

Р.Н.Вильданов

РЕАЛИЗАЦИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ АРМ В СПО ГД

Опыт эксплуатации системы СПО ГД [1] за последние три года выявил целый ряд недостатков ее подсистемы ПО АРМ [2]. К ним относятся: неудобное управление устройствами ввода, неоправданно большой объем оперативной памяти, занимаемый системой, плохие временные характеристики работы части программных средств системы, недостаточная индикация смысловых операций системы на дисплеях. Практически все перечисленные недостатки были устранены. Краткое описание ПО АРМ и его новых возможностей приводится в настоящей работе.

Комплексы АРМа имеют широкий набор различных периферийных устройств. Желательно обеспечить программе пользователя ввод команд, запросов и информации с этих устройств. Последние имеют обычно многоуровневую систему прерываний, а программа пользователя не может знать о состоянии прерывания, если она написана на некотором языке высокого уровня, например, на Фортране. Еще более удобно, если вся "кухня" обработки прерываний от устройств для него будет закрыта, и в то же время существует возможность ввода информации с любого устройства. Пользователь сам выберет наиболее удобное ему устройство.

Все программы работы с периферийными устройствами, канальная часть, поддерживающая связь с большой ЭВМ, различные вспомогательные программы расположены в базовой части (резиденте) ПО АРМа. Пользовательские программы (их количество ог-

раничено емкостью внешнего запоминающего устройства) имеют оверлейную структуру. Любое перекрытие может вызвать другое перекрытие. При этом новое перекрытие загрузится на место старого. Это новое перекрытие, в свою очередь, может вызвать следующее перекрытие и т.д. Затем может происходить обратный процесс. Вызванное перекрытие может вернуть управление старому, при этом последнее опять загружается в память, и управление передается на следующую за точкой вызова команду.

Естественно, что все собственные переменные загруженного перекрытия будут иметь начальное состояние, за исключением тех, которые размещены в COMMON - блоках резидента. Вложенность таких перекрытий может достигать 50. Как показал опыт, этого вполне достаточно. Некоторые из этих перекрытий могут вызываться только из большой ЭВМ, остальные - по командам с пульта АРМа. Весь механизм вызова перекрытий обеспечивается ПО.

Рассмотрим следующие периферийные устройства АРМа:

1. Видеотон VT-340.
2. Графический дисплей ЭПГ-400 с функциональной (ФК ГД) и алфавитно-цифровой (АЦК ГД) клавиатурами и световым пером (СП).
3. Внешнее запоминающее устройство на дисках ИЗОТ-1370.
4. Координатное устройство ввода ЭМ-709 с алфавитно-цифровой клавиатурой.
5. Ввод и вывод с перфоленты.

На экране ЭПГ-400 программно моделируются маркер, крест и световые кнопки (меню). Отметим, что если в комплекте АРМа нет какого-либо из этих устройств, ПО не требует никаких переделок, если же имеется иное устройство, то его достаточно просто встроить в ПО. Перфолента, маркер и крест вызываются набором соответствующего функционального кода (ФК), а все остальные устройства ввода готовы к работе в любой момент времени. Например, при вводе строки пользователь может набрать одну букву на видеотоне, другую - на АЦК ГД, следующую - на АЦК сколки. Для ПО АРМа безразлично, с каких устройств произошел ввод. Этим достигается универсальность по отношению к устройствам ввода.

Пользовательским программам могут понадобиться следующие типы вводимых данных: 1) функциональный код; 2) строка; 3) последовательность чисел (целое или вещественное); 4) указание.

В таблице знаком "+" отмечается возможность ввода данных с конкретного устройства, а знаком "-" - невозможность ввода.

Функция- код	Тип данных				Устройство
	целое	веществ.	строка	указан.	
+	+	+	+	-	VT-340
+	+	+	+	-	АЦК ГД
+	-	-	-	-	ФК ГД
+	+	+	+	-	АЦК сколки
-	-	+	-	-	Координатное уст- ройство сколки
+	+	+	+	-	Перфолента
-	-	-	-	+	Световое перо
-	-	+	-	-	Маркер
-	-	+	-	-	Крест
+	-	-	-	-	Меню

Ввод данных идентифицируется в поле служебных надписей как на ЭПГ-400, так и на видеотоне. На ЭПГ-400 служебные надписи расположены в верхней части экрана за пределами рабочего поля, на видеотоне они занимают три верхние строчки экрана.

Поясним каждый тип данных.

А. Если в поле служебных надписей появляется надпись * ФК *, значит, система запрашивает ввод функционального кода. Обращение: `CALL INF(ICODE)`. Ввод ФК вызывает инициацию определенной программы пользователя или же ввод вспомогательной функ-

ции резидента, например, вызов маркера на экран ЭПГ-400, включение или выключение служебных надписей на видеотоне, ввод с перфоленты и т.д. Ввод ФК с ФК ГД естествен, а возможность ввода с алфавитно-цифровых клавиатур обусловлена тем, что пользователь, набрав соответствующее восьмеричное число, ввод заканчивает не так, как обычно, — нажав на клавишу RETURN, а нажатием клавиши ω (код 100). Ввод ФК возможен и с меню. Меню расположено в правой части экрана ЭПГ-400 за пределами рабочего поля экрана. Меню может быть составлено максимум из из 16 надписей, расположенных одна под другой. Например, вид одной из надписей: "104 - ДИАЛОГ". Впереди стоит соответствующий ФК, затем, через дефис, 6-буквенная интерпретация ФК. Оператор указывает на одну из надписей в меню СП, вызывая мерцание надписи. Убедившись, что мерцает нужная надпись, СП подводят к нижней части экрана, где светится надпись: "ПОДТВЕРЖДЕНИЕ". После этого вызывается соответствующая программа пользователя, и надпись перестает мерцать. Пользовательским программам предоставляется возможность составления своего меню с помощью программы MENU.

Б. При появлении служебной надписи * ТЕКСТ * ПО запрашивает ввод текстовой информации. Обращение: CALL INS (A, I, K), где в A будет помещен массив текста, I — количество запрашиваемых символов, K — количество введенных символов. Если $I > K$, программа INS усекает число введенных символов до I для того, чтобы не испортить программу пользователя. Программа позволяет вводить до 80 символов.

В. Ввод чисел идентифицируется служебными надписями * ЦЕЛОЕ *, * ВЕЩЕСТВ *. Разделителем при вводе последовательности чисел является запятая. Целые числа можно ввести с одной из алфавитно-цифровых клавиатур. Программа INT преобразует поступивший символьный код в целое, и в программу пользователя поступает уже преобразованный код. Ввод вещественных чисел возможен как с алфавитно-цифровых клавиатур, так и с координатных устройств типа маркер, крест, сколка. Программа устанавливает, с помощью специального флагка, откуда поступила информация. В зависимости от этого она тем или иным способом преобразует символьный код в вещественное число, в соответствии с типом внешнего устройства.

Г. Когда ПО запрашивает ввод указания, в поле служебных надписей видеотона и дисплея загорается надпись **ЖУКАЗ**. Ввод указания производится только СП. Указанный элемент сегмента начинает мерцать, и в пользовательскую программу засыпается адрес элемента в памяти. Таким образом, мы рассмотрели все типы вводимых данных с пульта АРМа.

В качестве одного из многочисленных применений ПО приведем пример ввода графической информации (графики, изолинии, печатные платы, чертежи) со сколки ЭМ-709 в ЭВМ. Пакет программ **SEISMO** оформлен по правилам ПО в виде перекрытий, которое само, в свою очередь, вызывает несколько перекрытий. Для контроля вводимой информации используется дисплей ЭПГ-400, т.е. вводимые координаты графика тут же будут отображаться на дисплее, а данные записываться на магнитный диск. Если же оператор ввел ошибочно не те координаты, он может сделать исправление этих координат, при этом последние точки убираются как на дисплее, так и на диске. Повторяя несколько раз эту операцию, оператор может исправить весь текущий график. Лист, с которого вводится информация, может быть закреплен на сколке произвольно относительно прямоугольных координат сколки, кроме того, лист может иметь деформацию (прямоугольник на листе может иметь вид произвольного четырехугольника). Программа **SEISMO** автоматически скорректирует эти погрешности. Ввод графической информации в ЭВМ можно производить частями, т.е. сегодня можно ввести один лист, завтра другой и т.д., так как программа запрашивает оператора в начале сеанса о дополнении массива на диске. Если не требуется дополнения, массив на диске будет записываться с начала.

Так как на листе может быть несколько графиков, набором соответствующего ФК на АЦК сколки или на любом другом устройстве, позволяющем ввести ФК, можно закончить текущий график и начать новый, или же закончить лист. По желанию оператора можно отрисовать введенный лист на графопостроителе. Затем, по запросу от большой ЭВМ, можно передать весь массив введенных чисел и произвести математическую обработку результатов.

В заключение отметим, что данное ПО может успешно функционировать в различных областях науки и техники.

онировать как на АРМ-Р, так и (после небольших переделок, касающихся графического дисплея) на АРМ-М, а также на базовом комплекте СМ-3, СМ-4.

Л и т е р а т у р а

1. Дебелов В.А., Мацокин А.М., Чубарев А.И. СПО ГД - система программного обеспечения графического диалога. - В сб.: Проблемы машинной графики. Новосибирск, 1982, с. 64-71.
2. Чубарев А.И. Средства программирования сателлита в рамках СПО ГД. - Настоящий сборник, с. 23-44.

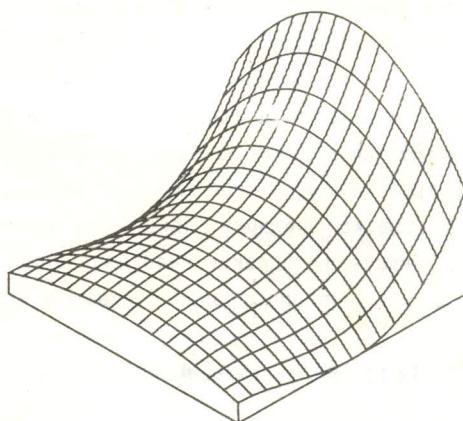
ВЫВОД ФОРМУЛЬНОЙ ИНФОРМАЦИИ
НА ГРАФОПОСТРОИТЕЛЬ

Разработка систем автоматизации процесса подготовки документации к пакетам прикладных программ (ППП) с использованием ЭВМ становится актуальной в связи со все более расширяющимся развитием и распространением ЭВМ, а также с увеличением выпуска программной продукции. В процессе подготовки документации к ППП иногда возникает потребность в подготовке статей, оформлении описаний математического аппарата и т.п. Но текст с математическими формулами не представим в общем случае на АЦПУ, поэтому и встает вопрос об изображении математических формул. По этому направлению существуют работы за рубежом [1] и у нас в стране [2,3]. Но эти разработки ориентированы на использование специализированной типографской техники, такой, как фотонаборные автоматы, и на данном этапе их широкое внедрение затруднительно. С другой стороны, получили распространение внешние графические устройства (ГУ) ЭВМ. Качество графического материала, получаемого на них, достаточно высокое для копирования либо издания на ротапринтной технике.

