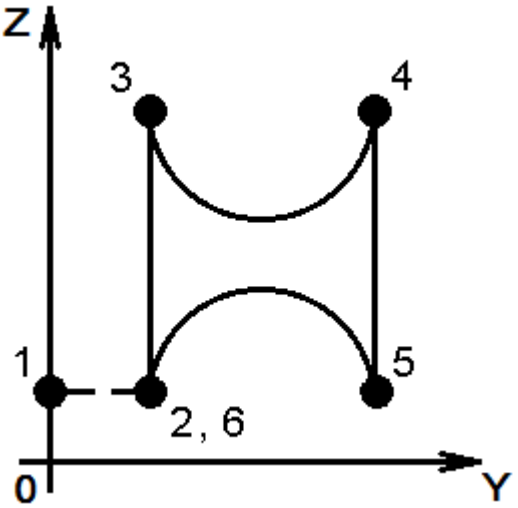
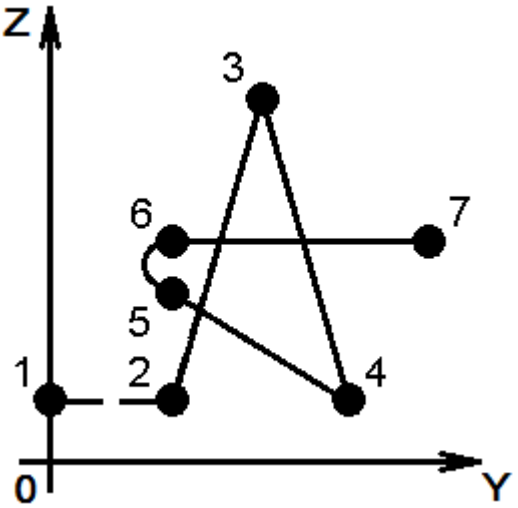
Задание: написать программу на языке KRL, задающую движение рабочего органа манипулятора по траектории в соответствии с номером варианта задания (таблица 2.3.1) и использующую вызовы функций PTP, LIN и CIRC. Штриховыми линиями на рисунках траекторий заданий обозначены вызовы функций PTP, сплошными линиями – вызовы функций LIN и CIRC. При цикличном движении рабочего органа по траектории после достижения её последней точки выполняется вызов функции WAIT, затем происходит переход в первую точку, используя вызов функции PTP. Цикличное движение реализуется циклом с параметром FOR.

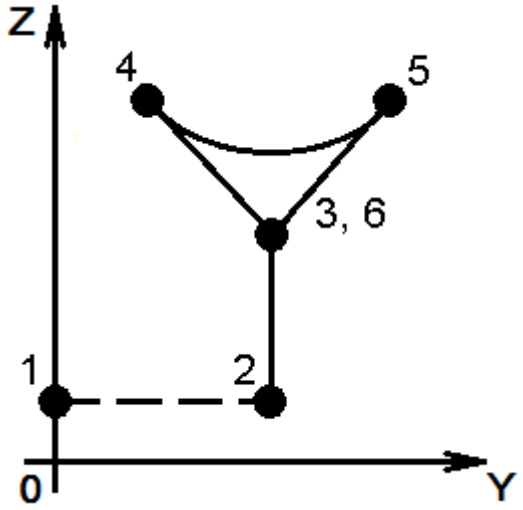
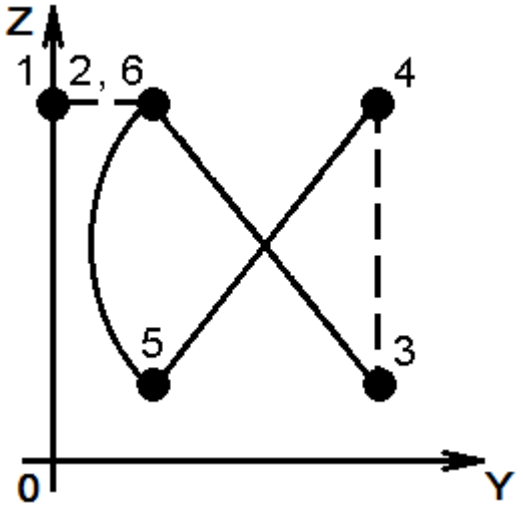
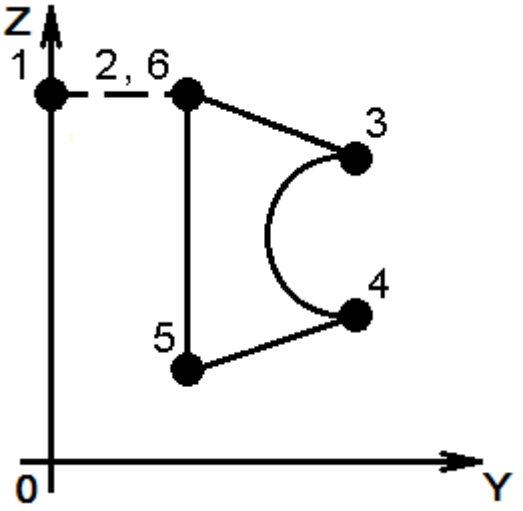
Таблица 2.3.1 – Варианты заданий

