

**АВТОМАТИЗАЦИЯ
ЭКСПЕРИМЕНТА
И
МАШИННАЯ ГРАФИКА**

**АВТОМАТИЗАЦИЯ ЭКСПЕРИМЕНТА
И
МАШИННАЯ ГРАФИКА**

**Издательство Томского университета
Томск - 1977**

В сборнике представлены работы, выполненные в лаборатории вычислительных систем Сибирского физико-технического института в области автоматизации научного эксперимента и машинной графики. Описывается общая структура и математическое обеспечение автоматизированной системы САФРА для регистрации, хранения, поиска и обработки данных, реализующей комплексную автоматизацию эксперимента - от регистрации электрических сигналов до получения готовых результатов в виде таблиц, графиков функций, диаграмм. Сборник предназначен для инженеров и научных работников, связанных с автоматизацией научных исследований.

Редактор сборника - канд. физ. мат. наук В.А. ФИЛОНЕНКО

АВТОМАТИЗИРОВАННАЯ СИСТЕМА РЕГИСТРАЦИИ, ХРАНЕНИЯ И ОБРАБОТКИ ЭКСПЕРИМЕНТАЛЬНЫХ ДАННЫХ "САФРА"-ОБЩЕЕ ОПИСАНИЕ

Б.А. Гладких

I. Введение

Электронная вычислительная техника начинает играть все более заметную роль в экспериментальных исследованиях. Если в недавнем прошлом она использовалась в основном для обработки готовых экспериментальных результатов, а сами результаты получались вручную, либо с помощью средств "малой" автоматизации, то в последнее время ЭВМ (особенно малые ЭВМ) широко применяются в ходе самого эксперимента. Весьма важным шагом в этом направлении явилась разработка международного стандарта КАМАК, предлагающего типовые программно-управляемые модульные структуры систем автоматизации на базе мини-ЭВМ и унифицированного периферийного оборудования. Аппаратура автоматизации, выполненная в стандарте КАМАК, обладает большими потенциальными возможностями, она легко перекомпоуется по ходу исследования, при этом основная работа по сопряжению отдельных блоков между собой и по управлению процессом эксперимента ложится на программы в управляющей ЭВМ. Программист постепенно становится центральной фигурой не только на этапе обработки экспериментальных данных, но и на этапе их получения.

Таким образом, намечается путь к комплексной автоматизации

экспериментальных исследований, когда все работы, начиная от регистрации электрических сигналов и кончая выдачей готовых таблиц, графиков, диаграмм, увязаны в единый технологический процесс, управляемый с помощью ЭВМ.

При разработке системы САФРА была поставлена задача создать сравнительно простую систему автоматизации, удовлетворяющую следующим основным требованиям. Во-первых, система должна быть универсальной, способной охватить достаточно широкий круг экспериментальных исследований. Во-вторых, система должна быть комплексной, предусматривающей автоматизацию всех основных этапов работы. В-третьих, система должна быть простой для изучения и обслуживания пользователями-экспериментаторами. В-четвертых, она должна быть рассчитана на возможно простой и доступный комплект технических средств.

Система САФРА представляет собой аппаратно-программный комплекс и ее структуру можно рассматривать с двух позиций: с точки зрения технических средств и с точки зрения математического обеспечения.

С точки зрения технических средств САФРА состоит из следующих групп устройств:

- аппаратура сопряжения с экспериментом;
- управляющая ЭВМ;
- канал передачи данных;
- обрабатывающая ЭВМ;
- устройства отображения.

Математическое обеспечение более удобно классифицировать по функциональному признаку и с этой точки зрения САФРА представляет собой совокупность четырех функциональных подсистем:

- подсистема управления экспериментом;
- подсистема регистрации и хранения информации;
- подсистема обработки информации;
- подсистема отображения.

Соотношение между элементами технического и математического обеспечения системы САФРА схематически изображено на рис.1. Из рисунка видно, что границы раздела между функциональными подсистемами в основном проходят "внутри" оборудования и поэтому не являются совершенно четкими.

В данной статье дается общее описание системы, достаточное для понимания ее функциональных возможностей. Более подробно

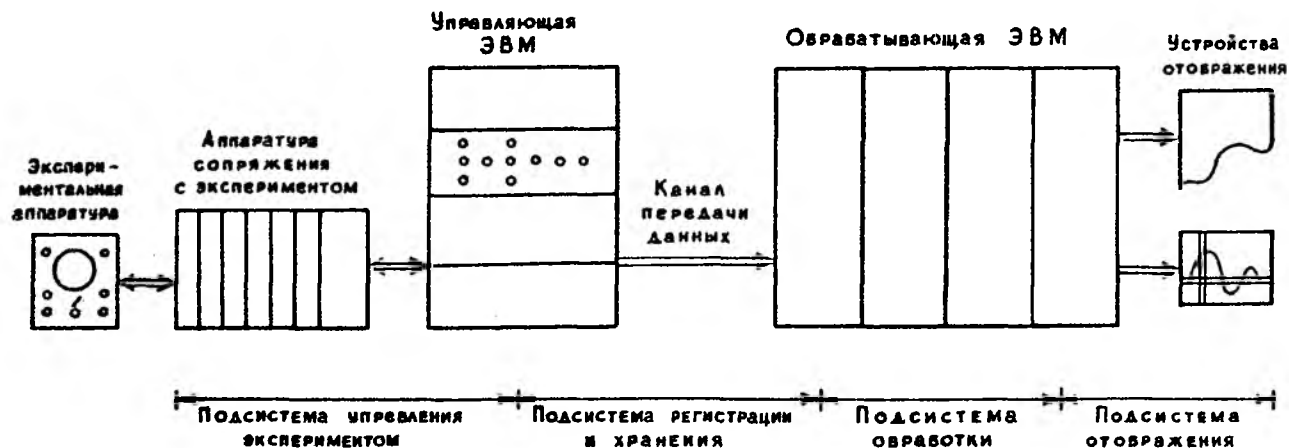


Рис. 1. Структурная схема системы СДФР

реализация отдельных подсистем описана в работах [1, 2,3,4,5] .

2. Технические средства

Аппаратура сопряжения с экспериментом выполнена на базе стандартных элементов (секций, модулей) системы КАМАК [8] . Она включает в себя служебные модули, служащие для подключения аппаратуры к управляющей ЭВМ (драйвер системы, каркасный контроллер, модуль связи с ЭВМ), а также модули, непосредственно обслуживающие эксперимент. В настоящее время различными организациями разработано очень большое число (несколько сот) модулей, осуществляющих самые различные функции: преобразование сигналов из аналоговой в цифровую форму и обратно, сопряжение с измерительной аппаратурой, отсчет времени и т.д. Пользователь, исходя из целей эксперимента и своих возможностей, должен сам определить необходимый состав модулей, обеспечивающих сопряжение регистрирующих приборов с экспериментом. В некоторых случаях имеющаяся стандартная номенклатура модулей, возможно, его не удовлетворит, тогда ему придется разработать собственные модули, наилучшим образом выполняющие требуемые функции. Однако и в этом случае выполнение рекомендаций стандарта КАМАК облегчит процесс проектирования, так как КАМАК предлагает типовые решения по конструкции, параметрам сигналов, логике работы модуля.

Управляющая ЭВМ. В качестве управляющей ЭВМ, работающей с аппаратурой КАМАК в режиме реального времени, наиболее удобна одна из получивших в последнее время большое распространение мини-ЭВМ с общей магистралью типа PDP-8 или PDP-11. В системе САФРА роль управляющей ЭВМ могут выполнять любые ЭВМ, система команд которых аналогична PDP-8, например, "Саратов", "Электроника-100", ТРА-1000 и др., при этом возможна работа в минимальном комплекте: МОЗУ емкостью 4 К слес, пультовая пишущая машинка, считыватель перфоленты, ленточный перфоратор.

Канал передачи данных. Как показал опыт, узким местом в системах автоматизации экспериментов является канал передачи данных от управляющей к обрабатывающей ЭВМ. Конечно, идеальным решением вопроса была бы немедленная передача информации по линии связи, однако такой путь не всегда возможен. Поэтому приходится производить промежуточную регистрацию данных на внешнем носителе.

Одним из самых распространенных, хотя, пожалуй, наименее удобных и надежных носителей является перфолента. В системе САФРА, рассчитанной на минимальный набор стандартного оборудования, предусмотрена возможность промежуточной регистрации данных на перфоленте, при этом для частичной компенсации недостатков, присущих такому способу, предпринят ряд мер. Во-первых, информация на выходе управляющей ЭВМ приводится к такому внешнему представлению, которое требует минимальной длины сообщений; это позволяет повысить скорость регистрации. Во-вторых, для повышения надежности записи применено специальное помехоустойчивое кодирование. Оно дает возможность восстанавливать данные даже тогда, когда перфоратор допускает серьезные ошибки: пропускает строки, совсем не пробивает отверстия в одной из дорожек и т.п.

Поскольку представление данных на внешних носителях унифицировано, имеется возможность, при наличии соответствующего оборудования, организовать канал передачи данных другим способом, например, с записью на магнитную ленту или путем прямой передачи по линии связи.

Обрабатывающая ЭВМ. Функцию обрабатывающей ЭВМ в системе САФРА может выполнять любая ЭВМ Единой системы, имеющая в своей конфигурации МОЗУ емкостью не менее 256 К, два накопителя на магнитных дисках, устройство ввода с перфоленты. Вычислительный процесс идет под управлением операционной системы ОС ЕС.

Устройства отображения. Выходная информация в системе САФРА может выводиться как на АЦПУ, так и на графопостроитель любой конструкции, имеющий стандартное подключение к ЕС ЭВМ (например, ЕС 7051, 7052, 7053).

3. Математическое обеспечение

Подсистема управления экспериментом [3] представляет собой комплекс программ в управляющей ЭВМ. Эта подсистема выполняет следующие основные функции:

- съем данных с модулей КАМАК и передача им данных, если в эксперименте предусмотрена обратная связь;
- первичную арифметическую и логическую обработку данных;
- обеспечение диалога с экспериментатором, что позволяет ему оперативно вмешиваться в процесс эксперимента, вводить

исходные данные, менять режимы работы программы.