В данной работе описывается система ФОРМУЛА, которая позволяет оформлять и выводить на ГУ текстовой документ (например, статью), содержащий математические формулы и иллюстрации (рисунки). Для подготовки исходного текстового документа могут быть использованы различные программные и аппаратные средства (редакторы, алфавитно-цифровые дисплеи, перфораторы и т.д.), которые обычно применяются при работе с текстовой информацией. По своему назначению система ФОРМУЛА подобна системе ДОКУМАТОР [4], с той разницей, что последняя выводит текстовой документ на АЦПУ, а ФОРМУЛА получает результат на ГУ. В связи с тем, что основное внимание уделено разработке средств вывода формул, описываемая

система предоставляет меньше средств по оформлению текста. Рисунки готовятся при помощи средств стандартной графической системы СМОГ [5].

В текстовом документе (ТД) выделяются следующие составные части: резервированные и центрированные строки, текстовые и формульные абзацы, список литературы, прямоугольные таблицы, рисунки. Текстовые абзацы состоят в свою очередь из слов и строчных формул. Из корректурных средств имеются средства управления переносом слов и нумерацией страниц.



Пример иллюстрации

Формульная информация двумерна, поэтому для ее описания нужен особый язык, который удовлетворял бы следующим требованиям:

- а) наглядность и простота в обращении;
- б) использование символов, имеющихся на алфавитно-цифровых дисплеях;
- в) однозначное определение формулы по описанию.

Один из языков был предложен в работе [1]. Широту класса описываемых формул, хорошую наглядность, наличие макросредств и т.п. можно отнести к его достоинствам. Но нельзя не отметить наличия неоднозначности формулы по описанию, если, конечно, не вводить

приоритет операций; а система с использованием приоритета требует дополнительных затрат и может приводить к ошибкам пользователей.

Элементы языка	Отображение
$\diamond\text{SQRT}(A)$	\sqrt{A}
$\diamond\text{SUB}(A, I)$	A_I
$\diamond\text{SUP}(A, K)$	A^K
$\diamond\text{SUM}(A, B, C)$	$\sum_{A}^B C$
$\diamond\text{INT}(A, B, C)$	$\int_A^B C$
$\diamond\text{DIV}(A, B)$	$\frac{A}{B}$
$\diamond\text{LSI}(A, B, C)$	$\frac{A}{B} C$
$\diamond\text{NORM}(A)$	$\ A\ $
$\diamond\text{MOD}(A)$	$ A $
$\diamond\text{MAT}(m \times n, A_{11}, \dots, A_{mn})$	$\begin{matrix} A_{11} & \dots & A_{1n} \\ \vdots & \ddots & \vdots \\ A_{m1} & \dots & A_{mn} \end{matrix}$
$\diamond\text{TAB}(m \times n, A_{11}, \dots, A_{mn})$	$\begin{array}{ c c c } \hline A_{11} & \dots & A_{1n} \\ \hline \vdots & \ddots & \vdots \\ \hline A_{m1} & \dots & A_{mn} \\ \hline \end{array}$
$\diamond\text{ALT}(m \times n, A_{11}, \dots, A_{mn})$	$\left\{ \begin{matrix} A_{11} & \dots & A_{1n} \\ \vdots & \ddots & \vdots \\ A_{m1} & \dots & A_{mn} \end{matrix} \right\}$
$\diamond\text{COMB}(n, A_1, \dots, A_n)$	$A_1 \dots A_n$
$\diamond\text{TEXT}(\text{TEKST})$	ТЕКСТ
$\diamond\text{PAR}(A)$	(A)
$\diamond\text{SPAR}(A)$	$[A]$
$\diamond\text{FPAR}(A)$	$\{A\}$

В системе ФОРМУЛА реализован язык описания формульной информации

мации, дающий однозначность определения формул. Он основан на функциональном принципе и базируется на элементах языка, перечисленных в таблице. В данной таблице A,B,C,I,K,A₁,...,A_n - это тоже элементы языка. Исключение составляет элемент TEXT, в котором TEXT - это строка, которую пользователь хочет написать в формуле. Кодируя определенным образом, пользователь может употреблять в своей формуле символы из русского, латинского и греческого алфавитов, а также из довольно обширного набора математических операций, например, V,Λ,Γ,Ε,⇒,Σ,Ε,Σ,Π,Π,Δ,Α,Ξ.

Пример 1. Описание и вид формулы:

ΦLSI(ΦTEXT(↓LIM↓),ΦTEXT(↓v107v0970↓),
ΦINT(ΦTEXT(↓v107↓),ΦTEXT(↓a↓),ΦTEXT(↓ln x dx↓)))

$$\lim_{x \rightarrow 0} \int_a^x \ln x \, dx.$$

По описанию ТД система ФОРМУЛА строит информационное дерево, затем производится трассировка его вершин, с целью выяснения размеров соответствующих подформул. По размерам формулы вычисляется ее размещение на странице. И наконец, дерево просматривается еще раз для окончательной отрисовки формулы. Каждая вершина дерева несет следующую информацию:

- ссылка на отца;
- указатель на элемент языка;
- число аргументов;
- номер вершины как аргумента в нумерации отца;
- признак отрисовки и трассировки;
- ссылка на размеры и координаты;
- показатель основного размера знака;
- ссылки на аргументы.

Отцом считается вершина, для которой данная вершина является аргументом.

Для системы ФОРМУЛА был разработан и реализован алгоритм, позволяющий расширять множество элементов языка. Для добавляемого элемента должно существовать правило геометрического размещения аргументов относительно друг друга. С помощью этого механизма

были реализованы прямоугольные таблицы, матрицы и т.д. Таким же образом можно ввести химические структурные формулы.

ФОРМУЛА может функционировать как отдельная система, так и совместно с другими. Например, язык задания формул в системе САПФИР [3] по структуре подобен нашему, хотя и несколько отличается. Поэтому, разработав конверторы с языка ФОРМУЛЫ на язык САПФИРа и наоборот, можно использовать систему ФОРМУЛА для получения на ГУ черновых, пробных выдач документа, подготовленного в системе САПФИР.

Система ФОРМУЛА реализована на языке Fortran Burroughs-6700 в ВЦ СОАН СССР и опирается на средства системы СМОГ [5] для вывода на ГУ. Отметим, что средства управления размещением собственно текста и средства описания иллюстраций в данный момент явно недостаточны, поскольку предполагают получение чернового варианта с последующей его доработкой. Включение этих средств в систему ФОРМУЛА носит поисковый характер. На основании опытной эксплуатации предполагается выполнить доработку языка задания полного текстового документа (статьи), включающего формулы, иллюстрации и таблицы. Неудобством, связанным с получением чернового варианта ТД, можно пренебречь в случае включения ФОРМУЛЫ в интерактивную графическую систему.

Пример 2.

$$\sup_{k \in V} \left\{ \sum_{\xi_k \in S} W_{\xi_k}(x) \right\} = \sqrt[3]{\begin{vmatrix} a_1 & b_1 & 0 \\ c_2 & a_2 & b_2 \\ 0 & c_3 & a_3 \end{vmatrix}}. \quad (7.2)$$

Пример 3.

$$H = \sum_{A_1+...+A_n=1} \frac{\int_{A_1}^{A_n} ... \int_{A_n}^{A_n} B(x_1, \dots, x_n) dx_1 \dots dx_n}{\sqrt{1 + \frac{A_1 x_1 + \dots + A_n x_n}{\gamma + H}}}. \quad (\alpha)$$

Пример 4.

$$A = \begin{vmatrix} A_1 + C^T C & -C^T \\ -C & A_2 \end{vmatrix}.$$

Пример 5.

$$\sqrt{\sum_{n=1}^{10} \sum_{k=1}^n A_n^{2+k}}$$

Приведенные выше примеры, рисунок, таблица и сам текст статьи показывают возможности системы ФОРМУЛА. В настоящее время осуществляется перенос программного обеспечения на ЕС ЭВМ. Предполагается использовать систему ФОРМУЛА для вывода документации на микрофото.

Л и т е р а т у р а

1. Kernighan B.W. and Cherry L.L. A System for Typesetting Mathematics. - Comm. ACM, 1975, v.18, p.151-157.
2. Берс А.А., Демушев В.А. Автоматизированный набор блочных текстов в системе САПФИР. - В сб.: Новые задачи информатики. Новосибирск, 1979, с. 33-46.
3. Демушев В.А. Реализация автоматического набора формул в системе САПФИР. - В сб.: Программное обеспечение задач информатики. Новосибирск, 1982, с. 61-69.
4. Братухина В.А. и др. Системы автоматизации подготовки документации. ДОКУМЕНТАТОР БЭСМ-6. ДОКУМЕНТАТОР ЕС ЭВМ : Препринт 370. - Новосибирск, 1982. - 22 с. - В надзаг. : ВЦ СОАН СССР.
5. Математическое обеспечение графопостроителей. 1 уровень. СМОГ: Инструкция по программированию / Под ред. Ю.А.Кузнецова. - Новосибирск, 1976. - 78 с.

В.Г.Сиротин

ЯЗЫК ГРАФИТ ДЛЯ ОПИСАНИЯ, РЕДАКТИРОВАНИЯ, ХРАНЕНИЯ И
ВИЗУАЛИЗАЦИИ МОДЕЛЕЙ ДВУМЕРНЫХ ГЕОМЕТРИЧЕСКИХ
ОБЪЕКТОВ И ЧЕРТЕЖЕЙ

I. Введение

Язык ГРАФИТ предназначен для использования в автоматизированных системах, ориентированных на проектирование и технологическую подготовку в машиностроении.

С помощью языка ГРАФИТ в памяти ЭВМ строятся модели двумерных (2D) и так называемых $2\frac{1}{2}$ D геометрических объектов и

чертежей. Под $2\frac{1}{2}$ D-мерными понимаются объекты, которые можно интерпретировать как двумерные, например, тела вращения.