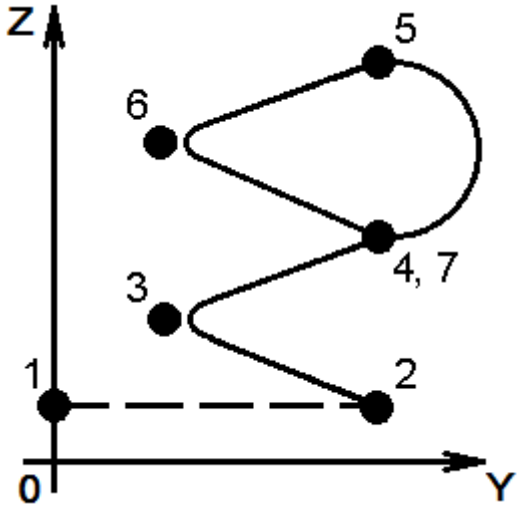
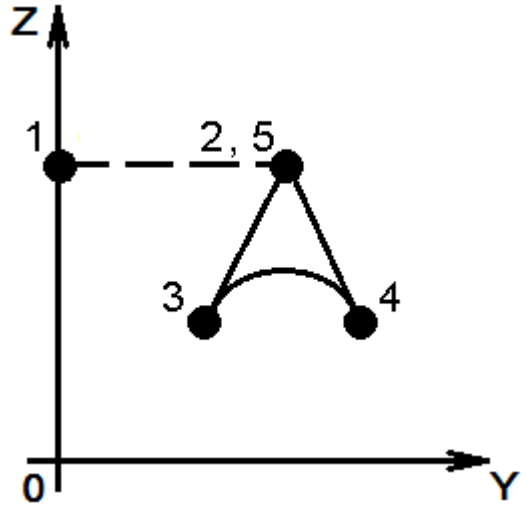
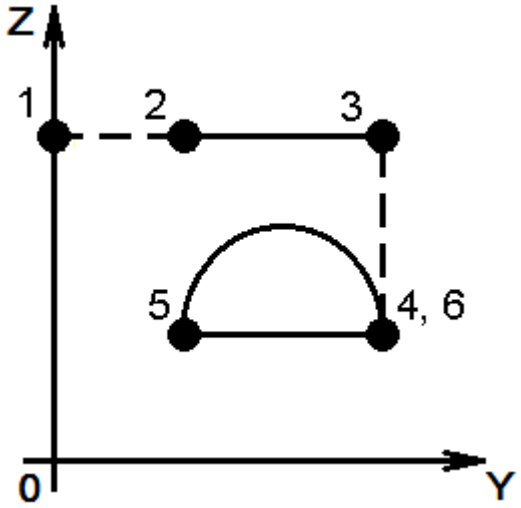
№ вар.	Форма траектории	Параметры
1		значения скоростей всех осей равны 2%, значения ускорений всех осей равны 2%, движение по траектории выполняется циклично 3 раза, с паузами по 1 сек. между циклами
2		значения скоростей всех осей равны 1%, значения ускорений всех осей равны 1%, движение по траектории выполняется циклично 4 раза, с паузами по 3 сек. между циклами
3		значения скоростей всех осей равны 1%, значения ускорений всех осей равны 1%, движение по траектории выполняется циклично 3 раза, с паузами по 3 сек. между циклами