Соответственно подсистема управления экспериментом состоит из трех блоков: арифметический блок, блок управления модулями КАМАК, и блок диалогового ввода-вывода. Первые два блока работают одинаковым образом. Каждый из них имеет собственный набор макрокоманд и интерпретатор, осуществляющий "раскрутку" макрокоманд до уровня машинных операций. Структура макрокоманд выбрана таким образом, чтобы минимизировать затраты труда на написание программы управления экспериментом. Так, в арифметическом блоке предусмотрены операции с плавающей запятой, специальные функции, операции управления (всего 24 макрокоманды). В блоке управления модулями КАМАК задействовано 20 наиболее употребительных команд обращения к модулям и обеспечена возможность выполнения остальных команд. Блок диалогового ввода-вывода дает возможность управлять ходом эксперимента с пультовой пишущей машинки. Система может выдавать на печать различные сообщения, запрашивать значения переменных, распознавать вводимые с машинки директивы и в соответствии с ними передавать управление на нужные участки программы.

Программа управления конкретным экспериментом должна быть написана на Ассемблере PDP-8 с использованием макрокоманд интерпретаторов. Такой способ, конечно, менее удобен для программиста, чем запись на языке высокого уровня, но зато позволяет получить более короткую и эффективную программу, что является исключительно важным фактором при использовании мало мощных управляющих ЭВМ.

Подсистема регистрации и хранения информации. Основная цель, поставленная при создании этой подсистемы, состояла в том, чтобы избавить пользователя от необходимости иметь дело с физическим представлением данных. Как на этапе регистрации, так и на этапе обработки, пользователь должен работать со стандартной логической структурой данных (она будет рассмотрена ниже), обращаясь к переменным просто по именам и не заботясь о форматах, точности представления и т.п. Эта цель достигается тем, что на каждый экспериментальный набор данных в системе САЭРА заводится словарь переменных, осуществляющий связь между логическим и физическим представлением. В словарях расшифровываются имена экспериментальных переменных, точность представления, форматы, единицы измерения (подробнее о представлении

данных см. [1]). С одной стороны, эти словари присутствуют во время эксперимента в управляющей ЭВМ, где по ним производится преобразование из внутреннего представления в физическое представление канала передачи данных. С другой стороны, словари хранятся вместе с данными в обрабатывающей ЭВМ, там осуществляется обратное преобразование. Как показал опыт, такой способ представляет большие удобства при пользовании системой, так как дает возможность легко перестраивать структуру данных по ходу исследования.

Подсистема регистрации и хранения информации выполняет следующие функции:

- преобразование данных из внутреннего представления управляющей ЭВМ во внешнее представление канала передачи данных;
- помехоустойчивое кодирование и регистрацию данных на перфоленте;
- ввод данных с перфоленты в обрабатывающую ЭВМ и исправление ошибок;
- преобразование данных из внешнего представления канала передачи данных во внутреннее представление обрабатывающей ЭВМ;
- создание архивных наборов данных, доступных подсистеме обработки.

Кроме того, имеется вспомогательная функция создания и ведения архива словарей.

Первые две функции реализуются программами в управляющей ЭВМ. Блок регистрации (см. [3]), тесно примыкающий к подсистеме управления экспериментом, представляет собой интерпретатор, в котором предусмотрены макрокоманды, обеспечивающие работу с данными на логическом уровне. Запись данных на перфоленту, а также считывание и исправление ошибок производятся программами помехоустойчивого кодирования [2], осуществляющими борьбу как с искажениями отдельных битов, так и со структурными нарушениями.

Работа блоков, выполняющих преобразование данных в обрабатывающей ЭВМ и создание архива данных, а также создание и ведение архива словарей, подробно описана в работе [5]. Результатом работы этих блоков являются сформированные наборы данных на магнитных носителях ЕС ЭВМ (лентах, дисках), готовые для подсистемы обработки данных.

Подсистема обработки данных. Задачи, которые ставятся при обработке экспериментальных данных, могут быть самые различные,

поэтому универсальной системы обработки нет и быть не может. Вопрос состоит в том, чтобы ценой некоторой унификации структур данных и программ реализовать более простыми способами чем те, которые возможны при использовании универсальных языков программирования, типичные, часто встречающиеся функции по обработке экспериментальных данных.

В основу подсистемы обработки данных в САФРе положена система статистического анализа САС [9]. Она представляет пользователю:

- простой язык для арифметической и логической обработки данных, рассчитанный на экспериментатора - непрограммиста;
- библиотеку стандартных процедур статистического анализа, включающую программы построения распределений, регрессионного, корреляционного, факторного анализа и т.п.;
- набор стандартных средств для создания новых процедур обработки на языках PL/1, Фортран, Ассемблер; при этом программисту предоставляется некоторый сервис, упрощающий процесс программирования выборки данных, задания режимов работы и т.п.

Давая экспериментатору широкие возможности по обработке, система САС вместе с тем накладывает жесткие ограничения на логическую структуру экспериментальных данных. Мы обсудим эту структуру несколько позже, здесь же отметим, что большинство реальных экспериментов, по-видимому, можно привести к требуемому виду и это ограничение не кажется нам принципиальным.

Более подробно применение системы САС в САФРе описано в работе [4].

Подсистема отображения предназначена для вывода результатов обработки на АЦПУ и графопостроитель. Если пользователь применяет процедуры стандартной статистической обработки, то ему, как правило, не требуется организовывать вывод результатов на печать, так как соответствующие функции реализованы в самих процедурах. В тех случаях, когда его не удовлетворяет предлагаемая форма представления, он должен написать свои процедуры вывода. Универсальные языки программирования имеют стандартные средства форматного вывода, однако для представления сложных двумерных изображений типа таблиц, графиков, диаграмм эти средства не совсем удобны. Поэтому в системе САФРА имеется специальная программа, обеспечивающая двухкоординатный вывод на АЦПУ. Эта программа с исчерпывающей полнотой описана в работе [6].

Большие возможности для наглядного представления информации предоставляет графопостроитель. Для реализации этих возможностей разработана специальная система математического обеспечения СМОГ [7]. Система СМОГ состоит из базиса и проблемно-ориентированных надстроек. Базисные процедуры выполняют основные графические функции: раскрой бумаги, изображение линий, символов, графических фрагментов. Каждая из надстроек осуществляет свою специфическую функцию. Например, имеется надстройка для изображения графиков функций одной переменной с произвольными функциональными масштабами, надстройка для изображения гистограмм.

Система СМОГ реализована в виде пакета программы, обращение к которым происходит из языков *PL/1*, Фортран или Ассемблер. Поэтому в системе САФРА вызов графических подпрограмм возможен через операторы процедур *SAS*.

4. Структура экспериментальных данных

Как уже отмечалось, комплексная автоматизация процессов измерения и обработки достигается стандартизацией структур данных. Можно говорить о логическом представлении структуры, независимо от способа регистрации, а также о различных физических представлениях.

С точки зрения логического представления результаты любого эксперимента в системе САФРА описываются прямоугольной таблицей, столбцы которой соответствуют измеряемым переменным, а строки - наблюдениям. На пересечениях стоят значения переменных в соответствующих наблюдениях. Эти значения могут быть как числовыми, так и символьными строками, в зависимости от характера переменной. Может случиться, что в некотором наблюдении отдельные переменные не измеряются вовсе, тогда соответствующие значения считаются пропущенными, они отмечаются специальным символом и обрабатываются особо.

Каждая переменная имеет свое имя, которое ей определяет пользователь; общее число переменных в эксперименте не должно превышать 254, число наблюдений не ограничивается. Совокупность наблюдений, относящихся к некоторому законченному эксперименту, образует набор данных. Каждый набор данных также

имеет имя. Эти имена стандартизованы, они имеют вид *FILE N* , где *N* - целое десятичное число. Состав переменных может меняться от одного набора данных к другому, но внутри одного набора данных все наблюдения имеют одинаковую структуру.

Определение экспериментального набора данных, то есть выбор количества регистрируемых переменных, установление их порядка, присвоение имен, проводятся пользователем на этапе подготовки эксперимента. При этом от удачного выбора структуры экспериментального набора данных могут существенно зависеть удобство и скорость обработки результатов.

Для обеспечения передачи данных от управляющей к обрабатывающей ЭВМ, например, с помощью перфоленки, необходимо определить ф и з и ч е с к о е представление структуры данных. Соответствие между логической и физической структурами устанавливается с помощью с л о в а р я п е р е м е н н ы х данного набора данных. В этом словаре для каждой переменной указываются: 1) номер, 2) имя, 3) формат физического представления, 4) принцип умолчания; кроме того, могут быть указаны единицы измерения, а также приведены комментарии.

Номер переменной служит для её идентификации в тех случаях, когда использование символьного имени неудобно или нерационально, например, при регистрации на перфоленке. Формат физического представления указывает тип переменной (символьная, цифровая целочисленная, цифровая с плавающей точкой), а также точность представления. Принцип умолчания вводится для сокращения количества информации, передаваемой по каналу передачи данных. Если экспериментальные данные содержат много повторений, то не имеет смысла при передаче каждого наблюдения указывать значения всех переменных, так как на приемном конце их можно восстановить по предыдущим наблюдениям. Предусмотрено два типа умолчания - последовательное и начальное. При последовательном умолчании пропущенное значение переменной берется из предыдущего наблюдения, а при начальном - непосредственно из словаря переменных. Использование умолчания дает возможность пользователю включать в набор данных достаточно много переменных, упрощающих процесс обработки и интерпретации результатов, таких как фамилия экспериментатора, дата эксперимента и т.п., при этом он может не опасаться, что канал связи будет перегружен несущественной информацией.

5. Система с точки зрения гользователя

Для того чтобы продемонстрировать функциональные возможности системы САФРА, мы попытаемся на простейшем примере подробно промоделировать все действия пользователя, которому нужно провести конкретный физический эксперимент.

