УДК 621.9.06-529.001.2

Г. М. Мартинов, д-р техн. наук, проф.,
А. И. Обухов, Р. Л. Пушков
МГУ "СТАНКИН"
pushkov@ncsystems.ru

Принцип построения универсального интерпретатора языка программирования высокого уровня для систем ЧПУ*

Проанализирована проблема создания интерпретатора структурированного языка высокого уровня по типу препроцессора с учетом специфики его использования для создания управляющих программ системы ЧПУ. Предложен способ интеграции препроцессора в архитектурную модель системы ЧПУ.

Ключевые слова: *система ЧПУ, интерпретатор, компилятор, синтаксический анализ, препроцессор, язык программирования высокого уровня, управляющая программа, стандартный цикл, язык диалога для создания управляющей программы, открытость, переменные, функция, оператор, макрос, условие, подпрограмма*

Понятие об интерпретаторе системы ЧПУ

В течение нескольких десятилетий стандартом языка описания траектории движения инструмента и команд управления электроавтоматикой является язык ISO-7bit (ISO 6983). Управляющая программа на этом языке содержит некоторую последовательность команд, описывающих движение инструмента, и вспомогательные операции, исполняемые электроавтоматикой. Стандартный пример программы (иллюстрация траектории и текст) приведен на рис. 1.

Структура языка не предусматривает создания параметрических циклов. Повторное использование участков управляющей программы посредством вызова из другой управляющей программы достаточно ограничено, это часто решается посредством многократного повторения одних и тех же фрагментов программы. Стандарт STEP-NC был призван улучшить ситуацию [1], но до сих пор должного распространения не получил.

Разработчики системы ЧПУ стали включать в свои системы поддержку отдельных синтаксических конструкций, присущих алгоритмическим языкам высокого уровня. Другой подход предполагает альтернативное использование диалогового языка создания управляющей программы, например, языка для системы Heidenhain [2] (рис. 2).

Программирование на этом языке заключается не в написании строк управляющей программы в текстовом редакторе, а в формировании контура с помощью визуального редактора. Такой способ создания управляющей программы удобен для подготовки конечного контура, но требует наличия специальных навыков и мало применим для написания сложных параметрических программ и стандартных циклов.

Использование языка высокого уровня для создания управляющих программ системы ЧПУ позволяет: реализовывать сложные пользовательские циклы; сокращать размер управляющей программы; создавать параметризованные функции; разрабатывать управляющие программы для групповых технологий; расширять сервисные возможности (например, организовывать эффективный диалог с оператором); создавать специальные тестовые программы для проверки работоспособности приводов, входов/выходов электроавтоматики и системы в целом.

Отсутствие единого стандарта для языка управляющих программ высокого уровня приводит к тому, что производители систем ЧПУ используют собственные решения, часто несовместимые друг с другом.

Весьма актуальна задача построения универсального интерпретатора языка высокого уровня для систем ЧПУ, использующих в качестве входного языка ISO-7bit.

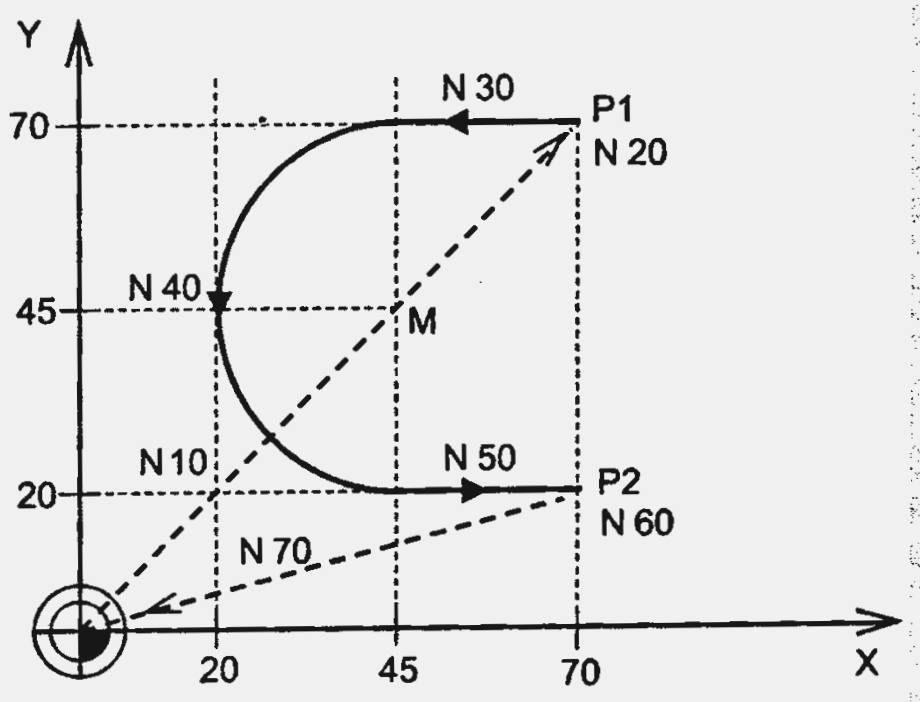
Предложен универсальный способ реализации интерпретатора языка высокого уровня, подходящий для большинства существующих систем ЧПУ. При этом язык должен быть максимально приближен к одному из имеющихся стандартных языков программирования и должен иметь возможность использования имеющихся файлов управляющих программ на языке ISO-7bit в структурированном коде высокого уровня.