Построенные модели в дальнейшем могут быть переданы какой-либо прикладной программе (ПП) для проведения расчетов, сохранены в базе данных для последующей работы, в том числе и модификации. Другими словами, язык ГРАФИТ обеспечивает:

- эффективный ввод и редактирование моделей геометрических объектов и чертежей;
- быстрый доступ ко всем данным, хранящимся в модели для обеспечения разного рода проектных и конструкторских операций;
- хранение моделей;
- изготовление чертежей (разрисовку).

Язык предназначен для использования как системными и прикладными программистами (при разработке подсистем САПР и САПП),

так и проектировщиками-непрограммистами (для описания чертежей деталей).

В моделях, строимых ГРАФИТом, хранится информация следующих типов:

- геометрическая (координаты и параметры чертежных элементов);
- иерархическая (какие элементы описывают данную деталь);
- атрибутивная (вес, стоимость и т.д.);
- информация о связях элементов (способах их описания).
Язык ГРАФИТ является частью большой системы машинной графики для САПР и САПШ, куда, кроме него, еще входят:
 - подсистема работы с 3D-объектами;
 - подсистема обеспечения диалога;
 - специализированная СУБД.

2. Функциональные возможности языка

Поскольку грамотно составленный чертеж дает возможность получить полное представление об изображенном на нем 2D-объекте, то логично предположить, что модель 2D-объекта должна являться подмоделью модели чертежа.

Таким образом, при определении ассортимента возможностей языка ГРАФИТ возникло две задачи:

- реализация стандартного набора чертежных операций ("программирование" учебника черчения);
- определение набора функций, необходимых для обеспечения расчетных и прочих операций на различных этапах проектирования.

Что касается первой задачи, то основные сложности при ее решении возникли из-за низкой формальности в определении чертежных операций, таких, например, как построение сопрягающих дуг, простановки разного рода размеров и т.д.

Анализ, проведенный автором, показал, что не удается выделить небольшой набор функций, который оказался бы достаточным для обеспечения широкого круга расчетных и прочих операций. В силу этого основной упор был сделан на возможность

предоставить пользователю доступ к построенным с помощью ГРАФИТА моделям. В качестве функций, удобных для многих приложений, предоставляются типовые операции работы с плоскими областями (полигонами):

- вычисление периметра, площади, центра тяжести, габаритов;
- штрихование и текстурирование;
- экранирование, окнирования полигона и т.д.

3. Системные требования к работе трансляторов языка ГРАФИТ

Наиболее типичный процесс проектирования выглядит следующим образом. Сначала с помощью подсистемы по работе с 3D-объектами (рассмотрение данной подсистемы не входит в рамки настоящей статьи) формируется объемная модель детали. После анализа ее изображения и численных характеристик (вес, моменты и т.д.) проектировщик получает плоскую (2D) модель, например, проекцию детали. Эта плоская модель может быть дополнена (заштрихованы некоторые области, добавлены элементы), а затем из нее получается модель чертежа - проставляются размеры, специальные символы и знаки. Примерно по такой же схеме работает целый ряд зарубежных систем [1].

В этой схеме ГРАФИТ может быть использован на следующих этапах. Плоская модель получается из объемной при помощи вызова операторов языка ГРАФИТ в пакетном режиме работы. Для получения технической документации (чертежей) необходимо применить к модели 2D-объекта операции обозначения, проставить технологические обозначения. Эти операции, а также редактирование выполняются, как правило, в диалоговом режиме.

3.1. Работа с языком ГРАФИТ в пакетном режиме

Учитывая тот факт, что основными требованиями, предъявляемыми к языку ГРАФИТ при работе в пакетном режиме, являются требования совместимости с программами-генераторами 2D-моделей из 3D-моделей и требования совместимости с языками пользователя, реализованными на Фортране, язык ГРАФИТ реализован как пакет на языке Фортран.

По отношению к пользователю пакет выступает в виде геометрического языка – расширенного Фортрана. Расширение производится за счет введения нового типа переменных (геометрических) и операторов по работе с ними. Операторы по работе с геометрическим переменными представляют собой формально корректные конструкции языка Фортран. Например,

`T = TO(F1, F2)` – точка пересечения элементов `F1` и `F2`,

`CALL HEB(X)` – редактирование объекта `X`.

Это позволяет не использовать специальных методов макроперенаправления, предпроцессорной обработки и т.д., как, например, в [2], а использовать стандартные возможности транслятора с Фортрана.

3.2. Работа с языком ГРАФИТ в диалоговом режиме

Модели, построенные с помощью как пакетной, так и диалоговой версий языка ГРАФИТ, совместимы. Это позволяет работать с моделью одного объекта то в диалоговом, то в пакетном режиме.

При разработке диалоговой версии языка ГРАФИТ был проанализирован практический опыт работы в пакетном режиме с пакетной версией языка ГРАФИТ [3], ФАП-КФ [4], диалоговым графическим редактором, входящим в состав СПО ГД [5], текстовым диалоговым редактором, а также публикации о зарубежных и отечественных диалоговых графических системах. Этот анализ позволил выработать три основных принципа реализации диалоговой версии языка ГРАФИТ:

- а) диалоговый ГРАФИТ – интерпретирующая система;
- б) имена объектов генерируются автоматически;
- в) имеется возможность перенастраивать интерпретатор на терминологию данного пользователя.

Интерпретация операторов языка. Одной из основных возможностей языка ГРАФИТ является возможность редактирования моделей чертежей и 2D-объектов в режиме диалога. При выполнении практически любого оператора языка ГРАФИТ ранее определенное изображение на экране графического дисплея модифицируется, т.е. оператор за пультом видит результат своих действий. Было бы нелогично отвлекать его внимание еще и на программу (протокол его действий). Поэтому в диалоговой версии языка

ГРАФИТ программы пользователя, которую можно саму исправлять и перезапускать, как таковой не существует (как и у большинства диалоговых текстовых редакторов). Каждый оператор, введенный пользователем с пульта, интерпретируется, в результате чего изменяется либо модель и ее графический аналог - изображение на экране, либо режим функционирования интерпретирующей системы.

Автоматическая генерация имен. Как показал опыт практической работы с пакетной версией языка ГРАФИТ, пользователь при описании сложного чертежа (более сотни элементов), начиная с некоторого момента, именует вновь определяемые объекты либо случайным образом, либо просто нумеруя их. Продолжая работу с этим же чертежом через несколько дней, пользователь, конечно же, не в состоянии вспомнить, какое имя он дал каждому из элементов.

В силу этих причин в диалоговой версии языка ГРАФИТ имена элементов генерируются автоматически. Имя **элемента** составляется из одной или двух букв, обозначающих тип элемента и его порядковый номер. Например, набор операторов, изображенных на рис. I слева, приведет к появлению на экране дисплея изображения, показанного на рис. I справа.

TKAH 50, 100
AKAH T1, 30
YCV T1, -70,0
SKAC T3,A2

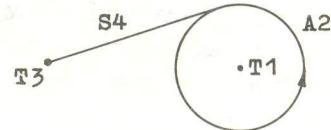


Рис. I

Оператор **TKAH** определил точку **T1** с координатами (50, 100), оператор **AKAH** определил окружность с центром в точке **T1** и радиусом 30, с помощью оператора **YCV** построена точка **T3**, полученная сдвигом точки **T1** на вектор (-70,0), оператор **SKAC** построил отрезок с началом в точке **T3** и касающийся **A2**. Существует "чистовой" режим получения изображений на дисплее и твердой копии - без вывода имен объектов и вспомогательных изображений.

Генерация диалектов языка. Интерпретатор языка ГРАФИТ реа-

лизован таким образом, что имеется возможность менять имена операторов, оставляя неизменным их семантический смысл. Так, например, имя "SKAC" можно заменить по просьбе пользователя (сгенерировав индивидуальную версию интерпретатора) на "ОТРКАС" или на " / (" и т.д. Необходимость этого объясняется тем, что для разных проектировщиков или коллективов удобны разные системы обозначений одних и тех же операций.

4. Общее представление о процессе построения и редактирования чертежа

Машиностроительные чертежи – это двумерные объекты, полученные, в соответствии с точно установленными правилами, из трехмерных объектов.

Анализ различных стадий проектирования показывает, что модель проектируемой детали в процессе проектирования становится все более сложной и детализированной, хотя с функциональной точки зрения деталь меняется несущественно. Соответственно на каждом этапе проектирования чертеж пополняется все большим количеством данных технологического и конструктивного характера.

Машиностроительный чертеж может быть рассмотрен как совокупность данных, логически разделенная на две части, – данные, представляющие форму и топологию детали, и данные, представляющие технологическую информацию [6].

Во многих немашиностроительных приложениях чертеж компонуют из крупных повторяющихся элементов – примитивов (электрические схемы, строительные конструкции и т.д.)

Машиностроительный же чертеж невозможно представить какими-либо базисными примитивами, его приходится структурировать на более низком уровне точек, отрезков и т.д. [6].

5. Моделирование 2D-объектов и их чертежей средствами языка ГРАФИТ

Рассмотрим принципы построения моделей 2D-объектов и их чертежей с помощью языка ГРАФИТ на простейшем примере. Между элементами простейшей детали, изображенной на рис. 2, существуют следующие отношения:

- точки T1 и T2 находятся на одной горизонтальной линии;
- окружности A3 и A4 имеют в качестве своего общего центра точку T2; аналогично для A5, A6 и T1;
- дуга D7 сопрягает окружности A6 и A4, а отрезок S8 их обеих касается.

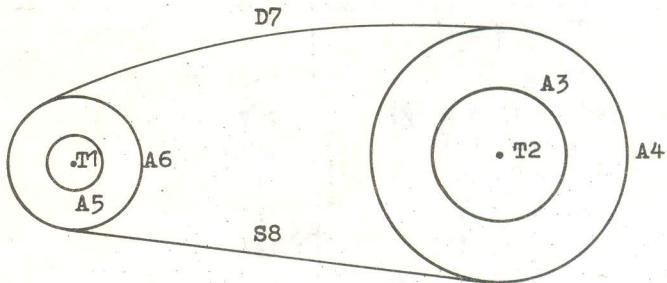


Рис. 2

Эти связи, записанные выше при помощи слов, можно представить в виде графа, отображающего сетевую структуру связей элементов, составляющих наш 2D-объект (рис. 3).