Пусть, например, задача состоит в изучении вольтамперных характеристик полупроводниковых диодов. У экспериментатора имеется некоторое количество диодов нескольких типов. Измерения проводятся в следующем порядке: берется диод, регистрируется его тип, диод подключается к зажимам измерителя, затем на диод подается несколько фиксированных напряжений (предположим, 10) и производятся отсчеты тока. После этого берется новый диод и цикл измерений повторяется. Полученные вольтамперные характеристики нужно занести в архив и подвергнуть статистической обработке. Обработка должна заключаться в том, что сначала полученные данные классифицируются по типам диодов, далее для каждого типа в предположении, что теоретическая зависимость тока от напряжения имеет вид $I = \delta_1 u + \delta_2 u^2$ методом наименьших квадратов вычисляются оценки коэффициентов параболы δ_1 и δ_2 . На основании этих оценок должны быть построены сглаженные ("предсказанные") вольтамперные характеристики с доверительными границами на уровне вероятности 0.95.

Итак, задача поставлена. С чего должен начать экспериментатор?

Практика показывает, что правильнее всего начинать с определения логической структуры данных. В нашем конкретном эксперименте целесообразно в каждое наблюдение включать три переменных: тип диода TYPE, напряжение U и ток I . Тогда вольтамперная характеристика одного диода будет представлена в архиве десятью (по числу точек на оси U) наблюдениям. Такая именно структура данных выбрана в основном из-за удобства последующей обработки. Пример логического представления файла, получаемого в этом эксперименте, приведен в таблице I.

После того как определена логическая структура, нужно выбрать физическое представление и составить словарь переменных. Обозначим наш экспериментальный набор данных именем FILE1. Словарь переменных этого набора данных может быть, например, таким (см. таблицу 2).

Таблица 1

Наблюдение	TYPE	U	I	
1	D 20I	0	0	Первый цикл измерений
2	D 20I	I	0.19	
...	...	...	...	
10	D 20I	10	2.64	Второй цикл измерений
11	D 202	0	0	
12	D 202	I	0.34	
...	...	...	...	Третий цикл измерений
20	D 202	10	4.57	
21	D 20I	0	0	
22	D 20I	I	0.21	Третий цикл измерений
...	...	...	...	
30	D 20I	10	2.59	
...	...	...	...	

Таблица 2

Номер переменной	И м я	Формат	Принцип умолчания	Ед. изм.	Примечание
1	TYPE	F (4)	D 20I		Тип ухода
2	U	E (2,4)	-	δ	Напряжение
3	I	E (2,4)	-	α	Ток

Описатель формата F (4) указывает на то, что данная переменная имеет символьный вид, длина её составляет 4 байта. Формат E (2,4) определяет числовую переменную в плавающем формате, причем на перфоленге её значение занимает 2 байта, в том числе 4 двоичных разряда на представление порядка. Вторая и третья переменные допускают последовательное умолчание (на это указывает прочерк), а переменная TYPE подразумевает начальное умолчание. То есть, когда эта переменная не регистрируется, система сама полагает тип ядра равным D20I.

После того как зафиксирован словарь переменных, можно разрабатывать схему эксперимента и подобрать необходимый состав модулей. В нашем случае весьма естественной является схема, изобра-

женная на рис.2. Кроме общесистемных модулей, обеспечивающих связь с ЭВМ, в секцию КАМАК включаются два функциональных модуля: цифро-аналоговый преобразователь (ЦАП), задающий напряжение на диоде, и аналогово-цифровой преобразователь (АЦП), регистрирующий величину тока. Расположим эти модули для определенности, в первой и второй позициях секции. Ввод типа диода в данном эксперименте целесообразно делать с пультавой пишущей машинки, машина будет сама запрашивать эту переменную перед началом измерений.

Конкретизировав схему эксперимента, приступаем к написанию программы в управляющей ЭВМ. Как отмечалось выше, подсистема управления экспериментом предоставляет программисту достаточный сервис при написании программ, так что программы для не слишком сложных экспериментов получаются достаточно простыми и обзорными. На рис.3 приведен текст программы, обеспечивающей наш эксперимент, с подробными пояснениями.

Результатом эксперимента будет рулон (или несколько рулонов) перфоленты с данными, причем информация закодирована помехоустойчивым кодом.

С помощью специальных программ, входящих в подсистему регистрации и хранение информации, эта перфолента должна быть введена в обрабатывающую ЭВМ, перекодирована и записана в архив в виде набора данных с именем *FILE1*. Предварительно в обрабатывающую ЭВМ также посредством специальных программ вводится словарь переменных, поэтому при обработке пользователь не должен заботиться о форматах переменных, возможном умолчании, он может обращаться к переменным своего набора данных просто по именам.

Итак, экспериментальные данные занесены в архив и можно приступить к их обработке. На этом этапе пользователь прежде всего должен решить, устраивают ли его стандартные процедуры статистической обработки, входящие в библиотеку SAS, или он должен писать свои. В рассматриваемом нами примере возможностей библиотечных процедур вполне достаточно: имеется специальная процедура *REG*. [4], осуществляющая регрессионный анализ и обладающая широким спектром возможностей. При этом программа обработки на языке SAS получается предельно простой. Мы её приводим полностью вместе с управляющими картами операционной системы.

```
// EXAMPLE      JOB   037805, 'ГЛАДКИХ'  
//              EXEC  SAFRA
```

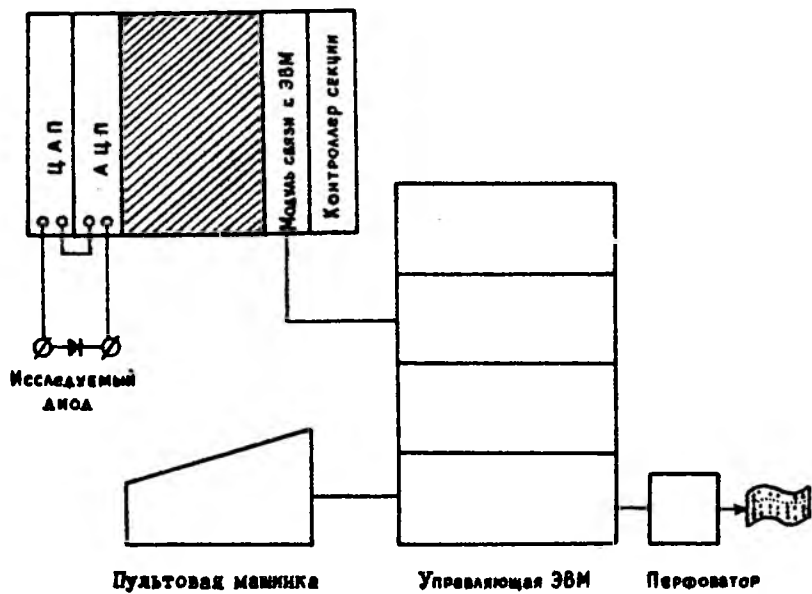



Рис. 2. Схема эксперимента.

```

// SAFRA. SYSIN DD *
DATA ;
SET FILE 1 ;
U2 = U * U ;
PROC REGR ;
BY TYPE ;
MODEL I = U U2 / PREDICTED CLM ;
TITLE 'ВОЛЬТАМПЕРНАЯ ХАРАКТЕРИСТИКА' ;
//

```

Программа содержит четыре оператора SAS. Оператор DATA объявляет начало программы; оператор SET вызывает из архива набор данных с именем FILE1; арифметический оператор присваивания образует новую переменную U^2 , сводящую квадратичную регрессию к линейной, и, наконец, оператор процедуры PROC вызывает процедуру линейного регрессионного анализа. Следующие за вызовом карты являются спецификациями и определяют режим работы процедуры. Так, спецификация BY говорит о том, что регрессию нужно рассчитывать для каждого типа диода в отдельности, спецификация MODEL определяет модель вида $I = \beta_0 + \beta_1 U + \beta_2 U^2$, причем на печать должны быть выведены "сглаженная" зависимость $I(U)$ и доверительные интервалы для неё (указания PREDICTED и CLM).

Результаты работы программы выдаются на АЦПУ с заголовком "Вольт-амперная характеристика". На рис. 4 приведена часть результатов, напечатанных этой программой. Для каждого из коэффициентов регрессии кроме оценки выводится его стандартная ошибка, а также проверяется статистическая гипотеза о равенстве этого коэффициента нулю. Например, для коэффициента β_1 эта гипотеза выглядит весьма правдоподобной (соответствующая статистика T мала, следовательно, вероятность $P(t > T/N_0)$ велика). Поэтому зависимость тока от напряжения можно аппроксимировать более простой зависимостью $I = \beta_0 + \beta_1 U$, содержащий единственный параметр β_0 .

При наличии в программе соответствующих предписаний могут быть выданы еще многие статистические характеристики набора

START	Подготовка системы к работе.
MO, TAD KC	Начало цикла по диодам:
DCA C	формирование счетчика цикла и
TAD A2	установка адреса начала массива
DCA 17	напряжений A2 .
INREG	Обращение к блоку регистрации:
TEXT	запрос переменной "TYPE", имеющей
TYPE =	номер 1, с пультовой машинки.
1	
END	
CAMAC	Обращение к блоку управления модулями
BEGMAS IMP	КАНАК: передача адреса начала массива
A2	напряжений A2 и подготовка к работе
END	
M1, INREG	Начало цикла измерений.
NIM POD REG	Обращение к блоку регистрации:
0	регистрация признака начала наблюдения
POD UM OWR REG	регистрация следующей переменной
	с последовательным умолчанием.
END	
TAD I 17	Выборка очередного значения напряжения.
DCA 30	
FLOAT	Обращение к арифметическому блоку:
REAL	перевод в плавающий формат.
MULT KU	Перевод (умножение на KU) из условных
END	единиц в вольты.
INREG	Обращение к блоку регистрации:
POD FORM REG	регистрация следующей переменной 2 (U)
END	без умолчания.
CAMAC	Обращение к блоку управления модулями
MOD1 OWL OWM	КАНАК: передача напряжения в модуль ЦАП
1	(№1) и сьем измерения тока с модуля АЦП
MOD2 EX RBL	(модуль №2)
2	
END	
FLOAT	Обращение к арифметическому блоку:
REAL	перевод тока в плавающий формат,
MULT KI	перевод в амперы.
END	
INREG	Обращение к блоку регистрации:
POD FORM REG	регистрация следующей переменной 3
FIN	(ток I); регистрация признака конца
END	наблюдения.
ISZ C	Конец цикла измерений тока.
JMP M1	
JMP MO	Переход на новый диод.