Представление о структурированном языке высокого уровня как средстве повышения открытости систем ЧПУ

Структурированный язык позволяет: составлять программы из независимых фрагментов кода; вызывать подпрограммы; применять алгоритмические конструкции ветвления, организовывать циклы, команды условного и безусловного переходов; использовать системные, глобальные и локальные переменные [3]. Управляющую программу на структурированном языке представляют в виде сочетания главной программы и нескольких вызываемых подпрограмм, состоящих из набора одиночных или заключенных в блоки операторов.

Структура типичной программы на языке высокого уровня в виде формального грамматического описания в расширенной форме Бэкуса—Наура (РБНФ) [4] представлена на рис. 3. Программа состоит из набора функций, одна из которых явля-

* Работа выполнена по Госконтракту № П926 на проведение НИР в рамках ФЦП "Научные и научно-педагогические кадры инновационной России" на 2009—2013 годы.




```
N10 G00 G90 X70 Y70 Z10 S5000 M3 ;Инструмент ускоренно перемещается в P1
N20 G01 Z-5 F2000                ;Врезание по оси по Z с подачей 2000 мм/мин.
N30 G01 X45                      ;Линейная интерполяция к началу дуги
N40 G03 X45 Y20 I45 J45          ;Круговая интерполяция
N50 G01 X70                      ;Линейная интерполяция к точке P2
N60 G00 Z10                      ;Вывод инструмента в безопасную плоскость
N70 G00 X0 Y0 M5                 ;Останов шпинделя и возврат в начальную точку
N80 M30                          ;Конец программы
```