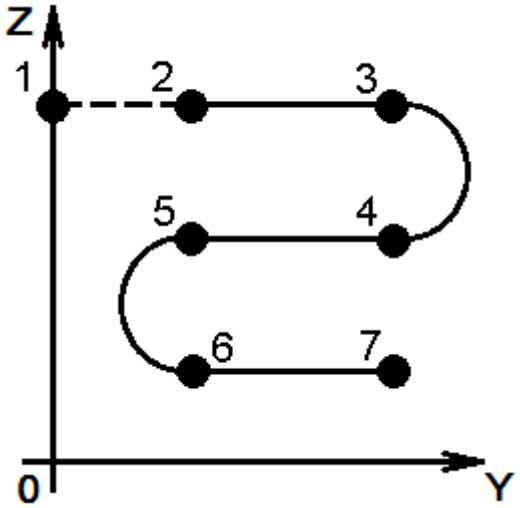
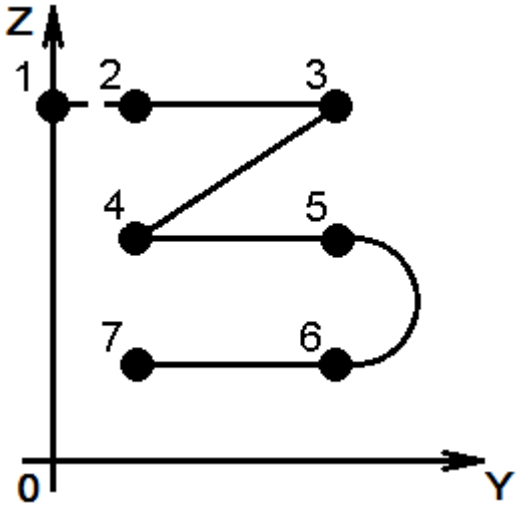
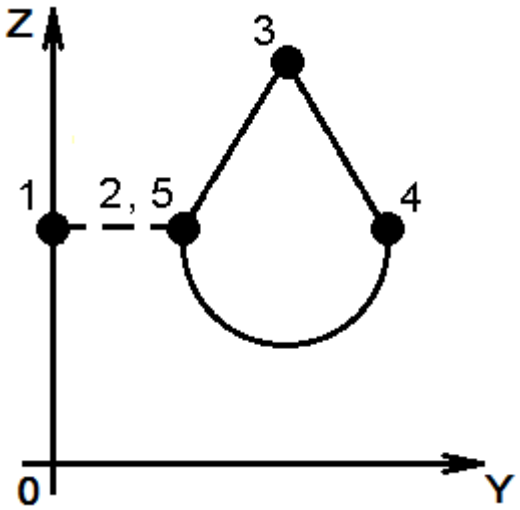
№ вар.	Форма траектории	Параметры
4		значения скоростей всех осей равны 10%, значения ускорений всех осей равны 10%, движение по траектории выполняется циклично 4 раза, с паузами по 1 сек. между циклами
5		значения скоростей всех осей равны 2%, значения ускорений всех осей равны 2%, движение по траектории выполняется циклично 3 раза, с паузами по 2 сек. между циклами
6		значения скоростей всех осей равны 1%, значения ускорений всех осей равны 1%, движение по траектории выполняется циклично 4 раза, с паузами по 3 сек. между циклами