Рис.3. Пример программы управления экспериментом

ВОЛЬТАМПЕРНАЯ ХАРАКТЕРИСТИКА
TYPE=D2a3

ANALYSIS OF VARIANCE TABLE, REGRESSION COEFFICIENTS, AND STATISTICS OF FIT FOR DEPENDENT VARIABLE 1

B VALUES	T FOR MS:B=B	PROB > T	STD ERR B	STD B VALUES
0.17913436 (β_1)	0.99860	0.3440	0.17036902	0.06946420
0.22024571 (β_2)	10.32210	0.0001	0.02511233	0.91095501

Вольтамперная характеристика
(измеренная)

OBSERVED
VALUE

(сглаженная)

PREDICTED
VALUE

RESIDUAL

Доверительные интервалы

LOWER 95% CL
FOR MEAN

UPPER 95% CL
FOR MEAN

0.0	0.0	0.0	0.0	0.0
0.37000000	0.60730007	-0.13730007	0.04905171	0.76400042
0.92000000	1.27125155	-0.35125155	0.66153267	1.89007044
2.19000000	0.59161446	-0.40161446	1.00261703	3.36001109
3.52000000	4.50006070	-0.84846070	3.40761093	5.23931004
5.07000000	0.00101493	-0.65101453	5.77710611	7.47643295
10.20000000	0.00104170	0.90834030	8.44002400	10.11457090
13.40000000	12.63790020	0.96201972	11.67003104	13.20512073
15.00000000	16.04000020	-0.14000020	15.25191010	16.84960140
21.40000000	20.10011172	1.29900020	19.05671013	21.14350531
23.20000000	24.6159107	-1.41591057	23.14411712	26.00771201

SUM OF RESIDUALS	0	-0.74600785
SUM OF SQUARED RESIDUALS	0	6.00750000
SUM OF SQUARED RESIDUALS = ERROR SS	0	0.00000000
FIRST ORDER AUTOCORRELATION OF RESIDUALS	0	-0.11672002
DURBIN-WATSON D	0	1.97910749

ANALYSIS OF VARIANCE TABLE, REGRESSION COEFFICIENTS, AND STATISTICS OF FIT FOR DEPENDENT VARIABLE 1

STATISTICS	T FOR MEAN	PROB > T	STD ERR B	STD F VALUE 3
1.12-1.11-1.10	8.44883	0.0016	0.23632452	8.83639698
1.12-1.11-1.10	8.19374	0.0016	0.23632452	8.83639698

Вольтажперная характеристика
(измеренная) (сглаженная)

OBSERVED VALUE	PREDICTED VALUE
1.32000000	2.0
1.20000000	2.43594824
1.11000000	1.4971874
1.01000000	3.18348131
0.91000000	0.43186337
0.81000000	11.99367345
0.71000000	16.18109531
0.61000000	22.99352943
0.51000000	26.43117532
0.41000000	32.49403448

RESIDUAL

0.0
-1.1194854
-1.4711874
-1.67348131
-2.0686337
-1.5112655
-2.0119349
-0.2004727
-0.001752
-0.000348

Доверительные интервалы
↓
LOWER 95% CL FOR MEAN
UPPER 95% CL FOR MEAN

2.0	2.0
-1.13443098	1.87859737
0.5771125	2.48720623
1.9202528	4.44613775
7.7385955	9.61985815
12.54123434	13.2651256
15.1184895	17.35257162
19.0847236	22.17853449
24.9181835	27.92184031
30.55077491	34.61729405

SUM OF RESIDUALS

=

-0.0221486

SUM OF SQUARED RESIDUALS

=

12.17967656

SUM OF SQUARED RESIDUALS - ERROR SS

=

0.00000000

FIRST ORDER AUTOCORRELATION OF RESIDUALS

=

0.18785846

UPPER 95% CL

=

1.61393727

12-29 FRIDAY, 26 DECEMBER 1977

Рис. 4 (продолжение)

данных, здесь мы специально привели самую простую программу.

Как видно из примера, использование возможностей САС в системе САФРА позволяет пользователю - непрограммисту самому задавать достаточно сложные и ёмкие процедуры статистической обработки, а наличие единого архива и сквозной технологии регистрации, хранения и обработки поможет упростить и ускорить нелегкий процесс экспериментального исследования.

Л и т е р а т у р а

1. Гладких Б.А., Матушевский В.В. Представление данных в системе САФРА. Наст. сборник.
2. Золотенков В.В., Матушевский В.В. Помехоустойчивое кодирование данных на перфоленте. Наст. сборник.
3. Золотенков В.В., Осипова Л.Б. Математическое обеспечение для управления экспериментом в системе САФРА. Наст. сборник.
4. Матушевский В.В. Применение системы статистического анализа САС для обработки экспериментальных данных в системе САФРА. Наст. сборник.
5. Буллер И.Я. Программы обслуживания архивов в системе САФРА. Наст. сборник.
6. Поталов Д.В. Система вывода сложных изображений на АЦПУ ЕС ЭВМ (на базе PL/I ОС). Наст. сборник.
7. Гладких Б.А., Костюк Д.Л. Система графического вывода СМОГ и её реализация на ЕС ЭВМ. Настоящий сборник.
8. Виноградов В.М. Дискретные информационные системы в научных исследованиях. (Программно-управляемые модульные структуры). М., Атомиздат, 1976.
9. Barr A.J., Goodnight J.H. SAS programmer's guide. Published by Institute of statistics North Carolina State University. Raleigh, N.Carolina, 27607, 1972.

ПРЕДСТАВЛЕНИЕ ДАННЫХ В СИСТЕМЕ САФРА

Б.А.Гладких, В.В.Матушевский

1. Введение

Отличительными чертами автоматизированной системы регистрации, хранения и обработки экспериментальных данных САФРА [1] являются универсальность и комплексность. Универсальность означает способность системы охватить достаточно широкий круг экспериментальных исследований, а комплексность предполагает наличие сквозной технологии автоматизации эксперимента - от процесса регистрации сигналов до выдачи на печать или графопостроитель обработанных результатов. Эти свойства системы во многом противоречивы и их увязка может быть достигнута лишь ценой определенной стандартизации, в том числе стандартизации структур данных.

Система САФРА предназначена для комплексной автоматизации не любых экспериментов, а лишь таких, чьи результаты могут быть уложены в обсуждаемую ниже двухкоординатную структуру. Эта структура во многом диктуется конечным звеном в системе САФРА - подсистемой обработки информации, ориентированной на пакет статистического анализа САС [2] .

Следует различать два представления структур данных - логическое и физическое. Логическое представление ориентировано на пользователей, занимающихся обработкой полученных данных. Это представление абстрагировано от таких технических деталей, как точность, формат, умолчание и т.п.; программист, работающий с данными на уровне логического представления, может обращаться к ним просто по именам.

Физическое представление связано с процессами передачи

данных и их регистрации на носителе. Здесь технические моменты могут иметь в ряде случаев решающее значение, причем для условий конкретного эксперимента весьма важно выбрать такую форму представления, которая бы в наибольшей степени ему соответствовала. В системе САФРА пользователь в процессе подготовки эксперимента имеет возможность настраивать в определенных пределах параметры физического представления, задавая словарь переменных. В дальнейшем на основании этого словаря система сама установит соответствие между логическим и физическим представлением данных.

2. Логическое представление данных

Логически целостная совокупность данных, относящихся к конкретному эксперименту, образует набор данных. Каждый набор данных должен иметь уникальное имя, под которым он значится в архиве системы. Имена наборов данных стандартизованы, они имеют вид *FILE N*, где *N* - целое десятичное число, состоящее не более чем из четырех цифр, например, *FILE 1* *FILE 2746*.

Набор данных в системе САФРА имеет двумерную матричную структуру. Строкам матрицы соответствуют наблюдения, а столбцам - переменные; число переменных в одном наборе данных не должно превышать 254, число наблюдений не ограничивается. Каждая переменная имеет имя. Имена состоят из букв латинского алфавита и цифр, начинаются с буквы, их длина не должна превышать 8 символов.

Переменные могут быть двух типов - символьные и арифметические. Значением символьной переменной является строка литер, значением арифметической переменной - действительное число.

Если в некотором наблюдении значение переменной не измеряется, то предполагается, что эта переменная принимает здесь "пропущенное" значение *MISS*. Пропущенные значения могут особым образом учитываться при обработке.

Пример. В эксперименте по проверке закона Ома измеряются напряжение, ток и сопротивление. Кроме этого, должны быть зарегистрированы дата, фамилия экспериментатора, температура.

Пример соответствующего этому эксперименту набора данных (назовем его именем *FILE 2*) приведен в табл. I. Переменные *DATE*, *NAME* - символьные, остальные - арифметические.

Таблица I						
Наблюдение	<i>DATE</i>	<i>NAME</i>	<i>T</i>	<i>U</i>	<i>I</i>	<i>R</i>
1.	15.04.77	ИВАНОВ	286	50.2	0.250	200
2.	15.04.77	ИВАНОВ	MISS	16.4	0.033	200
3.	15.04.77	ПЕТРОВ	290	1.75	0.009	200
4.	16.04.77	ИВАНОВ	MISS	46.7	0.234	200
5.	16.04.77	ИВАНОВ	291	44.9	0.224	200

3. Принцип умолчания

Логическое представление данных наиболее удобно для обработки и интерпретации, но для хранения и передачи оно часто непригодно из-за большой избыточности. Для уменьшения объема регистрируемой информации в системе САФРА используется принцип умолчания. Умолчание может осуществляться в двух вариантах - последовательном и начальном.