Рис. 1. Пример программы на языке ISO-7bit

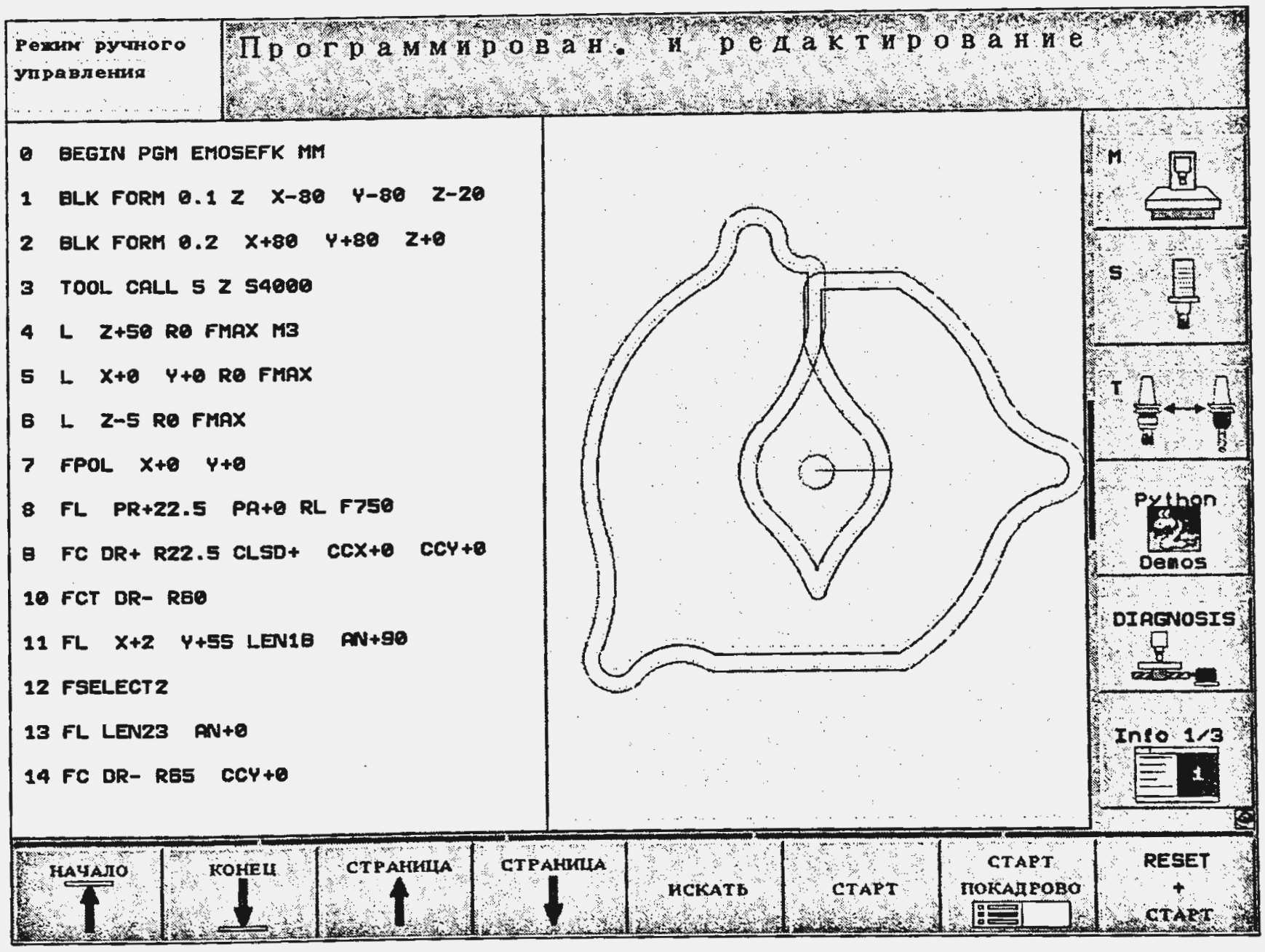


Рис. 2. Диалог программирования системы Heidenhain

```
; представление главной программы
<программа> = *(<подключение_библиотеки> / <описание_переменной> /
              <определение_константы> / <метка> / <оператор>)
<подключение_библиотеки> = "#use" <DQUOTE> <идентификатор> <DQUOTE> <CRLF>
<идентификатор> = 2<ALPHA> *(<ALPHA> / <DIGIT> / "_")
<описание_переменной> = <тип_данных> <идентификатор> ["=" <выражение>] *(","
              <идентификатор> ["=" <выражение>]) ";" <CRLF>
<выражение> = (<идентификатор> / <десятичное_число> / <строка> / <символ> /
              <вызов_функции>) *((<арифметическая_операция> /
              <логическая_операция>) ((<идентификатор> / <десятичное_число> /
              <строка> / <символ> / <вызов_функции>))
<условие> = (<идентификатор> / <десятичное_число> / <строка> / <символ> /
              <вызов_функции>) *(<логическая_операция> (<идентификатор> /
              <десятичное_число> / <строка> / <символ> / <вызов_функции>))
<десятичное_число> = ["-"] 1*<DIGIT> ["." 1*<DIGIT>]
<арифметическая_операция> = "+" / "-" / "*" / "/" / "|" / "&" "not" / "^" /
              "mod" / "div" / "%"
<логическая_операция> = ">" / "<" / "==" / "!=" / ">=" / "<=" / "||" / "&&" /
              "!"
<тип_данных> = "void" / "double" / "int" / "char" / "bool" / "string"
<символ> = "'" [<CHAR>] "'"
<строка> = <DQUOTE> *<CHAR> <DQUOTE>
<вызов_функции> = <идентификатор> "(" [<выражение>] *("," <выражение>) ")"
<определение_константы> = "#define" <идентификатор> <выражение> <CRLF>
<метка> = <идентификатор> ":" <CRLF>
<оператор> = (<цикл_for> / <цикл_while> / <ветвление> / <переход> /
              <присваивание> / <вызов_функции> / <кадр_ISO_7bit>) ";"
<присваивание> = <идентификатор> "=" <выражение>
<цикл_for> = "for" "(" <присваивание> ";" <условие> ";" <присваивание> ")"
              "{" *<оператор> "}"
<цикл_while> = "while" "(" <условие> ")" "{" *<оператор> "}"
<ветвление> = "if" "(" <условие> ")" "{" *<оператор> "}" ["else" "{"
              *<оператор> "}"]
<переход> = "goto" <идентификатор>
<кадр_ISO_7bit> = 1*(<слово_ISO_7bit> / <расширенное_слово_ISO_7bit>)
<слово_ISO_7bit> = <ALPHA> <десятичное_число>
<расширенное_слово_ISO_7bit> = 1*<ALPHA> [1*<десятичное_число>] "="
              <выражение>
; представление подключаемой библиотеки
<библиотека> = *(<подключение_библиотеки> / <определение_константы> /
              <описание_функции>)
<описание_функции> = <заголовок_функции> "{" *(<описание_переменной> /
              <метка> / <оператор>) "}"
<заголовок_функции> = <тип данных> <идентификатор> "(" [<тип данных>
              <идентификатор>] *("," <тип_данных> <идентификатор>) ")"
```

Рис. 3. Структура программы на языке высокого уровня в РБНФ

Основные характеристики языков высокого уровня

Характеристика	Применение
Системные, глобальные и локальные переменные	Использование именованных ячеек для хранения данных и проведения операций над ними
Макросы	Символьное имя, заменяемое при обработке препроцессором на последовательность инструкций управляющей программы
Набор операторов цикла, ветвлений и перехода	Конструкции языка, позволяющие изменять ход последовательного выполнения программы путем ветвления в зависимости от условия или многократного повторения части программы
Набор стандартных функций	Системные, математические, строковые и другие функции, которые используются как базовые при создании более сложных операций
Базовый набор размеров целых величин и типы с плавающей точкой двойной точности	Обеспечение точности на широком диапазоне станков, в том числе и с пикоточностями. Используются при создании более сложных типов данных, массивов и составных структур данных

ется главной. Выполнение программы начинается вызовом *главной функции*. Каждая функция имеет *блок* выполняемых операторов. Блок операторов — это последовательность выражений и *сложных операторов*. С точки зрения процесса выполнения программа представляет собой рекурсивную интерпретацию блоков операторов и выражений.

В качестве примера рассмотрим язык программирования высокого уровня для системы ЧПУ, разработанный на основе стандарта ANSI C [6] с учетом специфики применения на системе ЧПУ. Ключевые элементы этого языка сведены в таблицу.

Реализация интерпретатора структурированного языка высокого уровня

Одним из самых эффективных и простых способов реализации интерпретатора является его разработка на базе рекурсивного нисходящего синтаксического анализатора [7]. Схема работы такого анализатора определена логикой построения выражений структурированного языка.

Рассмотрим пример использования математических выражений в коде управляющей программы (рис. 4). Присваивание значения переменной X в кадре N20 является типичным выражением и вычисляется в результате последовательного выполнения операций, по порядку приоритета с учетом скобок: шаг 1) 3*a; шаг 2) +b; шаг 3) /2; шаг 4) + 1; шаг 5) присваивание значения переменной X.