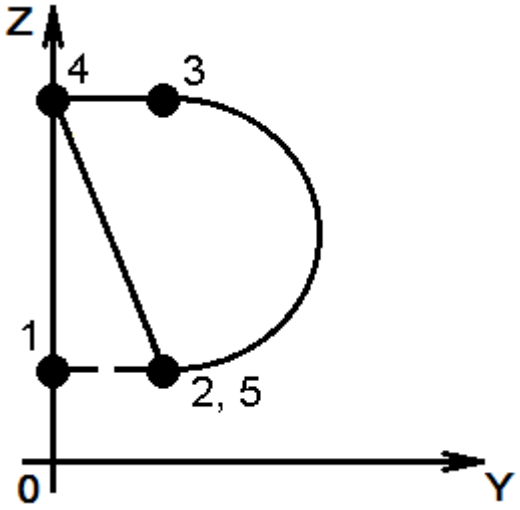
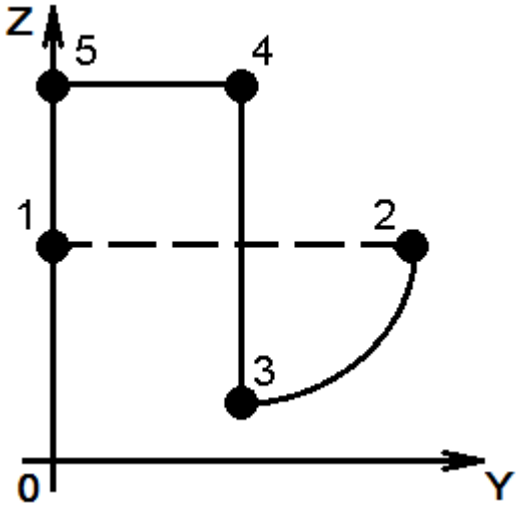
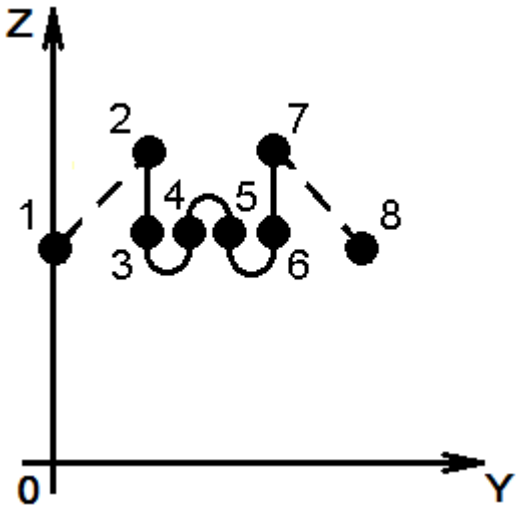
№ вар.	Форма траектории	Параметры
7		значения скоростей всех осей равны 1%, значения ускорений всех осей равны 1%, движение по траектории выполняется циклично 3 раза, с паузами по 2 сек. между циклами
8		значения скоростей всех осей равны 10%, значения ускорений всех осей равны 10%, движение по траектории выполняется циклично 4 раза, с паузами по 1 сек. между циклами
9		значения скоростей всех осей равны 2%, значения ускорений всех осей равны 2%, движение по траектории выполняется циклично 4 раза, с паузами по 1 сек. между циклами