Последовательное умолчание: если значение некоторой переменной в текущем наблюдении равно значению этой же переменной в предыдущем наблюдении, то это значение не регистрируется (умалчивается).

Умолчание по начальному значению: если значение переменной в текущем наблюдении равно заданному начальному значению, то это значение не регистрируется. Между умолчанием и пропущенным значением есть существенная разница: значение по умолчанию не есть пропущенное, в логическом представлении набора данных оно будет восстановлено по предыдущему либо начальному значению, в то время как пропущенное значение не восстанавливается.

Использование принципов умолчания дает возможность в ряде случаев значительно сократить объем данных. Например, в наборе данных *FILE 2* для переменных *NAME* и *R* целесообразно применить начальное умолчание для переменной *DATE* - последовательное, а для переменных *U*, *R* умолчание нецелесообразно.

4. Словарь переменных

Для каждого набора данных в системе САФРА необходимо завести словарь переменных. Этот словарь вводится как в управляющую, так и в обрабатывающую ЭВМ. В управляющей ЭВМ словарь используется для определения форматов при регистрации наблюдений, а также для скатия информации с помощью умолчания. В обрабатывающей ЭВМ словарь применяется для выполнения обратных действий по декодировке и развертыванию данных в форму логического представления.

Пример словаря переменных для набора данных *FILE2* приведен в таблице 2.

Таблица 2

номер перем.	имя перем.	формат переменных	принцип умолчания	ед. изм.	Примечание
1	<i>DATE</i>	<i>F</i> (8)	-		Дата в виде ДД.ММ.ГГ
2	<i>NAME</i>	<i>F</i> (10)	ИВАНОВ		Фамилия
3	<i>T</i>	<i>E</i> (2,4)	<i>MISS</i>	°К	Температура
4	<i>U</i>	<i>E</i> (2,4)		в	Напряжение
5	<i>I</i>	<i>E</i> (2,4)		а	Ток
6	<i>R</i>	<i>E</i> (2,4)	200.0	ом	Сопротивление

Переменные в словаре перечисляются в том порядке, в котором они должны быть в наборе данных. При этом каждая переменная кроме имени получает еще и порядковый номер, который в некоторых случаях удобно использовать вместо имени.

В графе "формат переменных" указывается тип переменной и форма внешнего представления. *F* означает символьную, *E* - арифметическую переменную. Цифры в скобках определяют формат внешнего представления, они будут объяснены ниже.

Принцип умолчания задается в следующей графе. Если для некоторой переменной в графе стоит пробел, то это означает регистрацию без умолчания в каждом наблюдении. Наличие прочерка "-" определяет для этой переменной последовательное умолчание. Любое другое число или текст (в зависимости от типа переменной) задает умолчание по начальному значению, которое берется в этом случае непосредственно из таблицы. Например, для переменной *NAME*

по умолчанию подразумевается литерное значение 'ИВАНОВ', а для переменной R - арифметическое значение 200. Особым образом воспринимается слово *MISS* : оно предписывает системе при умолчании подразумевать ПРОПУЩЕННОЕ значение. Для символьной переменной оно равно пустой строке, а для арифметической - отрицательному нулю.

Последние две графы словаря переменных системой не воспринимаются, они предназначены для экспериментатора.

5. Физическое представление для канала передачи данных

Канал передачи данных от управляющей к обрабатывающей ЭВМ в системе САЭРА может быть организован различными способами: с промежуточной регистрацией на перфоленту или магнитную ленту; может быть, при наличии соответствующего оборудования, налажена и прямая передача информации по линии связи. Однако при любом способе сохраняется принятый в ЕС ЭВМ принцип последовательного обмена восьмиразрядными байтами. Это дает возможность иметь единый стандарт "физического" представления данных для канала связи.

На уровне физического представления набор данных состоит из записей, а записи из полей переменных. Записи соответствуют наблюдениям логического представления, а поля - значениям переменных.

Использование принципа умолчания приводит к тому, что записи в физическом представлении набора данных имеют неопределенную длину, следовательно, необходимы разделители записей и полей внутри записи.

На рис. 1 представлен формат записи. Значком X на нем изображен байт, используемый для маркировки признаков. Он содержит единицы во всех разрядах (шестнадцатиричное число *FF*). Для идентификации записей в тех случаях, когда по каналу вперемежку поступают данные от нескольких экспериментов, в заголовке каждой записи указывается номер набора данных N (взятый из имени *FILE N*), к которому относится данная запись. После заголовка следуют поля тех переменных, которые в данной записи не умалчиваются. Для выделения переменных используются признаки, указывающие их порядковые номера по словарю переменных. Признак номера может быть опущен, если: а) первое поле соответствует

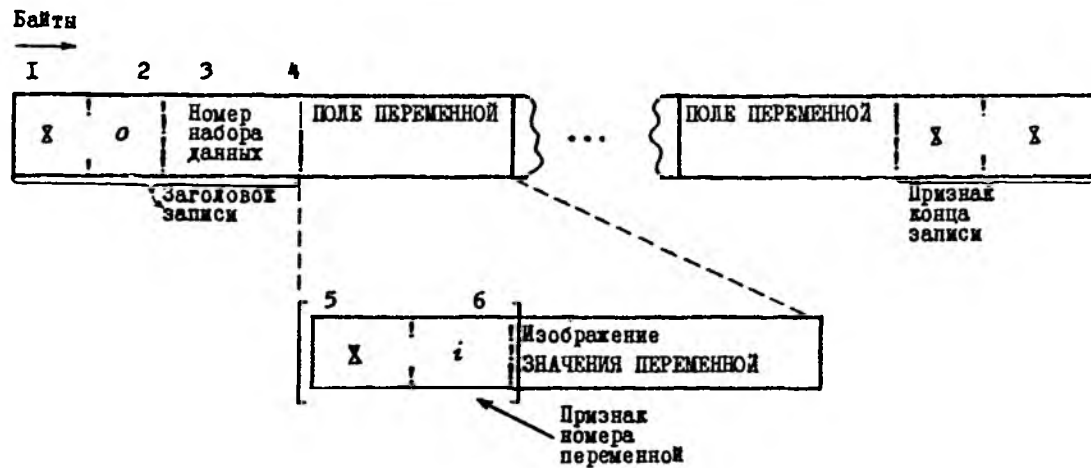
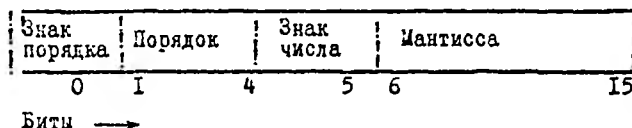


Рис. 1. Формат записи

переменной с номером I ; б) очередное поле соответствует переменной, номер которой на единицу больше, чем в предыдущем поле.

Изображение значения определяется типом переменной. Формат указывается цифрами в скобках в графе "формат" словаря переменных.

Изображение значения символьной переменной, описанной как $F(n)$ ($n \leq 64$), занимает n байтов, каждый байт соответствует одному символу в кодировке управляющей ЭВМ. Арифметические переменные, описанные в виде $E(n, m)$ ($n \leq 8, m \leq 6$), представляются в n байтах в плавающем формате, причем под порядок отводится m бит. Например, число в формате $E(2, 4)$ представляется следующим образом:



Для экономного представления арифметических значений имеется дополнительный целый формат $I(n)$, ($n \leq 4$), занимающий n байт и представляющий целое двоичное число без знака.

П р и м е р . Набор данных *FILE2*, приведенный в таблице 1, если его описать словарем переменных из таблицы 2, будет иметь следующее представление в канале передачи данных (прямоугольники изображают двухбайтовое плавающее представление соответствующих значений):

```

X002I5.04.77 X3 [286] [50.2] [0.250] XX
X002 I4 [16.4] [0.033] XX X002 X2 ПЕТРОВ ____
[290] [1.75] [0.009] XX X002I6.04.77 X4 [46.7]
[0.234] XX X 002 X3 [291] [44.9] [0.224] XX
  
```

Уже на этом простом примере заметно сжатие информации, достигаемое умолчанием: 84 байта вместо 160, которые понадобились бы, если бы умолчание не использовалось.

6. Представление на пятидорожечной перфоленте

Описанное в предыдущем разделе представление в канале передачи данных ориентировано на тот основной случай, когда данные регистрируются аппаратурой, сопряженной с управляющей ЭВМ и приводятся этой ЭВМ к требуемому формату. Однако в некоторых случаях возникает необходимость передать обрабатывающей ЭВМ данные, полученные вручную либо автономной регистрирующей аппаратурой.

С этой целью в системе САЭРА предусмотрена возможность работы с другими внешними представлениями. Сейчас мы рассмотрим более подробно представление на пятидорожечной перфоленте, ориентированное на ручную регистрацию данных с помощью обычного телеграфного аппарата. Словарь переменных для набора данных, вводимого с пятидорожечной ленты, составляется обычным порядком, с той лишь разницей, что для арифметических переменных нет необходимости указывать формат в скобках.

Данные представляются в коде МТК-2, формат записей следующий:



Как видно из рисунка, заголовок записи в данной представлении не обязателен, он нужен только в первой записи для идентификации набора данных. Разделитель записей — две наклонные черты. Каждое поле переменной имеет следующую структуру: символ X (клавиша "кто там?"), имя переменной либо ее номер, символ =, изображение значения переменной. Так же кодируется и заголовок записи. Поля переменных могут для наглядности разделяться пробелами, их порядок в записи несуществен, так как каждое поле содержит имя переменной. Более того, если при кодировке поля обнаружена ошибка, то его можно повторить в этой же записи.

Изображение арифметических значений допускается как с фиксированной, так и с плавающей точкой, например, 100, 10.27, -5.46,

-0.12E-13. Символьные значения изображаются обычным образом, конец изображения определяется либо следующим символом Σ , либо концом записи, при этом система сама приведет значение к длине, требуемой словарем переменных.