Набор порождающих правил вычисления выражений можно представить схемой (рис. 5), где стрелками показано направление от элемента к его составу. Выражение представляет собой рекурсивно вычисляемую структуру. Сначала надо вычислить последовательно каждое слагаемое A, которое, в свою очередь, требует последовательного вычисления составляющих его множителей B, каждый из которых может также представлять собой выражение. Таким образом, вычисление выражения идет, образно говоря, по лестнице — от нижнего уровня (наиболее приоритетного) к верхнему, и так рекурсивно несколько раз, пока не будут вычислены все слагаемые. В эту схему включаются любые другие действия (например, операторы сравнения) с заданным приоритетом.

На рис. 6 представлена упрощенная схема работы анализатора выражений в рамках интерпретатора управляющей программы. Скругленными прямоугольниками представлены рекурсивно вызываемые функции анализатора. На первом шаге вызывается функция сложения и вычитания слагаемых

```
int a = 10, b = 20;
N10 G00 G90 X70 Y70 Z10 S5000 M3
N20 G01 X=(3*a + b)/2+1  Z-5 F2000
N30 M30
```

Рис. 4. Пример выражения в коде программы на языке ISO-7bit

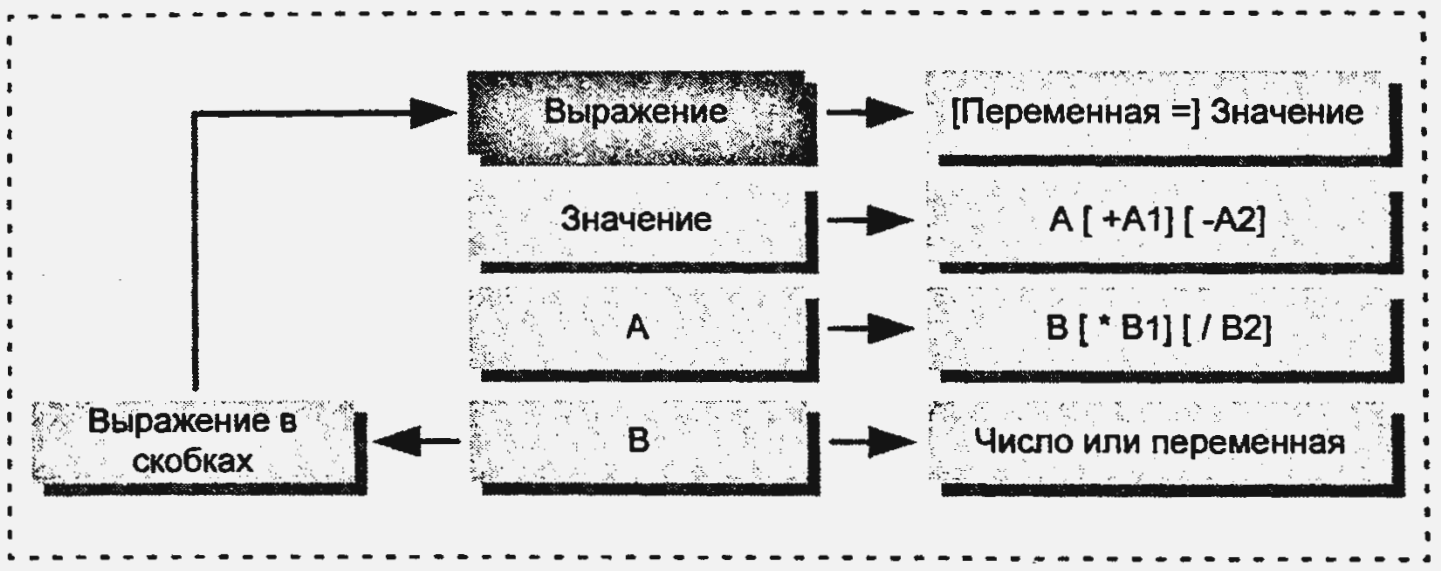


Рис. 5. Схема построения выражений

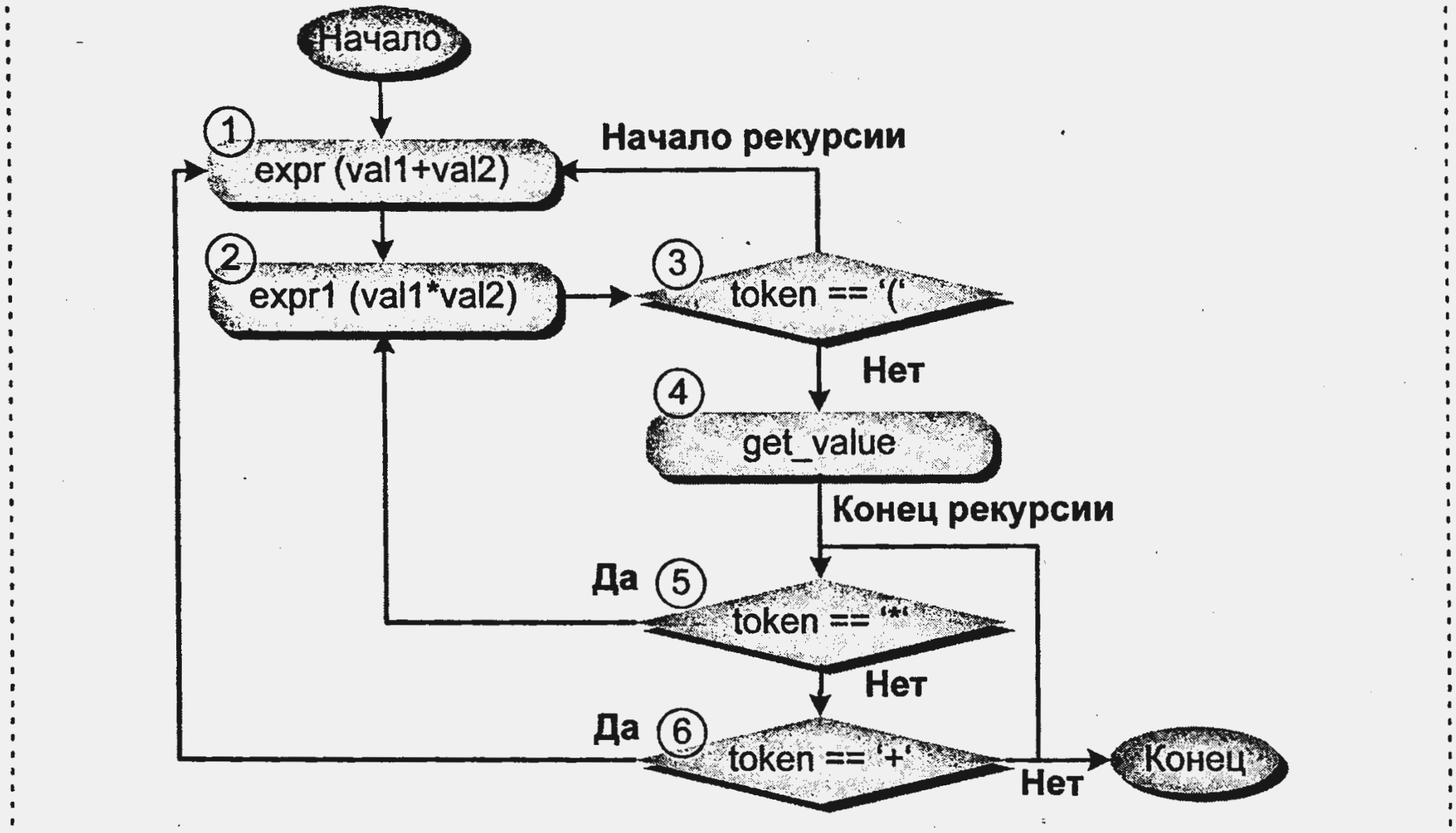


Рис. 6. Упрощенная схема работы анализатора выражений

```
// файл с подпрограммами "subprograms"
#define PI 3.14159          // определение константы
bool b_enabled = true;      // объявление переменной
                            // пример простейшей функции.
                            // x_coord,y_coord,z_coord - параметры функции.
                            // void -возвращаемое значение
void initialization(double x_coord, double y_coord, double z_coord)
{
  G00 X=x_coord Y=y_coord Z=z_coord
  G01 X=x_coord Y=y_coord Z=z_coord+10
}
```

```
#use "subprograms"          // включение файла с подпрограммами

if(!b_enabled)              // пример сложного оператора
{
  M30
}
initialization(70, 70, 10);
G01 X70 Y70 Z5