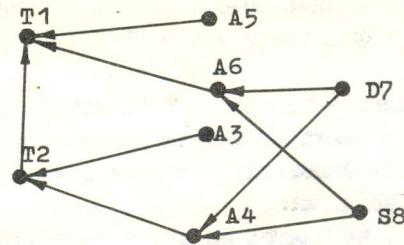


Рис. 3

С помощью языка ГРАФИТ [3] объект с рис. I можно описать следующим образом (I):

Таблица I

Пакетный вариант	Диалог
T1 = ТКАН (\emptyset , \emptyset)	ТКАН 20, 20
T2 = YCV (T1, 5 \emptyset , \emptyset)	YCV T1, 50, 0
T3 = АКАН (T2, I4)	АКАН 2,14
T4 = АКАН (T2, 3 \emptyset)	АКАН 2,30
T5 = АКАН (T1, 7)	АКАН 1, 7
T6 = АКАН (A5, I4)	АКАН 5, 14
T7 = D2FR (A6, A4, 10 \emptyset , I)	D2FRA6, A4, 100, I
T8 = SKAC (A4, A6)	SKACA4, A6

Операции ТКАН, YCV, АКАН, SKAC описаны в п. 3.2.

Операция D2FR служит для сопряжения дугой радиуса двух произвольных элементов (точек, отрезков, прямых, дуг ...). Естественно, эта операция не всегда выполнима. Поскольку мы при определении A6 в качестве центра указали не T1, а центр A5, то и та часть структуры данных, в которой отображаются связи элементов между собой, сгенерированная после проработки программ (табл. I), будет отличаться от структуры с рис.3. Вместо дуги (A6, T1) появится дуга (A6, T5). Направление связи (E_i, E_j) обозначает, что при построении элемента E_i использовался элемент E_j .

Простановка размеров детали (рис. 4) добавит к описанию на языке ГРАФИТ еще ряд операторов (табл.2) и увеличит количество элементов и связей между ними в модели. Еще более увеличивает количество связей отображение того факта, что все элементы принадлежат к чертежу одной детали, т.е. внесение в модель иерархических связей (рис. 5).

На рис. 5 не отображены связи, которые могут возникнуть при присвоении элементам разного рода атрибутов (марки материала, тип резьбы для отверстия и т.д.).

Таблица 2

Пакетный вариант	Диалог
RL9 = RL2T (T1, T2, Ø, 5Ø, Ø, Ø)	RL2T T1, T2, 0, 50, 0, 0
RD1Ø = RDK (A3, 8Ø, I, 2Ø, 5Ø)	RDK A3, 80, I, 20, 50
RD11 = RDK (A4, -2Ø, Ø, 3Ø, 2Ø)	RDK A4, -20, 0, 30, 20
⋮	⋮

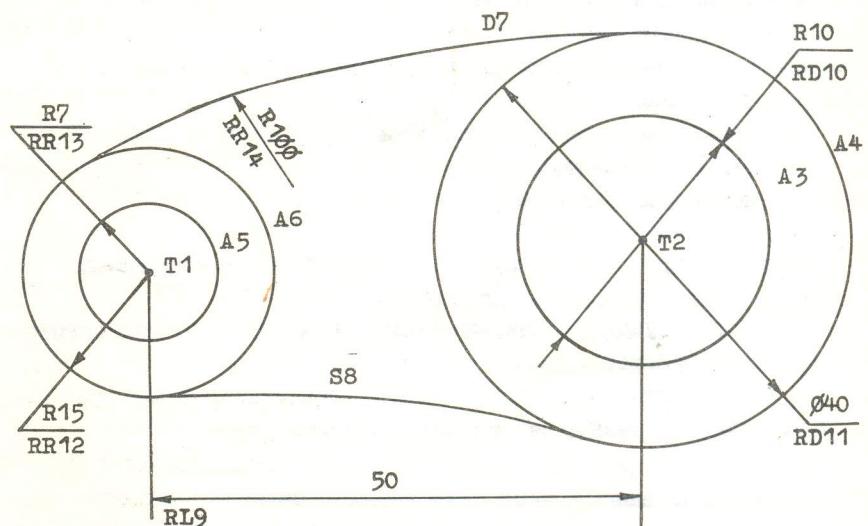


Рис. 4

71

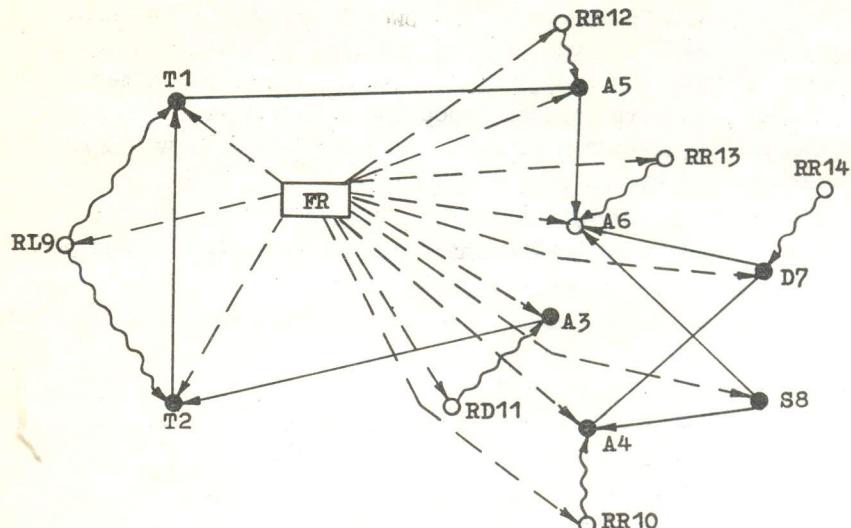


Рис. 5. Связи между элементами чертежа рис. 4:

- - геометрические элементы;
- - технологические элементы;
- - фрагмент;
- ← - связи между геометрическими элементами (топологические связи);
- ~~~~ - связь между парами (геометрический элемент, технологический элемент);
- ↔ - иерархические связи

Изучение рис. 5 показывает, что связи между элементами чертежа можно отнести к двум типам:

- "ассоциирующие", как, например, объединение всех элементов в один фрагмент;
- "конструктивные", т.е. связи, с помощью которых строится очередной элемент, если известны предыдущие.

Как видно из рис. 5, множество связей, относящихся к геометрическим элементам (сплошные линии), является подмножеством всех связей. Тем самым, с точки зрения связей, модель самого 2D-объекта является подмоделью его чертежа.

Пользователь имеет возможность получить полную информацию

о связях либо с помощью оператора отображения их на АЦПУ (в диалоге и в пакете), либо с помощью оператора передачи данных о связях элементов для расчетов в ПП (в пакетном режиме).

Считается, что все связи отображены в некоторой таблице связей, строки которой имеют переменную длину. Ниже рассмотрим только представление связей, которые называли "конструктивными". Совокупность всех связей, отображенная в таблице, называется объектным модулем (ОМ). Каждая строка в таблице объектного модуля (ТОМ), отображающая "конструктивную" связь, независимо от того, какие элементы эта связь связывает, имеет вид:

ТИП СТОЯЩЕГО ОПЕРАТОРА	ПОДТИП СТОЯЩЕГО ОПЕРАТОРА	ПАРАМЕТРЫ
------------------------	---------------------------	-----------

В качестве параметра, по аналогии с записью самого строящего оператора, могут выступать либо числа, либо номера геометрических или технологических объектов.

Для объекта с рис. 5 ТОМ будет выглядеть следующим образом (рис. 6):

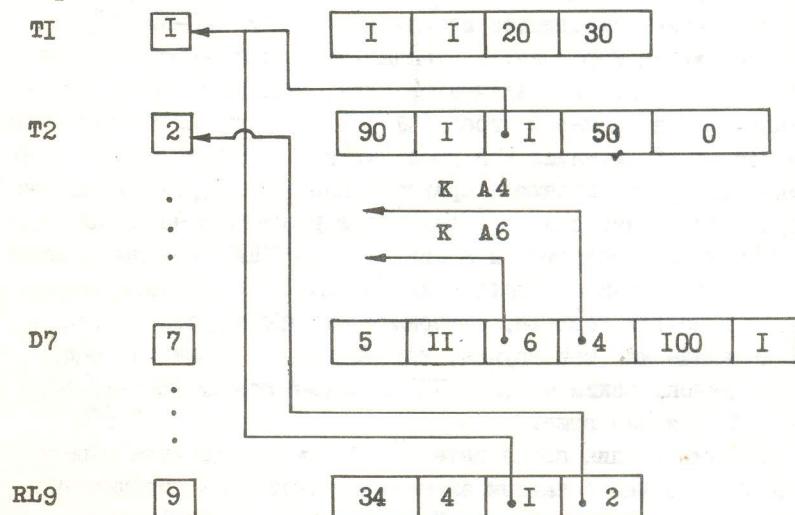


Рис. 6. ТОМ для объекта рис. 4:
в указанных позициях находятся
номера, а не числа элементов

Замечание I. Символьное описание чертежа (табл. I, 2), изображение чертежа (рис. 4) и ОМ эквивалентны в том смысле, что по любому можно получить два других. Их можно рассматривать как результаты последовательных проходов транслятора.

Интересным является вопрос о том, какие из представлений программы целесообразно хранить в процессе диалога. В интерпретаторе языка ГРАФИТ, по соображениям, высказанным в [8] и в п. 3.2., хранится внутреннее представление – ОМ, а не сам текст программы, описывающей чертеж.

Хотя в ОМ содержится информация, по своей полноте эквивалентная самому чертежу, для практического использования при расчетах ОМ мало пригоден. Так, например, для того чтобы оценить вес детали, нам необходимо знать, например, про дугу D7 по крайней мере ее величину в радианах и радиус. Для других расчетов могут понадобиться координаты центра, начальной, конечной точек и т.д.

Для обеспечения разного рода расчетов из ОМ генерируется (транслируется) так называемый канонический модуль – КМ. (Этот термин заимствован из описания языка ФАП-КФ [4]). В КМ содержится информация, стандартным образом форматированная для каждого типа элементов. Так, например, точка представляется в КМ парой чисел (x, y), а дуга окружности – пятеркой (x, y, F -начальный угол, R, F – величина дуги). Пользователю предоставляются средства передать в режиме пакетной обработки данные о любом элементе в расчетную ПП и/или распечатать их (и в пакете, и в диалоге) на АЦПУ. Вся информация о чертеже (и как ее часть – информация о 2D-объекте) содержится в модели чертежа, которая представляет собой данные, разделенные на три группы, или модуля, – объектный модуль (ОМ), канонический модуль (КМ) и атрибутивный модуль (АМ), ОМ и КМ описаны выше.

С точки зрения пользователя, АМ тоже представляет собой таблицу, только с каждым элементом может быть связано несколько строк этой таблицы. Каждая строка имеет формат:

ТИП АТРИБУТА	ТИП ДАННЫХ	ДЛИНА ЗАПИСИ	ДАННЫЕ
--------------	------------	--------------	--------

Тип атрибута – номер, соответствующий данным с конкретным семантическим смыслом.

Соответствие ТИП ДАННЫХ : ТИП АТРИБУТА задается самим пользователем (см., например, табл. 3):

Таблица 3

Названия	Тип данных	Тип атрибута
Марка материала	3	1
Вес	I	2
•	•	•
•	•	•
•	•	•
Чистота обработки	I	27

Атрибутивные данные могут быть либо целыми, либо вещественными числами, либо набором символов. Этот факт отражается в позиции "тип данных".

Пользователю предоставляются средства задания и распечатки на АЦПУ атрибутивных данных (в диалоге и в пакете), а также выдачи в ПП информации об атрибуте (в пакете).

6. Объекты и основные операции

Операции языка ГРАФИТ определены над элементами:

1. Типовыми геометрическими: точка, отрезок прямой, дуги кривых второго порядка;
2. Составными: полигоны, фрагменты;
3. Технологическими: линейные, угловые, диаметральные, радиальные и т.д. размеры, строки текста, шероховатости, допуски и т.д.

Каждый объект представляется некоторой совокупностью данных, основной интерес среди которых для пользователя, как правило, имеет набор из n чисел, который называется каноническим представлением. Совокупность этих представлений составляет канонический модуль. Канонические представления гео-

метрических, составных и технологических элементов приведены в Приложении 2.

Над объектами всех типов определены следующие операции:

- геометрические: сдвиги, повороты, отражения относительно прямой;
- редактирование;
- визуализация;
- выяснение канонического, объектного представления операции и атрибутов.

Кроме того, объекты каждого конкретного типа могут быть определены разными способами. Так, например, точки могут быть определены при помощи явного задания их координат (ТКАН (x, y)), как пересечение двух кривых или отрезков (ТО(F_1, F_2)), как начальная, конечная точка объекта (ТНА (F), ТКО (F)) и т.д. Полный перечень операций языка приведен в Приложении I.

7. Некоторые вопросы реализации

В процессе разработки программ трансляторов языка ГРАФИТ был разработан ряд алгоритмов машинной графики, имеющих общее значение.

7.1. Смешанные координаты

При работе с элементами чертежа типичными являются операции поэлементного сдвига, поворота, зеркального отражения. Трансформируя элемент, можно поступать двояко - либо менять представление элемента, либо хранить представление и связывать с ним параметры преобразования.

Каноническое представление элементов в ГРАФИТе организовано следующим образом:

$$K_g = \{(x, y, \varphi), \{k_i\}\},$$

где (x, y, φ) - так называемый вектор привязки, а $\{k_i\}$ - собственно каноническое представление (например, для отрезка $K_S = \{(x, y, \varphi), \{l\}\}$, где l - длина отрезка).

Такое представление позволяет при универсальных преобразованиях, описанных выше, менять единообразно лишь значения компонент вектора привязки, не меняя собственно канонического представления.

При этом вычисление новых значений компонент вектора привязки осуществляется по очень простым и экономичным, с точки зрения количества арифметических операций, формулам.

Например, поворот элемента с вектором привязки (x, y, φ) на угол φ_{Π} вокруг точки (x_T, y_T) :

$$\begin{aligned}x' &= c * (x - x_T) - s * (y - y_T), \\y' &= s * (x - x_T) + c * (y - y_T), \\ \varphi' &= \varphi + \varphi_{\Pi},\end{aligned}\quad (1)$$

где $c = \cos(\varphi_{\Pi})$, а $s = \sin(\varphi_{\Pi})$;

Отражение относительно прямой (x_p, y_p, φ_p) :

$$\begin{aligned}x' &= (s^2 - c^2)(y - y_p) + c \cdot s(x - x_p) + x_p, \\y' &= (c^2 - s^2)(x - x_p) + c \cdot s(y - y_p) + y_p, \\ \varphi' &= 2 \cdot \varphi_p - \varphi,\end{aligned}\quad (2)$$

где $c = \cos(\varphi_p)$, $s = \sin(\varphi_p)$;