Пр и м е р . Набор данных *FILE 2* из таблицы I может быть вручную закодирован на перфоленте следующим образом:

```
XFILE=2  XDATE=15.04.77  T=206  ZU=50.2  XI=0.250//  
X4=16.4  X5=0.033//  X2=ПЕТРОВ  X3=290  X4=1.75  X5=.009//  
X1=16.04.77  X4=46.7  X5=.234//  X3=201  X4=44.9  X5=.224
```

7. Специальные внешние представления

Для обеспечения совместимости со специальной аппаратурой регистрации данных, работающей в собственном формате, в системе САФРА имеется программный модуль сопряжения [3]. Этот модуль реализован как синтаксически-управляемый анализатор. Подготовив для него таблицу, описывающую формальный синтаксис входного языка, можно настроить модуль на различные внешние представления. Таким образом, данные, зарегистрированные на перфоленте автономной аппаратурой, могут быть занесены в общий архив системы и стать доступными подсистеме обработки информации.

Л и т е р а т у р а

1. Гладких Б.А. Автоматизированная система регистрации, хранения и обработки экспериментальных данных САФРА – общее описание. Наст. сборник.
2. Матусевский В.В. Применение системы статистического анализа САС для обработки экспериментальных данных в системе САФРА. Наст. сборник.
3. Буллер И.Я. Программы обслуживания архивов в системе САФРА. Наст. сборник.

МАТЕМАТИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДЛЯ УПРАВЛЕНИЯ ЭКСПЕРИМЕНТОМ В СИСТЕМЕ САФРА

В.В.Золотенков, Л.Б.Осипова

Описываемое в настоящей работе математическое обеспечение представляет собой часть общего математического обеспечения системы САФРА [1] и предназначено для реализации функций подсистемы управления экспериментом и частично (в той части, которая ложится на управляющую ЭВМ) подсистемы регистрации и хранения информации. Эти функции следующие:

- управление модулями КАМАК: прием и передача данных в устройства сопряжения с экспериментом;
- первичная арифметическая и логическая обработка данных;
- преобразование данных из внутреннего представления управляющей ЭВМ во внешнее представление канала передачи данных;
- помехоустойчивое кодирование и регистрация данных на перфоленде;
- обеспечение диалога с экспериментатором.

В соответствии с перечисленными функциями математическое обеспечение состоит из четырех основных частей: арифметического блока, блока управления модулями КАМАК, блока регистрации, диалоговой системы ввода-вывода.

Роль управляющей ЭВМ в системе САФРА может выполнять любая малая ЭВМ, система команд которой аналогична PDP-8 (Саратов, Электроника-100, ТРА-1000). Так как базовый комплект поставки малых ЭВМ, как правило, включает МОЗУ минимального объема (4 К слов) и простейшую конфигурацию внешних устройств, то при разработке математического обеспечения была поставлена задача обеспечить полную работоспособность системы при минимальном комплекте технических средств. Это требование, а также стремление обеспе-

Читы достаточное быстродействие определили ориентацию математического обеспечения на путь организации макрокоманд, реализующих отдельные функции. Макрокоманды записываются в определенной форме машинного представления, их распознавание и выполнение ложится на специальные интерпретаторы макрокоманд.

Особое внимание при разработке было обращено на удобство чтения и понимания программ. Поэтому создаваемое специализированное программное обеспечение предусматривает использование осуществляющего стандартного транслятора с языка АССЕМБЛЕР [2]. С этой целью в состав постоянной таблицы символов транслятора включены все mnemonic обозначения вновь вводимых команд интерпретаторов. Кроме того, дополнительно введены команды обращения к интерпретаторам и выхода из них. Эти дополнительные команды сведены в таблицу I.

Таблица I

КОД команд	Мнемоническое обозначение	Выполняемое действие
4405	<i>INREG</i>	Обращение к интерпретатору блока регистрации.
4406	<i>SAMAC</i>	Обращение к интерпретатору блока управления модулями КАШАК.
4407	<i>FLOAT</i>	Обращение к интерпретатору арифметического блока.
0000	<i>END</i>	Выход из интерпретаторов.
4420	<i>START</i>	Начальная подготовка системы к работе. Эта макрокоманда ставится в самом начале программы управления экспериментом.

Кроме того, для организации диалога ЭВМ с оператором в состав транслятора введена макрокоманда *TEXT*. По этой команде транслятор воспринимает последующую символьную информацию до знака "=" как сообщение для оператора (см. описание команды *QU* в блоке регистрации).

Реализованный вариант математического обеспечения для управления экспериментов, включающий все четыре основных блока, занимает около 2000 ячеек памяти. Для сравнения отметим, что стандартный арифметический блок, выполняющий операции с плавающей запятой, один занимает 1424 ячейки [2].

2. Арифметический блок

При разработке арифметического блока были поставлены две основные задачи. Во-первых, это создание набора макрокоманд, выполняющих арифметические операции сложения-вычитания, умножения и деления над числами, представленными в плавающем формате; Во-вторых, - организация различных логических функций над этими числами и вычисление стандартных функций.

При создании набора макрокоманд арифметических операций за основу был взят блок с плавающей запятой (БПЗ), имеющийся в стандартном математическом обеспечении, однако с целью уменьшения объема занимаемой памяти была проведена его модернизация.

Стандартный БПЗ имеет 8 макрокоманд, представленных в таблице 2.

Таблица 2

Код команды	Мнемоническое обозначение	Ф у н к ц и я
1000	<i>FADD</i>	Алгебраическое сложение
2000	<i>FSUB</i>	Алгебраическое вычитание
3000	<i>FMUL</i>	Умножение
4000	<i>FDIV</i>	Деление
5000	<i>FBET</i>	Вызов операнда на сумматор
6000	<i>FPUT</i>	Запись операнда в память
7000	<i>FNOR</i>	Нормализация
0000		Выход из БПЗ

Вычисления выполняются с точностью до местного десятичного знака. Поэтому число, используемое блоком, занимает 3 ячейки памяти: в первой хранится порядок числа, в двух последних - мантисса. Для выполнения операций в памяти машины выделено место под псевдосумматор (ПСМ) и псевдорегистр (по 4 ячейки каждый). Для повышения точности вычислений в состав ПСМ и псевдорегистра включено по дополнительной ячейке, запоминающей младшие разряды мантиссы. Отрицательные числа (как порядки, так и мантиссы) представлены в дополнительном коде.

Как показал анализ программы стандартного БПЗ, его объем можно значительно сократить, если применить процедуры общего пользования для ПСМ и псевдорегистра и модифицировать алгоритмы операций сложения-вычитания и умножения. Произведенная переработка данной части БПЗ позволила на 40% сократить объем занимаемой

блоком памяти и на 10-15% повысить быстродействие. Среднее быстродействие блока на ЭВМ Саратов составляет 150 оп/сек.

Состав макрокоманд блока также был изменен (см. таблицу 3).

Таблица 3

КОД команды	Мнемоническое обозначение	Ф у н к ц и я
2000	<i>GET</i>	Заслать операнд на ПСМ
3000	<i>PUT</i>	Заслать операнд в память
4000	<i>MULT</i>	Умножение
5000	<i>DIV</i>	Деление
6000	<i>ADD</i>	Алгебраическое сложение
7000	<i>SUB</i>	Алгебраическое вычитание

Во-первых, из него исключена команда нормализации, т.к. результаты выполнения всех операций нормализуются в самом алгоритме операции. Для удобства пользования программой нормализации она оформлена как процедура.

Вместо команды нормализации в состав интерпретируемых команд введена макрокоманда ДЕЙСТВИЕ. Для этой макрокоманды создан специальный интерпретатор.

Как и команда ДЕЙСТВИЕ ЭВМ [4], эта макрокоманда работает с операндом, находящимся на ПСМ. Результат также остается на ПСМ. Интерпретатор данной операции рассматривает слово команды как набор директив на выполнение процедуры. Каждому биту слова команды соответствует своя процедура. В принципе, процедуры могут быть любой сложности, а также быть легко заменяемы (для этого необходимо скорректировать таблицу адресов интерпретатора). Всего определены 24 интерпретируемые команды. Они разбиты на 4 группы, приводимые в таблице 4.

Таблица 4

Группа I

КОД команды	Мнемоническое обозначение	Ф у н к ц и я
0004	<i>RAD</i>	Перевод из градусной меры угла в радианную
0010	<i>DOR</i>	Нахождение угла, дополнительного до $\pi/2$
0020	<i>SIN</i>	Синус x
0040	<i>TG</i>	Тангенс x
0100	<i>ATN</i>	Арктангенс x

0200	<i>DEG</i>	Перевод угла из радианной меры в градусную
0400	<i>DIVX</i>	Нахождение величин $1/x$

Группа 2

0006	<i>EXP</i>	Функция e^x
0012	<i>LOG</i>	Функция $\log x$
0202	<i>SQT</i>	Функция $\sqrt{x}$
0402	<i>DIVX</i>	Функция $1/x$

Группа 3

0005	<i>SIGN</i>	Функция сигнум x
0011	<i>ABS</i>	Нахождение абсолютного значения числа
0021	<i>ITR</i>	Нахождение целого, не превосходящего значение числа
0061	<i>EXT</i>	Функция x^y , где y - целое число
0101	<i>REAL</i>	Перевод целого числа в вещественное
0201	<i>INV</i>	Функция $y = -x$

Группа 4

0007	<i>IF (-)</i>	Используются совместно с командой <i>GO</i> для условных переходов
0013	<i>IF (0)</i>	
0023	<i>IF (+)</i>	
0043	<i>IF (>1)</i>	
0103	<i>R</i>	Возврат из процедуры
0203	<i>DO</i>	Вход в процедуру
0403	<i>GO</i>	Безусловный переход

Команды одной группы могут комбинироваться. Например, совместная команда *DOP* и *SIN* используется для нахождения функции $\cos x$, *TG* и *DIVX* - для нахождения функции $\operatorname{ctg} x$. Команды *IF* группы 4 используются только совместно с командой *GO* для осуществления условных переходов.