G01 X45 Y70 Z-5
G03 X45 Y20 I45 J45
G01 X70 Y20 Z-5
G01 X70 Y20 Z10
M30
```

Рис. 7. Пример программы на языке высокого уровня

(expr). На втором шаге вызывается функция умножения и деления множителей (expr1). На третьем шаге, если множитель начинается со скобки, выполняется вычисление выражения в скобках, т. е. рекурсивный вызов алгоритма вычисления выражения. В противном случае осуществляется чтение значения числа или переменной (get_value — четвертый шаг).

После вычисления значения проверяется следующая за значением лексема (пятый шаг). Если это оператор, соответствующий функции expr1 (умножение в данном случае), происходит возврат в expr1 и вычисление следующего множителя, и так до тех пор, пока не перестанут встречаться операторы умножения.

По аналогии, после вычисления значения, состоящего из всех множителей (шестой шаг), происходит проверка следующей лексемы, и если это плюс, происходит возврат в expr и вычисление следующего слагаемого. Если же лексемы кончились, выражение считается вычисленным. При необходимости поддержки других операций (сравнений, унарных операторов, возведения в степень и т. п.) в эту схему просто встраиваются функции, аналогичные expr и expr1. При этом для каждого приоритета операций должна быть своя функция анализа. Чем выше приоритет операции, тем ниже в схеме синтаксического разбора должна встраиваться соответствующая функция.

На рис. 7 приведен пример управляющей программы на языке высокого уровня, в которой блоки подпрограмм и составных операторов реализованы с помощью фигурных скобок.

Систематизируем условия реализации интерпретатора:

1. Все конструкции языка строятся по строго определенным правилам. Отсюда следует, что при чтении очередной лексемы из текста подпрограммы мы можем определить, какая конструкция языка начинается с этой лексемы и прочитать (интерпретировать) эту конструкцию целиком.

2. Выполнение программы начинается с вызова основного текста программы и представляет собой рекурсивный анализ блоков операторов и выражений.