Формулы преобразований типа сдвига на вектор, поворота вокруг точки привязки, самопараллельного переноса имеют еще более простой вид.

Применение описанного аппарата позволило единообразно, с минимальным расходом памяти под запись канонического представления элемента, выполнять преобразования над элементами всех типов.

7.2. Аппроксимация кривых ломаными

Задачи аппроксимации плоской кривой ломанными линиями, состоящими из отрезков прямой, возникают из-за того, что многие современные устройства графического вывода не дают аппаратной возможности визуализировать кривые в их истинном виде.

Ломаные, аппроксимирующие кривую, выбираются, как правило, из следующих соображений:

1. Первый подход. $f(x)$ — аппроксимирующая кривая, а $\varphi(x)$ — аппроксимирующая ее ломаная. Тогда $\varphi(x)$ выбирается такой, чтобы выполнялось неравенство

$$\sup_{x \in \Omega} \inf_{x' \in \Omega} \| (x, f(x)) - (x', \varphi(x')) \| \leq \varepsilon ,$$

здесь Ω — область определения f и φ , а $\| \cdot \|$ — евклидова норма на плоскости.

2. Во втором подходе считается, что $x = f_1(t)$, $y = f_2(t)$, где t — некоторая параметризация кривой. В этом случае $\varphi(x)$ — ломаная из отрезков, концы которых лежат в точках $(x(t_i), y(t_i))$, а $\Delta t_{i+1} - t_i = \varepsilon$ для $i = 1, N$, где N — число звеньев ломаной.

Оба эти подхода по-разному работают для случая быстроизменяющейся и плавной кривой (рис. 7).

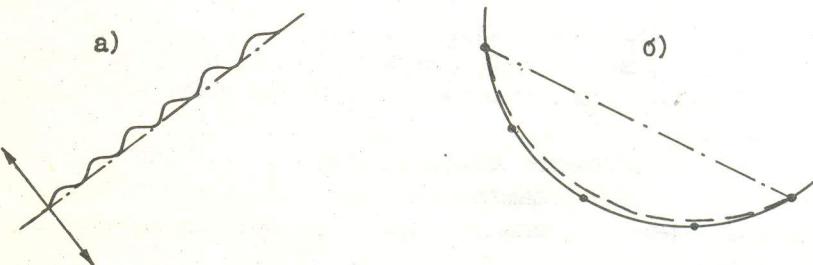


Рис. 7. Аппроксимация кривых двумя способами:

— · — — первый способ; - - - второй способ

В случае а) при задании точности аппроксимации выше амплитуды изменения кривой аппроксимирующий алгоритм "не замечает", что кривая имеет сложную форму; в случае б) на гладких кривых при аппроксимации вторым способом производятся лишние действия. В блоках транслятора ГРАФИТА, выполняющих функции аппроксимации, использованы разработанные автором комбинированные алгоритмы аппроксимации кривых второго порядка, дающие удовлетворительный эстетический эффект при возможно меньшем количестве (для заданного ϵ) звеньев аппроксимирующей ломаной и использующие индивидуальные особенности кривых.

На рис. 8 показана зависимость числа звеньев ломаной для эллипса с полуосами 70 и 40 мм для различных ϵ .

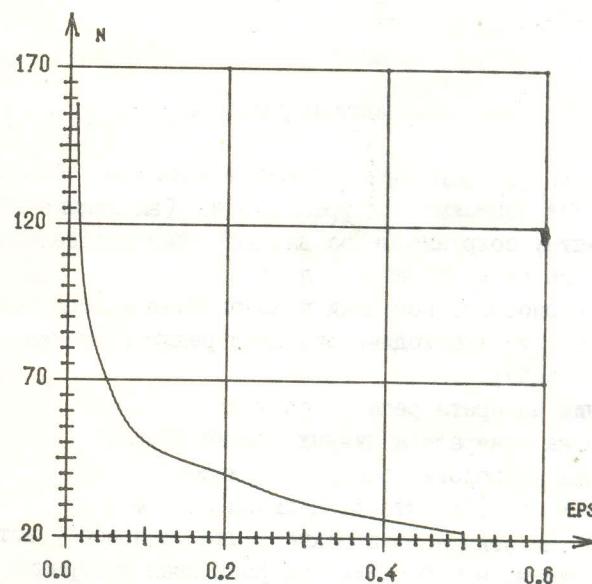


Рис. 8

8. Эксплуатационные характеристики, готовность

Все трансляторы языка реализованы на Фортране ІУ. Объем версии 1982 г. около 15 тыс. операторов. В качестве базовой графической системы при получении твердой копии использовалась система СМОГ [8], а в качестве базовой интерактивной графической системы - СПО ГД [5] для конфигураций БЭСМ-6 - АРМ-Р, М4030 : АРМ-Р и СМОГ [9] для конфигурации М4030 : Электроника 100Н с дисплеем ДЕЛЬТА-М. Время отклика составляет около 3-5 с астрономического времени при "пустых" больших ЭВМ и увеличивается в зависимости от их загрузки другими задачами. Время ЦП составляет доли секунды.

Ввод, редактирование и выпуск реальных чертежей с помощью средств языка ГРАФИТ осуществляется быстрее, чем вручную.

Разработка описываемой версии языка заняла около 6 человеко-лет.

9. Перспективы развития

Планируется развивать язык ГРАФИТ в следующих направлениях:

- углубление возможностей фрагментации (вложенные друг в друга фрагменты, сохраняемые на внешней памяти отдельные фрагменты, а не весь чертеж и т.д.);
- развитие способов передачи пользователю информации о связях элементов (это необходимо для нужд редактирования без разрушения связей);
- упрощение аппарата редактирования;
- разработка генератора нужных версий языка;
- улучшение технологии работы с языком;
- перенос всех трансляторов целиком на АРМ.

В разработке эскизного проекта языка активное участие принимал В.Ю.Шилков; В.М.Голубев разрабатывал программы задания технологических данных и осуществлял перенос ГРАФИТА с БЭСМ на ЕС; Н.С.Шулта разработал программы диалогового интерпретатора языка.

Автор благодарит сотрудников и заведующего лаборатории

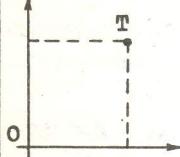
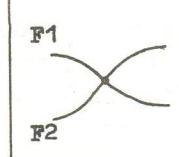
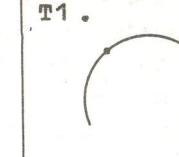
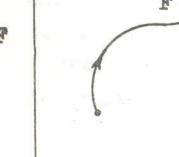
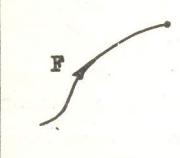
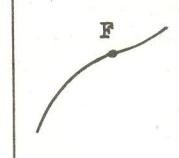
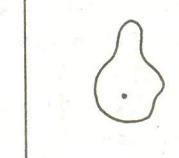
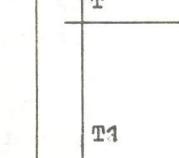
Л и т е р а т у р а

1. Система проектирования, построения чертежей и изготовления ДДМ/. ВЦП. - Б-3749. - 38 с.
2. Тодорой Д.Н. Принципы построения расширяемых систем машинной графики. - В сб.: Программное обеспечение банков данных: Материалы семинара. М., 1979, с. 71-73.
3. Сиротин В.Г. ГРАФИТ - СПИП для автоматизации ряда проектных и конструкторских работ в машиностроении. - В сб.: Проблемы машинной графики. Новосибирск, 1982, с.103-110.
4. Горелик А.Г. Автоматизация инженерно-графических работ с помощью ЭВМ. - Минск:Высшая школа,1980.
5. Дебелов В.А., Мацокин А.М., Чубарев А.И. СПО ГД - система программного обеспечения графического диалога. - В сб.: Проблемы машинной графики. Новосибирск, 1982, с. 64-71.
6. Cavagna C., Cugini U. Data - structure for the description and handling of engineering drawings. - Comput. Aided Des., 1977, v.9, no. 1, p. 17-22.
7. Магариу Г.А. Выбор форм представления входной программы, сохраняемых на весь период диалога. - В сб.: Системное и теоретическое программирование. Кишинев, 1982, вып.69, с.55-61.
8. Математическое обеспечение графопостроителей, СМОГ I уровень: Инструкция по программированию. - Новосибирск, 1976.- 118 с. - В надзаг.: ВЦ СО АН СССР.
9. Торшин В.И., Труфанов О.Д. СМОД для ЭВМ М-4030. - В сб.: Машинная графика и ее применение. Новосибирск, 1979, с. 98-102.

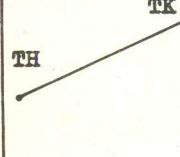
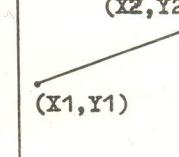
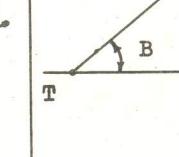
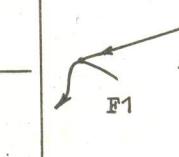
Приложение I

Перечень операторов языка ГРАФИТ

I. Операторы определения точек

Каноническое определение	Общая точка двух элементов	Ближайшая к данному элементу	Начальная точка
 $T = \text{TKAH}(x, y)$	 $T = \text{TO}(\bar{F}_1, \bar{F}_2)$	 $T = \text{TMIN}(\bar{F}, \bar{T}_1)$	 $T = \text{THA}(\bar{F})$
Конечная точка	"Средняя" точка	Опорная "центральная" точка	Вершина квадрата
 $T = \text{TKO}(\bar{F})$	 $T = \text{TAB}(\bar{F}, A, B)$	 $T = \text{TC}(\bar{F})$	 $T = \text{TBK}(\bar{T}_1, \bar{T}_2)$

2. Операторы определения отрезков

По начальной и конечной точкам	По координатам	Каноническое определение и	Касательный отрезок
 $S = \text{SHK}(\bar{TH}, \bar{TK})$	 $S = \text{SYI}_{x, y}(x_1, y_1, x_2, y_2)$	 $S = \text{SKAH}(\bar{T}, B, H)$	 $S = \text{SKAC}(\bar{F}_1, \bar{F}_2)$

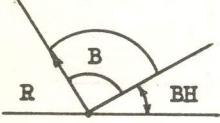
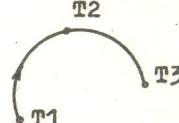
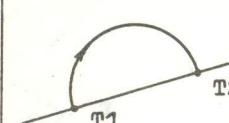
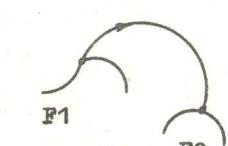
3. Операторы определения прямых

<p>через точку, под углом к прямой</p> <p>$P = PTA(T, A)$</p>	<p>Вертикальная прямая</p> <p>$P = PY(X)$</p>	<p>Горизонтальная прямая</p> <p>$P = PX(Y)$</p>
--	--	--

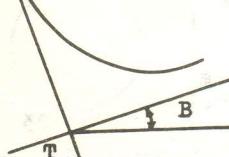
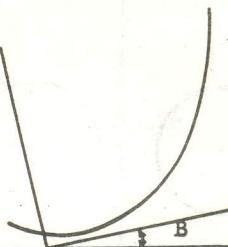
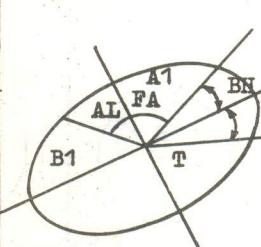
4. Операторы определения окружностей

Каноническое определение	По содержащей- ся дуге	Через касание	По трем точкам
<p>$A = AKAH(T, R)$</p>	<p>$A = AD(\overline{D})$</p>	<p>$A = AKAC(\overline{C}, \overline{F})$</p>	<p>$A = A3T(T1, T2, T3)$</p>

5. Операторы определения дуг окружностей

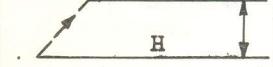
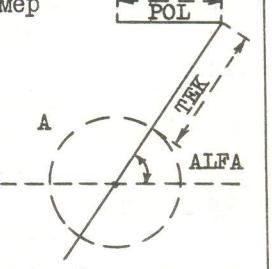
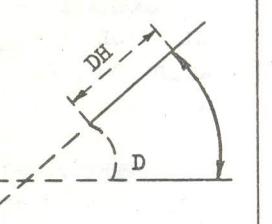
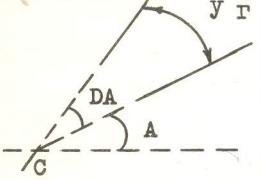