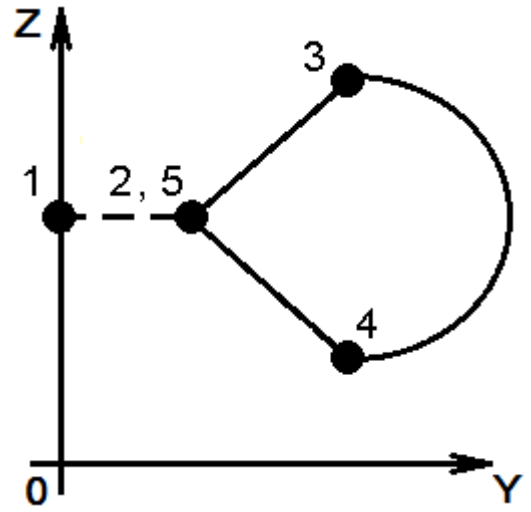
№ вар.	Форма траектории	Параметры
10		значения скоростей всех осей равны 1%, значения ускорений всех осей равны 1%, движение по траектории выполняется циклично 4 раза, с паузами по 3 сек. между циклами
11		значения скоростей всех осей равны 1%, значения ускорений всех осей равны 1%, движение по траектории выполняется циклично 3 раза, с паузами по 2 сек. между циклами
12		значения скоростей всех осей равны 1%, значения ускорений всех осей равны 10%, движение по траектории выполняется циклично 4 раза, с паузами по 1 сек. между циклами

№ вар.	Форма траектории	Параметры
13		значения скоростей всех осей равны 1%, значения ускорений всех осей равны 2%, движение по траектории выполняется циклично 4 раза, с паузами по 2 сек. между циклами
14		значения скоростей всех осей равны 5%, значения ускорений всех осей равны 1%, движение по траектории выполняется циклично 4 раза, с паузами по 3 сек. между циклами
15		значения скоростей всех осей равны 1%, значения ускорений всех осей равны 1%, движение по траектории выполняется циклично 3 раза, с паузами по 1 сек. между циклами

№ вар.	Форма траектории	Параметры
16		значения скоростей всех осей равны 5%, значения ускорений всех осей равны 5%, движение по траектории выполняется циклично 2 раза, с паузами по 1 сек. между циклами
17		значения скоростей всех осей равны 2%, значения ускорений всех осей равны 2%, движение по траектории выполняется циклично 3 раза, с паузами по 2 сек. между циклами
18		значения скоростей всех осей равны 5%, значения ускорений всех осей равны 5%, движение по траектории выполняется циклично 2 раза, с паузами по 2 сек. между циклами

№ вар.	Форма траектории	Параметры
19		значения скоростей всех осей равны 1%, значения ускорений всех осей равны 1%, движение по траектории выполняется циклично 3 раза, с паузами по 1 сек. между циклами
20		значения скоростей всех осей равны 1%, значения ускорений всех осей равны 1%, движение по траектории выполняется циклично 2 раза, с паузами по 2 сек. между циклами
21		значения скоростей всех осей равны 2%, значения ускорений всех осей равны 2%, движение по траектории выполняется циклично 4 раза, с паузами по 1 сек. между циклами

№ вар.	Форма траектории	Параметры
22		значения скоростей всех осей равны 1%, значения ускорений всех осей равны 1%, движение по траектории выполняется циклично 4 раза, с паузами по 3 сек. между циклами
23		значения скоростей всех осей равны 2%, значения ускорений всех осей равны 1%, движение по траектории выполняется циклично 3 раза, с паузами по 5 сек. между циклами
24		значения скоростей всех осей равны 1%, значения ускорений всех осей равны 1%, движение по траектории выполняется циклично 2 раза, с паузами по 1 сек. между циклами

№ вар.	Форма траектории	Параметры
25		значения скоростей всех осей равны 1%, значения ускорений всех осей равны 3%, движение по траектории выполняется циклично 4 раза, с паузами по 3 сек. между циклами

Список литературы

1) Интернет ресурс. РОБОТОТЕХНИКА И АВТОМАТИЗАЦИЯ http://www.kuka-robotics.com/russia/ru
2) Интернетресурс: KUKA Industrial Robot http://www.kuka-industries.com/de/career/
3) Интернет ресурс: WIKIKUKA. https://en.wikipedia.org/wiki/KUKA