Все команды, за исключением *DO* и *GO*, не требуют дополнительных сведений для исполнения. Команды *DO* и *GO* требуют указать адрес ячейки программы, к которой осуществляется переход. Это достигается тем, что в следующей после команды ячейке программы указывается полный адрес ячейки, в которую передается управление.

П р и м е р . Допустим, что нужно вычислить выражение $y = \operatorname{cosec} (x/z - \kappa \cdot A)$. Далее, если $y < 0$, то выполнить $Z = y$, иначе вычислить $Z = \sqrt{y}$. Программа запишется следующим образом:

```

GET K
MULT A
PUT R           R - рабочая ячейка
GET X
DIV Z
SUB R
DOP SIN DIVX     Вычисление функции
IF(-) GO
B                Адрес перехода
SQT
B, PUT Z

```

3. Блок управления модулями КАМАК

При создании данного блока ставилась задача создания набора программ, обеспечивающих взаимодействие вычислительного процесса с периферийным оборудованием, выполненным в стандарте КАМАК.

Как известно [3], один КАМАК - модуль имеет 32 команды и 16 субадресов для каждой команды. Кроме того, каждая секция имеет 24 адресуемых модуля, каждая ветвь - 7 секций. К одной ЯВМ может быть подключено до 7 ветвей. Естественно, что непосредственно охватить такое множество команд и адресов невозможно, тем не менее создаваемый комплект программ должен иметь достаточно возможностей для реализации всех необходимых функций.

Анализ команд модулей КАМАК показал, что по частоте использования команды резко отличаются друг от друга. Поэтому было принято решение реализовать наиболее часто встречающиеся команды, оставив принципиальную возможность для выполнения любой команды КАМАК.

Блок управления модулями КАМАК организован по принципу интерпретатора команд ДЕЙСТВИЕ БПЗ. Он также имеет набор микроопераций и интерпретирующую программу, которая просматривает слово

команды и при наличии "I" в очередном бите слова вызывает соответствующую процедуру. Интерпретатор имеет 2I команду, они разделены на 3 группы. В первую группу сведены команды передачи данных из ЭВМ в КАМАК. Во второй группе – команды управления сервисным регистром. В третью группу собраны команды управления модулями и чтения информации из модулей в ЭВМ.

В каждую группу входят команды подготовки интерпретатора, позволяющие сформировать полный адрес модуля *CNA*. Для ввода адреса *CNA* в интерпретатор, он размещается в следующей после команды ячейке программы.

Для передачи информации в модули КАМАК, она должна быть сведена в массив. Адрес начала массива передается в интерпретатор аналогично адресу *CNA*.

Для обеспечения возможности выполнения любой команды КАМАК введена специальная команда *NECT*. В следующих за командой ячейках программы размещаются: в первой ячейке – команда *F*, во второй – адрес *CNA*.

Список команд интерпретатора КАМАК приведен в таблице 5.

Таблица 5

Группа I

Код команды	Имемоническое обозначение	Ф у н к ц и я
0004	<i>IMP</i>	Инициирование магистрали
0010	<i>MOD 1</i>	Установка номера модуля
0020	<i>BE6MAS</i>	Установка адреса начала массива данных
0040	<i>OWL</i>	Запись нижнего полуслова в драйвер системы
0100	<i>OWN</i>	Запись верхнего полуоова в драйвер системы
0200	<i>OWD</i>	Запись полного слова в драйвер системы
2000	<i>WWM</i>	Запись слова в модуль,чей адрес установлен командой
4000	<i>NECT</i>	Неопределенная интерпретатором операция

Группа 2

0011	<i>MOD 2</i>	Установка номера модуля
0021	<i>EX</i>	Команда ИСПОЛНИТЬ
0205	<i>WIR</i>	Ожидание конца исполнения команды

просто описывать последовательность измеряемых переменных, производить их обработку и регистрировать в принятых форматах и с заданным характером умолчания (последовательное или начальное).

Блок регистрации построен по проверенной схеме интерпретатора команд ДЕЙСТВИЕ. Достоинства такой схемы были описаны выше. Интерпретатор имеет 9 команд (см.таблицу 6).

Таблица 6

Код команды	Мнемоническое обозначение	Ф у н к ц и я
0002	<i>QU</i>	Запрос данных
0004	<i>NUM</i>	Устанавливается номер выводимой переменной
0010	<i>DO</i>	Обращение к стандартной подпрограмме
0040	<i>POD</i>	Подготовка программы к выводу регистрируемой переменной
0100	<i>FORM</i>	Преобразование в выходной формат
0200	<i>UM</i>	Сообщается, что должен действовать принцип умолчания
0400	<i>OWR</i>	Перепись в массив умолчания
2000	<i>REG</i>	Регистрация переменной
4000	<i>FIN</i>	Регистрация признака конца записи

Рассмотрим структуру команд подробнее.

QU - запрос с пультовой машинки данных или символьной информации.

Если в следующей за командой ячейке программы хранится число 7777, то программа рассматривает содержимое следующих ячеек программы как сообщение оператору и выводит его на печать (до символа =), если же это число не равно 7777, то запрашивается переменная типа *F*. Команда *NUM* устанавливает номер выводимой переменной, если вывод идет не по порядку (в частности, начальный номер). Номер занимает следующую ячейку программы. Команда *DO* аналогична команде *DO* БПЗ. Она введена для облегчения написания программ.

Следующая команда *POD* готовит программу к выводу значения регистрируемой переменной. По номеру переменной процедура находит в списке параметры переменной (формат, адрес, принцип умолчания) и сообщает их интерпретатору для последующего использования.

Команда *FORM* приводит результат вычислений, представлен-

040I	<i>RBL</i>	Чтение нижнего полуслова
100I	<i>RBH</i>	Чтение верхнего полуслова

Группа 3

0006	<i>MOD 3</i>	Установка номера модуля
0012	<i>SVRO</i>	Установка субадреса сервисного регистра
0022	<i>CSV</i>	Сброс всего сервисного регистра
0042	<i>CLS</i>	Гашение маски запроса
0102	<i>SMT</i>	Установка статусного регистра
0202	<i>SLM</i>	Установка регистра маски
0402	<i>RSVN</i>	Чтение верхнего полуслова сервисного регистра
1002	<i>RSVL</i>	Чтение нижнего полуслова сервисного регистра

П р и м е р . Пусть нам необходимо задать воздействие на внешнее устройство путем выдачи константы на выходной регистр, находящийся на $I7_8$ позиции секции, а затем измерить внешний сигнал модулем АЦП, находящимся на $2I_8$ позиции секции.

Соответствующая программа выглядит так:

<i>MOD 1 BEGMAS OWD OWM</i>	Передача числа на выходной регистр
<i>I7</i>	Адрес модуля
<i>A</i>	Адрес константы
<i>MOD 2 EX RBL</i>	Измерение и регистрация
<i>2I</i>	Адрес модуля
<i>A</i> , константа, выдаваемая на выходной регистр.	

Как показал опыт работы, приведенные команды охватывают большую часть используемых команд стандарта КАМАК. При этом в случае необходимости интерпретатор позволяет легко изменить набор определяемых команд. Быстродействие системы зависит от типа используемых комбинаций команд и составляет 1500-2000 команд в секунду. Такого быстродействия вполне достаточно для проведения широкого ряда физических экспериментов.

4. Блок регистрации

Несмотря на принятые принципы хранения информации на перфокартах в системе САФРА блок регистрации должен позволять легко и

ный в машинном формате к формату вывода. При обращении к процедуре чиоло должно быть на ПСМ.

Процедура *UM* сравнивает полученный результат с результатом, хранящимся в памяти ЭВМ. При совпадении поднимается флаг умолчания.

Команда *OWR* используется для переписи текущего значения переменной и массива умолчания при последовательном умолчании. При начальном умолчании команда *OWR* не используется. Если переменная выводится без умолчания, не используется команда *UM*.

Команда *REG* выводит результат на перфорацию, если флаг умолчания опущен, или не выводит и поднимает флаг пропуска, если флаг умолчания поднят. Если флаг пропуска поднят и переменная регистрируется, предварительно выводится на перфорацию ее номер. Затем номер переменной увеличивается на единицу. Последняя команда *FIN* регистрирует на перфоленте признак конца записи.

Так же как команды ДЕЙСТВИЕ БПЗ, команды блока регистрации могут комбинироваться. Таким образом, комбинированная команда в одной ячейке указывает всю процедуру обработки данной переменной.

П р и м е р . Рассмотрим следующую операцию. Пусть нам последовательно нужно зарегистрировать переменные 0, I, 2, IO, II, 5, 6 и закончить запись. Переменная 0 не умалчивается вообще; I, 2, 5 - последовательное умолчание; IO, II, 6 - начальное. Переменная 0 под названием ФАЙЛ запрашивается машиной.

Эта программа запишется следующим образом:

<i>TEXT</i>	По команде <i>TEXT</i> АССЕМБЛЕР оформит и
<i>ФАЙЛ</i> =	запишет команды печати на пишущей машинке
	слова <i>ФАЙЛ</i> ; затем оператор с клавиатуры
	вводит его значение
<i>NUM POD REG</i>	регистрация переменной <i>ФАЙЛ</i>
<i>0</i>	номер переменной
<i>POD FORM UM OWR REG</i>	Регистрация первой переменной
<i>POD FORM UM OWR REG</i>	
<i>NUM POD FORM UM REG</i>	
<i>IO</i>	
<i>POD FORM UM REG</i>	
<i>NUM POD FORM UM OWR REG</i>	
<i>5</i>	

5. Диалоговая система ввода-вывода

Диалоговая система ввода-вывода информации работает со штатными периферийными устройствами ЭВМ в режиме прерываний. Такой режим выбран по многим причинам, в частности потому, что оператор должен иметь возможность вмешиваться в работу программы. Для обеспечения одновременной работы перфоратора и пишущей машинки в некоторых типах управляющих ЭВМ, например, Саратол должны быть произведены незначительные доработки стандартного устройства ввода-вывода. В результате удастся программно разделить вывод информации на перфоратор и на пишущую машинку.