3. Чтобы вызвать определенную функцию, надо заранее знать ее положение в тексте программы. Также, чтобы использовать определенную переменную или константу в выражении, надо заранее прочитать ее объявление. Все используемые файлы с текстами подпрограмм должны заранее анализироваться.

На рис. 8 представлена обобщенная блок-схема функционирования интерпретатора, построенного в соответствии с правилами выполнения программы на структурированном языке высокого уровня. Сначала выполняется чтение всех функций, включаемых файлов, определений переменных и констант (prescan) (шаг 1). Затем осуществляется вызов главной функции программы (call) (шаг 2). На третьем шаге интерпретируется очередная конст-

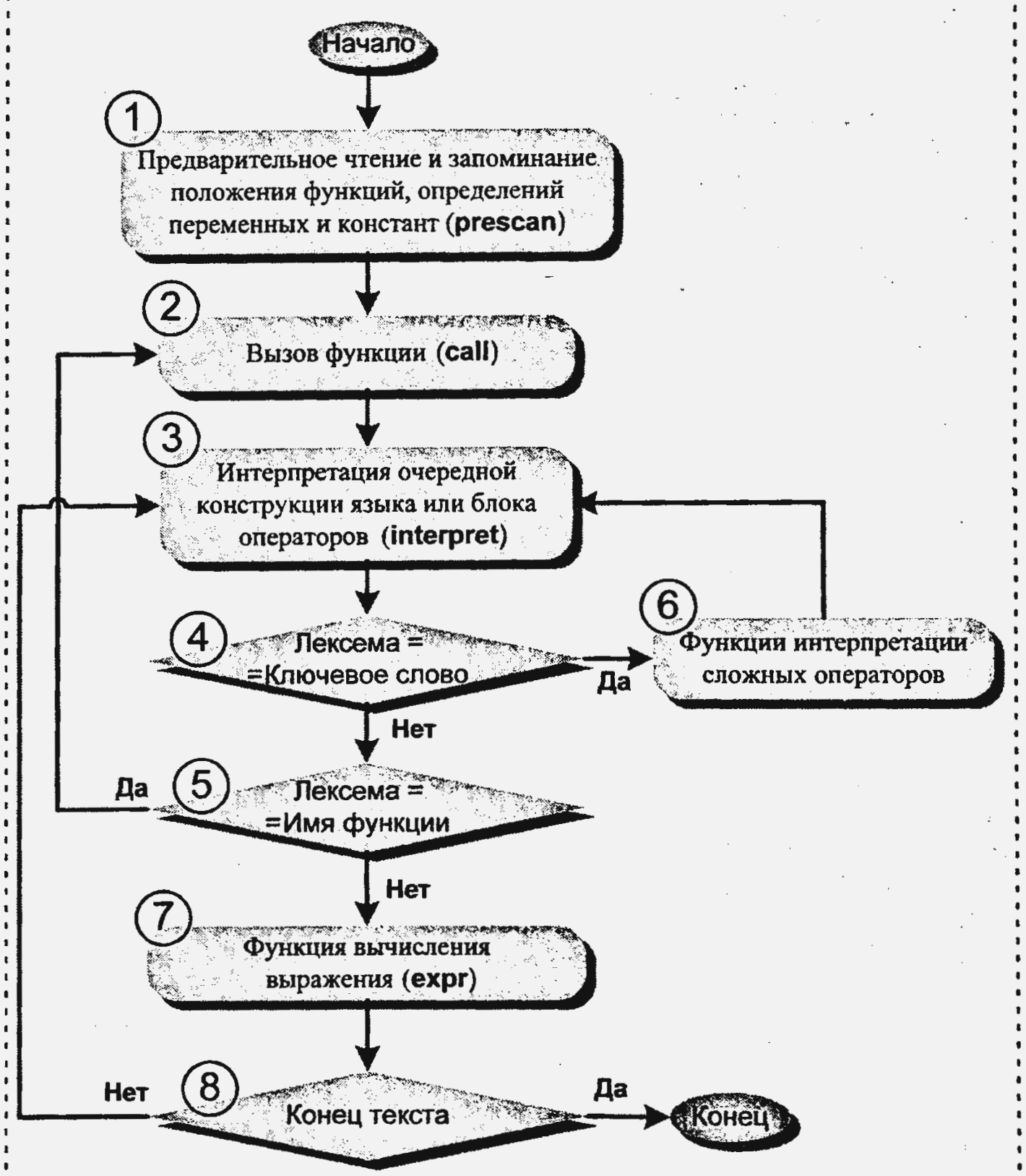


Рис. 8. Интерпретация программы на языке высокого уровня

рукция тела функции (interpret), при этом выполняется проверка лексемы (шаг 4). Если встретилось ключевое слово, определяющее оператор, вызывается нужная функция обработки оператора (шаг 6), в противном случае выполняется еще одна проверка (шаг 5). Если лексема является именем функции, осуществляется вызов (шаг 2). В противном случае обрабатывается выражение (шаг 7). На восьмом шаге выполняется проверка лексемы. Если оказалось, что текст закончился (или встретился конец блока операторов), интерпретация завершается. Если нет, интерпретируется очередная конструкция тела функции (interpret) (шаг 3).