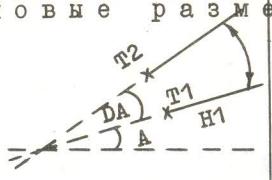
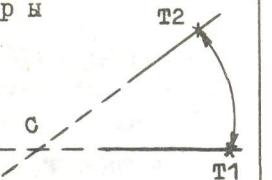
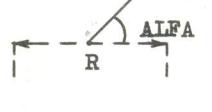
Каноническое определение  $D = DKAH(\bar{T}, BH, R, B)$	По трем точкам  $D = D3T(\bar{T}1, \bar{T}2, \bar{T}3)$	Полуокружность  $D = DK(\bar{T}1, \bar{T}2)$
Через начальную, конечную точки и радиус  $D = DHKR(\bar{TH}, \bar{TK}, R)$	Дуга, касательная к двум элементам  $D = D2FR(\bar{F}1, \bar{F}2, R, i)$	

6. Операторы определения дуг кривых второго порядка

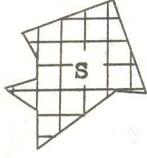
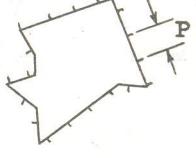
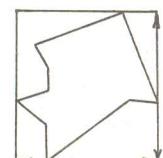
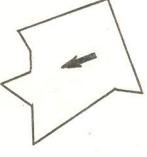
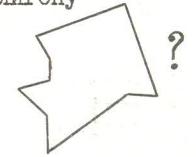
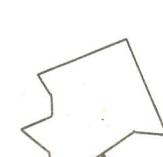
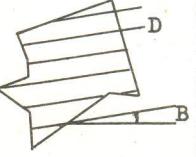
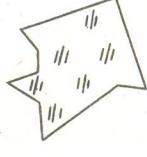
Дуги гиперболы  $D = DGIP(T, B, XH, DX, P)$	Дуга параболы  $D = DPAR_x(T, B, XH, DX, A1, B1, C1)$	Дуга эллипса  $D = DAL_{\alpha}(T, B, A1, B1, BN, ALFA)$
---	---	---

7. Операторы работы с фрагментами
 $W = HFRAG1(X, Y, F, AX, AY)$
 CALL EFRAG1
8. Операторы работы с полигонами
 $W = HPOL(X, Y, F, AX, AY)$
 CALL KPOL
9. Операторы работы с текстами
 CALL SRIET (H, K)
)
 $Z = CTPTEK(T, POV,$
10. Операторы определения размеров
 CALL RPBP(N)
 CALL RCP(N, A)
- начало фрагмента,
 - конец фрагмента.
- начало полигона .
 - конец полигона .
- определение размера и
 типа шрифта ,
- определение строки.
- определение режима выво-
 да размеров ,
- модификация системных
 параметров ,
- $P1 = PAM2(N, NZON)$
 $P2 = PAM2(N, NZON)$
 $P3 = PAM3(N, NZON)$
- } - рамки чертежные,
- $P = PNAD(R, CM, NZ; 'СТРОКА')$ - размерная надпись ,
 $P = PDOP(R, CM, 'СТР1', NZ, 'СТР2', TIP, 'СТР3', 'СТР4')$
- размерная надпись с пре-
 дельными отклонениями (до-
 пусками),
- $P = RSHT(T, POV, ITIP, POL, 'СТРОКА')$
- $P = RSHT(S, K, CM, ITIP, POL, 'СТРОКА')$
- $P = RPT(T, POV, DL, FP, POL, NZ, RC)$
- } шероховатости,
 - размерная надпись общего
 вида.

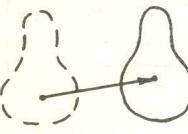
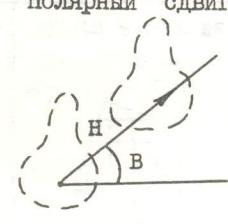
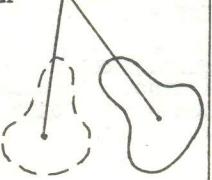
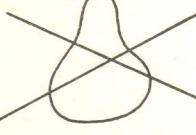
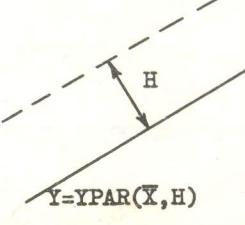
10. Операторы определения размеров

Линейный размер	Диаметральный размер	Угловой размер
 $R=RLS(S, ALFA, H, POL, TEK)$ $R=RL2T(T1, T2, ALFA, ...)$ $R=RLXY(X1, Y1, X2, Y2, ALFA, ...)$ $R=RDS(S, ALFA, ...)$ $R=RD2T(T1, T2, ALFA, ...)$ $R=RDXY(X1, Y1, X2, Y2, ALFA, ...)$	 $R=RDK(A, ALFA, ICTP, TEK, POL)$	 $R=RYD(D, DH, H1, H2, ICTP, TEK)$
Угловые размеры		
 $R=RYCRA(C, R, A, DA, H1, H2, ICTP, TEK)$ $R=RYXY1(XC, YC, X1, Y1, X2, Y2, H1, ...)$	 $R=RY2TA(T1, T2, A, DA, H1, ICTP, TEK)$ $R=RYXY2(X1, Y1, X2, Y2, A, ...)$	 $R=RYCC2T(C, T1, T2, H1, H2, ICTP, TEK)$ $R=RYXY3(XC, YC, X1, Y1, X2, Y2, H1, ...)$
Радиальные размеры		Размерная полка
 $R=RRD(D, DL, POL)$	 $R=RRK(A, ALFA, DL, POL)$	 $R1=RPR(R, ALFA, DL, POL)$

II. Операторы определения и работы с полигонами

Вычисление площа-ди 	Вычислениеperi-метра 	Вычисление габа-ритов 
CALL SPOL(\bar{X}, S)	CALL PPOL(\bar{X}, P)	CALL XYPOL(X, X_{GAB}, Y_{GAB})
Вычисление центра тяжести 	Вычисление ини-дентности(принад-лежности) точки полигона 	Вычисление рассто-яния до полигона 
CALL DENPOL(\bar{X}, X_C, Y_C)	CALL INCPOL(W, X, Y, A)	CALL RPOL(W, X, Y, R)
Штриховка полигона 		Текстурирование по-лигона 
CALL STPPOL(\bar{X}, D, B)	CALL TEKPOL(\bar{X}, i)	

12. Универсальные операторы

Сдвиг на вектор  $Y = YCN(\bar{X}, DX, DY)$	Сдвиг в точку  $Y = YCT(\bar{X}, T)$	"Полярный" сдвиг  $Y = YCP(\bar{X}, B, H)$
Вращение вокруг центра  $Y = YVC(\bar{X}, B)$	Вращение вокруг точки  $Y = YVT(\bar{X}, T, B)$	Зеркальное отражение  $Y = YOTP(\bar{X}, F)$
Уничтожение объекта  $CALL YH(\bar{X})$	Изменение ориентации  $Y = YMIN(\bar{X})$	Установка типа линии  $CALL YTl(\bar{X}, i)$
Исправление $CALL YHEB(\bar{X})$	Самопараллельный перенос  $Y=YPAR(\bar{X}, H)$	Выдача канонического представления $CALL YKAM(\bar{X}, \hat{A}, \hat{N})$
$CALL YOBM(\bar{X}, \hat{A}, \hat{N})$		

13. Системные операторы

CALL CBK(i ШИФР)	- инициализация,
CALL TOK	- трансляция Объектных пред- ставлений в Канонические,
CALL BBP(i)	- включение режима вывода,
CALL KAH(i)	- настройка канала вывода,
CALL BB	- вывод уже определенных эле- ментов,
CALL KOH	- окончание работы,
CALL СКОР(X,Y,AX,AY)	- задание системы координат,
CALL FORMAT(N,M,K)	- задание формата листа,
CALL TAP(EPS)	- установка точности аппрокси- мации кривых,
CALL COX(NAME)	- сохранить чертеж,
CALL GIVE(NAME)	- начать работу с чертежом, который был сохранен,
CALL POMT(NH,NK)	- включение режима печати ОМ в процессе счета,
CALL PKMT(NH,NK)	- включение режима печати КМ в процессе счета,
CALL POM(NH,NK)	- печать листинга ОМ,
CALL PKM(NH,NK)	- печать листинга КМ.

Приложение 2

Канонические представления

	тип	2	3	4	5	6	7	8	9	10	II	II	I3
Точка	1	X	Y	∅	DL								
Отрезок прямой	2	XH	YH	F									
Прямая	3	X	Y	F	R								
Окружность	4	X	Y	∅.	R								
Дуга окружности	5	X	Y	F	A	DF							
Эллипс	7	X	Y	F	A	B							
Дуга эллипса	8	X	Y	F	A	B	FH	DF					
Дуга параболы	12	X	Y	F	XH	DX	A						
Дуга гиперболы	13	X	Y	F	XH	DX	B		C				
Размер и тип приёма	3I	H	K										
Режим нанесения Р.Н.	32	IVLD											
Др. систем. параметры	33	N	A										
Линейный размер	5I	X	Y	F	DL	B1	B2	H1	H2	ICTR	TEK	NZ	APL
Угловой размер	52	X	Y	F	R	DF	H1	H2	ICTR	TEK	ARD	RC	ARV°
Диаметральный размер	53	X	Y	F	R	ICTR	TEK	POL	ICTR	NZ			
Размерные полки	54	X	Y	F	DL	B	POL	ITIP	POL	NAT			
Шероховатости	55	X	Y	F	ITIP	POV	NZ						
Строки текста	56	X	Y	NZN									
ФОРМА 1	57	N	NZN										

	ТМII	2	3	4	5	6	7	8	9	10	II	I2	I3
ФОРМА 2	58	N	NZON										
ФОРМА 2A	59	N	NZON										
Рациональный размер 1	60	X	Y	F	R	ТЕК	POL	N					
Рациональный размер 2	61	X	Y	F	R	DL							
Допуск	62	X	Y	POV	NA1	NZ	NA2	ITIP	NA3	NA4			
Начало фрагмента	71	X	Y	F	AX	AY							
Конец фрагмента	72	W	Y	F	AX	AY							
Начало полигона	75	X	W										
Конец полигона	76	W											

Примечание: W - константа специального вида, W = 289I.289I

Н.С.Шупта

ИНТЕРАКТИВНЫЙ ИНТЕРПРЕТАТОР ВХОДНОГО ЯЗЫКА ПАКЕТА ПРИКЛАДНЫХ ПРОГРАММ ГРАФИТ

I. Введение

В данной работе описывается интерактивный интерпретатор входного языка ППП ГРАФИТ [1], обладающего мощными и удобными средствами по описанию плоских геометрических объектов и выпуску машиностроительных чертежей. Тем самым фактически закончена разработка графического диалогового языка ГРАФИТ – одна из первых в данной области.

Интерпретатор предназначен для использования в автоматизированных системах проектирования, подготовки чертежно-конструкторской документации и технологической подготовки производства.

С точки зрения конструктора-проектировщика или технолога, диалоговый режим работы с ППП ГРАФИТ, по сравнению с пакетным, имеет следующие преимущества:

- возможность визуально контролировать процесс создания и редактирования чертежей;
- язык программирования заменен входным языком (ВЯ), более простым и наглядным.

Директивы (предложения) входного языка позволяют пользователю описывать и редактировать чертеж в терминах составляющих элементов (точки, дуги, размеры и т.д.). Семантика директив ВЯ определяется программами ППП ГРАФИТ.

При разработке входного языка учитывалось, что число ошибок, совершаемых пользователем, существенно зависит от выбора мнемоники и длины директив.

Примеры директив:

TO - F1, F2 - общая точка элементов F1 и F2,
RL2T - T1, T2, A, H, C, T - простановки размера к двум
точкам T1 и T2.

2. Интерпретация директив графического языка

Общая схема интерпретации директив интерактивного графического языка приведена на рис. I.