Рассмотренная схема позволяет относительно просто реализовать интерпретатор Си-подобного языка, используемого для системы ЧПУ в качестве языка высокого уровня. Помимо указанных функций в интерпретаторе должно быть реализовано чтение параметров функций, сохранение в стеке локальных переменных и др. Их реализация алгоритмически довольно проста и зависит только от конкретного синтаксиса языка.

Модель интеграции интерпретатора языка высокого уровня в систему ЧПУ

Обработка файла на языке ISO-7bit в системе ЧПУ возложена на модуль интерпретатора программ. Интерпретатор входит в реализацию геометрической задачи системы ЧПУ [8] и выполняет: синтаксический анализ строк управляющей программы; преобразование координат и единиц измерения; эквидистантную коррекцию траектории; вызов и подстановку подпрограмм, чтобы в результате сформировать данные для интерполятора.

Любая система ЧПУ должна поддерживать язык ISO-7bit как базовый, так как в наличии у любого пользователя системы ЧПУ имеется большое число управляющих программ на этом языке. Таким образом, при интеграции интерпретатора языка высокого уровня в систему ЧПУ нужно учитывать необходимость поддержки языка ISO-7bit.

Предложена схема реализации интерпретатора языка высокого уровня в качестве надстройки (препроцессора) над имеющимся в системе интерпретатором ISO-7bit (рис. 9, см. третью сторону обложки).

Файлы управляющей программы на языке ISO-7bit пропускаются через препроцессор без обработки и передаются интерпретатору.

Файл на языке высокого уровня обрабатывается препроцессором, а полученные строки в коде ISO-7bit передаются на выполнение интерпретатору.

Препроцессор настраивается с помощью файла конфигурации языка в формате XML. В файле конфигурации описаны: лексемы структурных элементов языка; соответствие между входными и выходными командами языка ISO-7bit, что позволяет настроить препроцессор на обработку файлов разных диалектов ISO-7bit.

Набор операторов языка высокого уровня остается жестко заданным, но посредством файла конфигурации лексемы операторов, блоков и других элементов языка можно свободно расширить [9].

При реализации препроцессора следует обратить внимание на два момента. Первый — обработка включений кода ISO-7bit в тексте языка высокого уровня — решается с помощью добавления в интерпретатор функции, обрабатывающей цепочки ISO-команд как цельные конструкции языка. Второй — реализация библиотечных функций языка высокого уровня для генерации ISO-команды — решается с помощью таблицы имен, где имени каждой функции сопоставлен обработчик. Функция call (см. рис. 8) вызывает соответствующий обработчик при встрече в тексте управляющей программы имени библиотечной функции.

Практическое применение препроцессора в системе ЧПУ WinPCNC

Интеграция препроцессора по предложенной схеме реализована в системе ЧПУ WinPCNC [10], что позволило значительно расширить возможности программирования системы управления. Любой сложный цикл легко реализуется в виде параметрической функции, вызываемой управляющей программой [11]. В качестве иллюстрации создания сложных станочных циклов приведем пример кода фрезерования восемнадцати круглых карманов (рис. 10).

Фрезерование кармана оформляется в виде подпрограммы `cut_round ()` с параметрами (рис. 11), которая, в свою очередь, вызывается в подпрограмме `G384_routine ()`, реализующей стандартный цикл фрезерования отверстий по шаблону G384 (рис. 12).

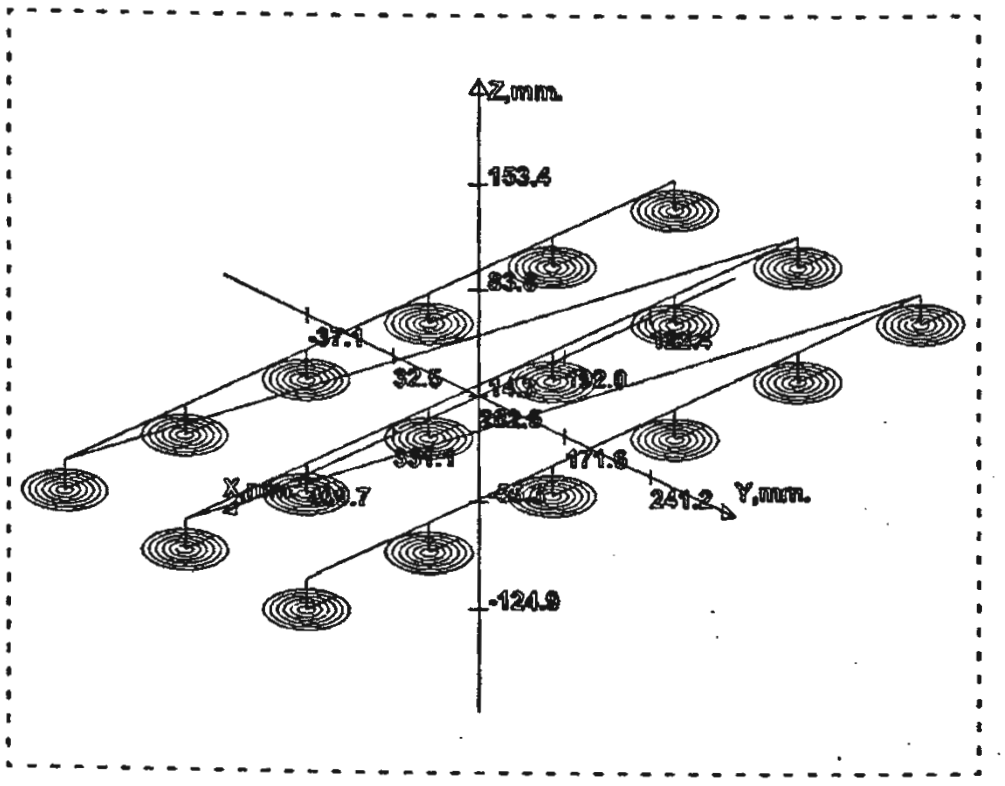


Рис. 10. Пример траектории инструмента при обработке контура, построенного с помощью пользовательских циклов

```
// файл подпрограмм "subprograms"
// функция фрезерования кармана
// Параметры:
//   x_center - координата центра кармана по оси x
//   y_center - координата центра кармана по оси y
//   radius - радиус кармана
//   tool_radius - радиус инструмента
void cut_round(double x_center, double y_center, double radius, double
tool_radius)
{
   double rm;
   G1 X=x_center Y=y_center Z=20
   G1 X=x_center Y=y_center Z=0
   for(rm = radius; rm > tool_radius *2; rm=rm-tool_radius *2)
   {
     G1 X=x_center -rm, Y=y_center Z=0
     G2 X=x_center -rm, Y=y_center I=x_center J=y_center
   }
    G1 X=x_center Y=y_center Z=0
    G1 X=x_center Y=y_center Z=20
}
```

Рис. 11. Пример подпрограммы реализации фрезерования кармана

```
// файл "standard_cycles"
// подпрограмма описания стандартного цикла
// фрезерования отверстий по шаблону

#use "subprograms"
// Q1 - начальное значение координаты X
// Q2 - конечное значение координаты X
// Q3 - начальное значение координаты Y
// Q4 - конечное значение координаты Y
// Q5 - шаг вдоль оси X
// Q6 - шаг вдоль оси Y
// Q7 - радиус отверстия
// Q8 - радиус инструмента
void G384_routine(double Q1, double Q2, double Q3, double Q4, double Q5,
double Q6, double Q7, double Q8)
{
  int index1, index2;
  for(index2 = Q1; index2 < Q2; index2=index2+Q5)
  {
    for(index1 = Q3; index1 < Q4; index1=index1+Q6)
    {
      cut_round(index1, index2, Q7, Q8);
    }
  }
}
```

Рис. 12. Пример подпрограммы, реализующей цикл G384

Использование станочного цикла G384 в управляющей программы на языке ISO-7bit проиллюстрировано на рис. 13 (см. третью сторону обложки).

Заключение

Применение языка высокого уровня повышает открытость систем ЧПУ. С помощью этого механизма станкостроители способны встраивать собственные станочные и измерительные циклы, а конечные пользователи — разрабатывать групповые технологии обработки без изменения ядра системы управления.

Предложенная модель позволяет интегрировать препроцессор языка высокого уровня в существующие системы ЧПУ без кардинальных изменений в ядре системы, обеспечивая совместимость с накопившейся базой отлаженных управляющих программ.

Структурированная природа Си-подобного языка высокого уровня позволяет относительно легко реализовать препроцессор для систем ЧПУ на основе нисходящего рекурсивного синтаксического анализатора.

Использование отлаженных программных модулей в управляющей программе повышает надежность программирования системы ЧПУ и сокращает время разработки управляющей программы.

Список литературы

1. **Сосонкин В. Л., Мартинов Г. М.** Методика разработки управляющей программы ЧПУ соответственно стандарту ISO 14649 STEP-NC (Standard for the Exchange of Product model data for NC) // Мехатроника, автоматизация, управление. 2005. № 6. С. 45—52.

2. **User's** Manual. Heidenhain. Conversational Format. iTNC 530 (http://www.heidenhain. com).

3. **Мартинов Г. М., Пушков Р. Л.** Построение инструментария отладки управляющих программ систем ЧПУ на языках высокого уровня // Приборы и системы. Управление, контроль, диагностика. 2008. № 11. С. 19—24.

4. **Crocker D., Ed., Oversll P.** Augmented BNF for Syntax Specifications: ABNF. RFC 5234 (http://tools.ietf.org/html/rfc5234).

5. **Сосонкин В. Л.** Программное управление технологическим оборудованием: Учеб. для вузов по специальности "Автоматизация технологических процессов и производств". М.: Машиностроение, 1991. 512 с.

6. **Шилдт Г.** Полный справочник по C, 4-е изд-е. М.: Диалектика, 2003. 800 с.

7. **Ахо А. В., Лам М. С., Сети Р., Ульман Д. Д.** Компиляторы. Принципы, технологии и инструментарии. М.: Вильямс, 2008. 1184 с.

8. **Сосонкин В. Л., Мартинов Г. М.** Системы числового программного управления: Учеб. пособ. М.: Логос, 2005. 296 с.

9. **Сосонкин В. Л., Мартинов Г. М.** Программирование систем числового программного управления: Учеб. пособ. М.: Логос; Университетская книга; 2008. 344 с. + 1 компакт-диск.

10. **Мартинов Г. М.** Университетская система ЧПУ WinPCNC для обучения и производства // Стружка. 2008. № 1. С. 29—30.

11. **Мартинов Г. М., Григорьев А. С.** Разработка пользовательских токарных циклов в системе ЧПУ WinPCNC // Объединенный научный журнал. 2007. № 6. С. 8—50.