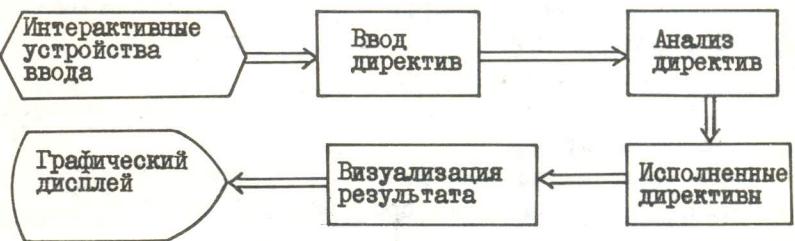


Рис. I

Программы ввода директив осуществляют ввод директив пользователя от различных вводных устройств и преобразование их в стандартную форму (например, текстовую), используемую при анализе.

Анализ и исполнение директив может осуществляться традиционным для неграфических систем способом [2, 3].

Результаты исполнения директив должны быть преобразованы в программу дисплейного процессора и выведены на экран графического дисплея (ГД).

3. Диалог с ППП ГРАФИТ

3.1. Организация диалога

При использовании описываемой версии интерпретатора диалог с ППП ГРАФИТ осуществляется на аппаратном комплексе АРМ-Р – большая ЭВМ (БЭСМ-6 или ЕС ЭВМ) с помощью средств системы СПО ГД [4]. В качестве входного устройства для ввода директив при разработке интерпретатора была выбрана алфавитно-цифровая клавиатура (АЦК) алфавитно-цифрового дисплея (АЦД), входящего в состав АРМа. Схема диалога приведена на рис. 2.

Запрос на ввод директивы ВЯ, формируемый интерпретатором, с помощью СПО ГД передается на АРМ, декодируется, и на экране АЦД появляется приглашение ввести директиву: —> .

Введенная директива передается на большую ЭВМ и поступает на вход интерпретатора.

После анализа введенной директивы, в случае ее корректности, интерпретатор запускает программу ППП ГРАФИТ, исполняющую директиву.

Исполняющая программа возвращает результат интерпретатору.

Интерпретатор оформляет результат в виде графического сегмента СПО ГД, полученного с помощью графической системы СМОГ [5]. Затем генерирует имя и приписывает **это имя** сформированному сегменту. В дальнейшем полученный сегмент, с точки зрения СПО ГД, идентифицируется этим именем. В частности, под этим именем он помещается в архив графических сегментов. После этого интерпретатор выдает СПО ГД приказ добавить в дисплейный файл сформированный сегмент из архива сегментов.

Переданный сегмент появляется на экране ГД.

3.2. Функции интерпретатора

Приведенное выше описание диалога показывает, что к традиционным функциям интерпретатора – анализу и исполнению директив – добавляются функции, **связанные с инициированием устройств ввода и генерации дисплейного файла** [6, 7].

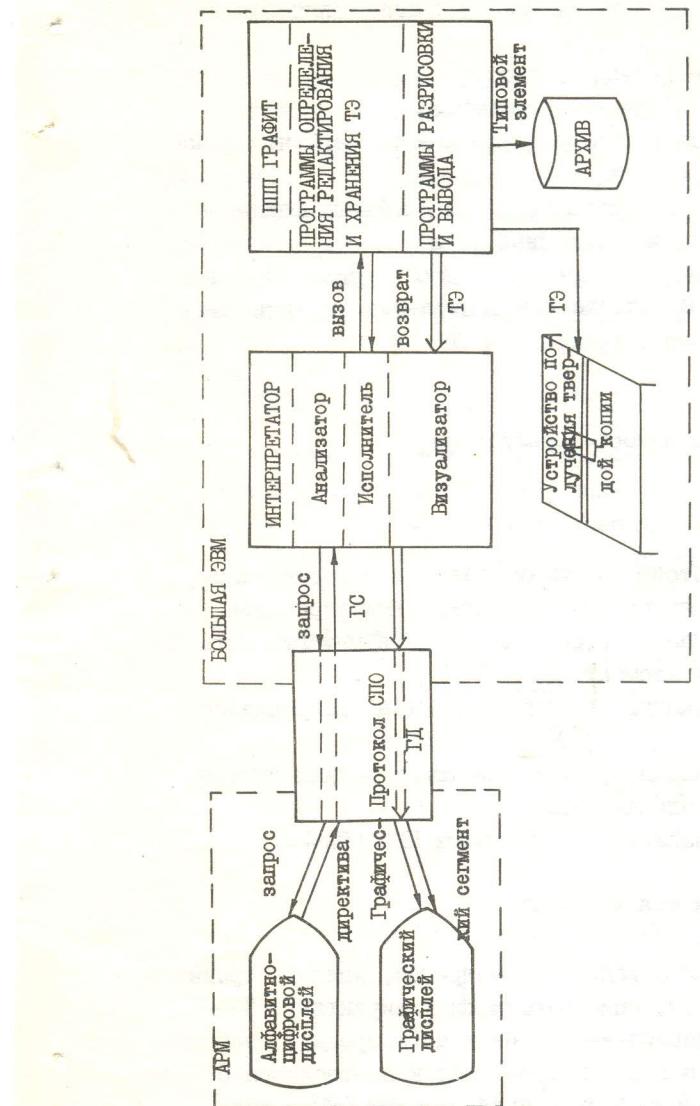


Рис. 2 Общая схема работы интерпретатора:
 ТС - графический сегмент,
 ТЭ - типовой графический элемент

Приведем полный список дополнительных функций интерпретатора:

- формирование запроса ввода директивы;
- генерация имени графического сегмента;
- оформление результата в виде графического сегмента и помещение его в архив сегментов;
- удаление графического сегмента из дисплейного файла;
- добавление сегмента в дисплейный файл.

Имя графического сегмента генерируется с помощью счетчика порожденных сегментов. Остальные дополнительные функции реализуются в виде обращения к программам СПО ГД.

4. Особенности реализации

4.1. Описание языка

Каждая директива состоит из имени операции и списка операндов, разделенных запятыми или пробелами. Имя директивы отделяется от списка операндов пробелом. Директивы могут быть:

- описаниями типовых элементов;
- директивами модификации типовых элементов (исправление, уничтожение);
- системными директивами, устанавливающими режимы генерации или вывода типовых элементов.

Семантика директив задается программами ППП ГРАФИТ.

4.2. Описание синтаксиса директив

Синтаксис директив ВЯ описывается матрицей, каждая строка которой является вектором, описывающим одну директиву. Первый элемент вектора содержит имя директивы (операции), остальные - число и список типов параметров. Под имя отводится 6 байт. Фактически имя занимает одну ячейку, в ЕС ЭВМ - два слова памяти.

Примеры описания директив приведены в таблице.

ИМЯ	N	TP1	TP2	TP3	TP4	TP5	TP6	TP7	TP8	TP9	TP10
RL2T	6	3	3	I	I	I	I	Ø	Ø	Ø	Ø
RNAD	4	3	I	I	2	Ø	Ø	Ø	Ø	Ø	Ø
RYXY3	10	I	I	I	I	I	I	I	I	I	I
SHK	2	3	3	Ø	Ø	Ø	Ø	0	0	Ø	Ø

Примечание: 3 - параметр - типовой элемент; 2 - текстовой параметр; I - числовой параметр; Ø - параметр отсутствует.

Число и тип параметров хранятся в упакованном виде и занимают одно слово ЕС ЭВМ или ячейку памяти БЭСМ-6.

4.3. Эксплуатационные характеристики

Входной язык разрабатывался контекстно-независимым. Это позволило реализовать анализатор как однопросмотровый без возвратов. Время анализа пропорционально длине директивы [8].

Программы интерпретатора написаны на языке Фортран. Объем программ составляет приблизительно 2000 операторов.

Скорость анализа на ЭВМ ЕС 1050 составляет приблизительно 50 директив в секунду при средней длине директивы в 30 символов.

Время исполнения директивы существенно зависит от загруженности большой ЭВМ и скорости обмена по каналу связи.

5. Пример работы в диалоге

После запуска диалога пользователь получает приглашение ввести директиву. Ниже приведен простой пример диалога с ППП ГРАФИТ. Вводимые директивы выделены прописными буквами и под директивами дана их расшифровка.

—>
ТКАН 5Ø, 5Ø

(определение точки с координатами $5\varnothing$, $5\varnothing$);

—>
ТКАН $II\varnothing$, $5\varnothing$

—>
АКАН TI , $I\varnothing$

(определение окружности с центром в точке I радиуса $I\varnothing$);

—>
АКАН $\sqcup 20$, $I5$

~~*** ОШИБКА *** НЕОПРЕДЕЛЕННЫЙ ЭЛЕМЕНТ~~

АКАН $\sqcup 20$, $I5$

++++ + -----

} ответ
интерпретатора;

—>
АКАН $\sqcup T2$, $I5$

—>
SKAC $\sqcup A3$, $A4$

(отрезок, касательный к окружностям $A3$ и $A4$);

—>
SKAC $\sqcup A4$, $A3$

—>
RL2T $\sqcup TI$, $T2$, \varnothing , 25 , \varnothing , \varnothing

(линейный размер между точками TI и $T2$);

—>
RDK $\sqcup A3$, $I35$, \varnothing , $I\varnothing$

(диаметральный размер окружности 3);

—>
RDK $\sqcup A4$, 45 , \varnothing , $I\varnothing$.

К этому моменту диалога на экране графического дисплея появится изображение, показанное на рис. 4;

—>
BVP $\sqcup 2$

(установка "чистового" вывода чертежа);

—>
КАН $\sqcup I$

(переключение на графопостроитель);

—>
BB
(вывод на графопостроитель);

KOND

(очистить экран, закончить диалог)
КОНЕЦ СЕАНСА.

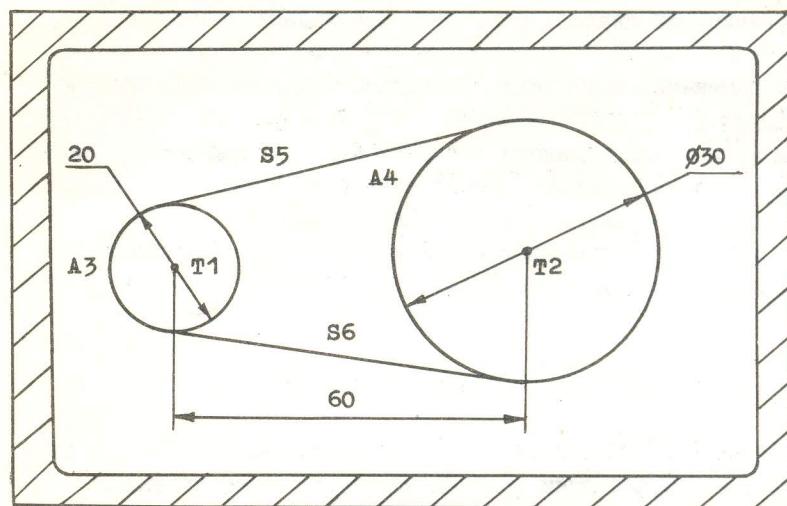


Рис. 4

После этого сеанса диалога на графопостроитель будет выведен чертеж, представленный на рис. 5.

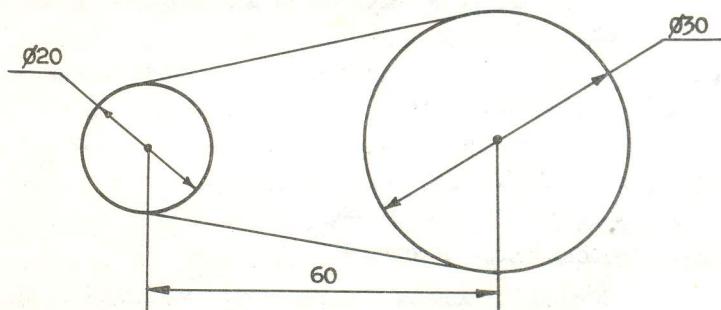


Рис. 5

6. Заключение

Описываемый интерпретатор может служить модельным примером организации диалога с пакетом прикладных программ в рамках системы СПО ГД.

При изменении мнемоники необходимо изменить лишь список имен директив.

Для организации диалога с другим пакетом прикладных программ необходимо заменить список имен и паспортов программ, а также список, обращенный к программам пакета.

Интерпретатор применяется при решении чертежно-конструкторских задач на ряде предприятий СССР.

Л и т е р а т у р а

1. Сиротин В.Г. Язык ГРАФИТ для описания, редактирования, хранения и визуализации моделей двумерных геометрических объектов и их чертежей. - Настоящий сб., с.63-91.
2. Пратт Т. Языки программирования: разработка и реализация. - М.: Мир, 1979.
3. Грис Д. Конструирование компиляторов для цифровых вычислительных машин. - М.: Мир, 1975.
4. Дебелов В.А., Мацокин А.М., Чубарев А.И. СПО ГД - система программного обеспечения графического диалога. - В сб.: Проблемы машинной графики. Новосибирск, 1982, с. 64-71.
5. Математическое обеспечение графопостроителей СМОГ. I уровень. Инструкция по программированию / Под ред. Ю.А.Кузнецова. - Новосибирск: Б.и., 1976. ВЦ СО АН СССР.
6. Ньюмен У., Спрулл Р. Основы интерактивной машинной графики.-М.: Мир, 1976.
7. Гилой В. Интерактивная машинная графика.- М.:Мир, 1981.
8. Ахо А., Ульман Дж. Теория синтаксического анализа, перевода и компиляции. - М.: Мир, 1978.