Ввод информации работает следующим образом. При нажатии клавиши на пишущей машинке устройство ввода-вывода формирует сигнал ЗАПРОС в процессор. По этому сигналу прерывается работа программы и вызывается процедура обработки входной информации. Эта процедура вводит символы, выдает их на печать и формирует из них признак вводимой директивы. По окончании ввода директивы (символы ПРОБЕЛ или ВОЗВРАТ КАРЕТКИ) управление передается процедуре распознавания директивы. Она по словарию опознает введенную директиву и настраивает процедуру ввода на дальнейшую работу (это может быть ввод параметров директивы или исполнение ее). Если введенная директива не опознается, на печать выдается сообщение об ошибке. Ввод директив и значений осуществляется посимвольно, т.е. после обработки введенного символа состояние процедуры ввода запоминается и управление передается программе с момента прерывания.

Вывод всей информации (на печать и на перфоратор) осуществляется через буфер вывода емкостью 64 байта. Символ, предназначенный на вывод, записывается в буфер, при этом поднимается флаг вывода. Если устройство вывода свободно, символ выводится по сигналу прерывания от устройства вывода после его освобождения. Вся информация, выводимая на перфоленгу, кодируется помехоустойчивым кодом [5]. Размер буфера вывода позволяет выгружать все про-

верочную часть, формируемую кодером, чем обеспечивается безостановочная работа программы. В случае заполнения буфера полностью работа программы приостанавливается до момента его освобождения.

6. Заключение

Созданное математическое обеспечение для управления экспериментом существенно облегчает работу программиста, связанную с описанием процессов обработки информации и управления физическим экспериментом. Как показал опыт использования системы, программист может построить программу несколькими путями. Во-первых, это последовательная программа, где все процессом описаны с помощью соответствующих интерпретаторов в заданной последовательности. Во-вторых, возможно построение программы, отдельные функции которой оформлены в виде процедур. Интерпретатор управления экспериментом командой *DO* обращается к соответствующей процедуре и затем обрабатывает полученный результат.

Л и т е р а т у р а

1. Гладких Б.А. Автоматизированная система регистрации, хранения и обработки экспериментальных данных САФРА - общее описание. Наст.сборник.
2. Математическое обеспечение ЭВМ "Саратов" инструкции О.І4І.015, О.І4І.019, О.І4І.020.
3. Виноградов В.И. Дискретные информационные системы в научных исследованиях. М., Атомиздат, 1976.
4. Флорсо А. Организация вычислительных машин. М., "Мир", 1972.
5. Золотенков В.В., Матусевский В.В. Помехоустойчивое кодирование данных на перфокарте. Наст.сборник.

ПОМЕХОУСТОЙЧИВОЕ КОДИРОВАНИЕ ДАННЫХ НА ПЕРФОЛЕНТЕ

В.В.Золотенков, В.В.Матушевский

В системе САФРА [3] функции регистрации и хранения данных разделены между двумя ЭВМ. Для передачи информации из ЭВМ, управляющей экспериментом, в обрабатывающую ЭВМ организован канал связи. Под общим понятием "канал связи" имеется в виду комплекс аппаратных средств ввода/вывода информации и ее физических носителей. Таким образом, каналом связи может быть перфолента с перфоратором и устройством ввода, магнитная лента с магнитофоном, радиоканал с передающей и приемной аппаратурой и т.д. В настоящей реализации системы САФРА носителем информации является перфолента.

Можно выделить три группы каналов связи для передачи дискретной информации. В последовательном канале информация передается последовательно по битам. В параллельном канале каждый бит сообщения передается по отдельному независимому подканалу. Наконец, в смешанном (параллельно-последовательном) канале сообщение передается последовательными порциями (символами) фиксированной длины по нескольким подканалам. Такой канал связи характерен для обмена информацией ЭВМ с внешними устройствами.

Во всем канале связи присущи два типа ошибок, неизбежно возникающих из-за воздействия помех. Ошибки первого типа, или аддитивные ошибки, — это ошибки, возникающие из-за инверсии значения бита или группы битов (переходы $1 \rightarrow 0$ и $0 \rightarrow 1$). Ошибки второго типа это структурные ошибки, которые приводят к исчезновению или появлению бита или группы битов. Обычно эти ошибки связаны с нарушением синхронизации между передающей и приемной аппаратурой и часто называются ошибками синхронизации. Эти ошибки наиболее опасны, так

как могут значительно исказить сообщение, а в параллельном канале и полностью уничтожить его.

Так как экспериментальные данные представляют известную ценность, а некоторые вообще могут быть уникальными, необходимо защищать информацию в канале связи от искажений. Точнее говоря, нужно организовать такое кодирование информации, которое позволяло бы корректировать ошибки — помехоустойчивое кодирование.

Способ кодирования зависит от режима работы канала связи. Так, в режиме работы с переспросом достаточно использовать коды, обнаруживающие ошибки. Обнаруженные ошибки исправляются при повторном приеме. В том случае, когда невозможно организовать канал с переспросом, или такой режим оказывается невыгодным, необходимо применение корректирующих кодов [1]. Корректирующие коды способны исправлять одиночные или групповые аддитивные ошибки. Одним из способов борьбы со структурными ошибками является включение в передаваемое сообщение специальных синхронизирующих последовательностей. Для последовательных каналов наиболее удобная синхронизирующая последовательность — это псевдослучайная последовательность Баркера [2] с ярко выраженным пиком автокорреляционной функции. Все сообщение разбивается на зоны фиксированной длины, между зонами кодирующая программа помещает синхронизирующие последовательности. Декодирующая программа отыскивает синхронизирующие последовательности и определяет длину информационной зоны. Ошибка синхронизации обнаруживается, если длина зоны не совпадает с заданной. Структура информационной зоны восстанавливается и ошибка синхронизации превращается в аддитивные ошибки, методы коррекции которых достаточно хорошо разработаны [1].

Для смешанных каналов, которые нас особенно интересуют, характерно возникновение специфических пакетов аддитивных ошибок. Во-первых, это ошибки, связанные с отказом одного из подканалов. Примером источника такой ошибки может быть оломанный цуансон в ленточном перфораторе. Во-вторых, это ошибки связанные со случайным сбоем нескольких или всех подканалов одновременно. Такой пакет ошибок возникает, например, при преобразовании ошибок синхронизации в аддитивные и приводит к искажению одного или нескольких последовательных символов.

Для коррекции пакетов ошибок требуются коды с большой избыточностью [1]. Однако специфика смешанного канала позволяет раз-

нести (декоррелировать) пакеты ошибок соответствующей организацией кодирования. Превратив, таким образом, пакет ошибок в одиночные некоррелированные ошибки, можно использовать коды с меньшей избыточностью — коды, исправляющие лишь одиночные ошибки.

В смешанном канале возможны, по крайней мере, два способа помехоустойчивого кодирования. Первый способ заключается в том, что каждый символ защищается помехоустойчивым кодом, то есть каждый символ представляет собой кодовую последовательность. Существенный недостаток такого способа кодирования состоит в том, что нарушение синхронизации (исчезновение одного символа) вызовет исчезновение всей кодовой последовательности. Исправление пакета ошибок, вызванного нарушением синхронизации, становится невозможным. Другой способ заключается в том, что кодирование организуется отдельно в каждом из подканалов. Тогда пакет ошибок, вызванный нарушением синхронизации (исчезновение одного символа), приведет лишь к одиночным ошибкам в каждой из кодовых последовательностей. Очевидно, что такой способ организации кодирования более целесообразен. Итак, в дальнейшем предполагается, что кодирование ведется отдельно в каждом из подканалов.

Декорреляцию пакета ошибок, связанных с отказом одного из подканалов, можно осуществить посредством циклического сдвига последовательных символов на один или несколько битов друг относительно друга. Циклический сдвиг символов приведет к тому, что в группе последовательно переданных символов ошибочными окажутся разные биты. Нужно учесть при этом, что длина кодовой последовательности при циклическом сдвиге по одному биту не должна превышать длины символа в канале, иначе ошибка повторится внутри кодовой последовательности для разных символов.

Пакет ошибок, связанных с исчезновением нескольких последовательных символов, можно декоррелировать, если идущие последовательно символы отнести к разным кодовым последовательностям. Так как кодирование ведется отдельно по подканалам, такой пакет ошибок распадется на одиночные ошибки в разных кодовых последовательностях разных подканалов.

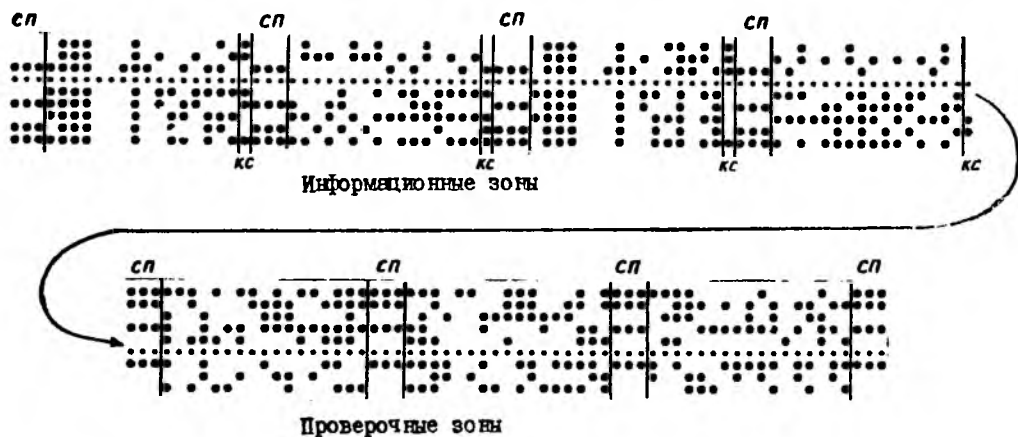
Предоставление кодируемой информации на перфокарте. В системе САФРА организован смешанный канал связи с произвольной передачей сообщений. Физическим носителем информации в настоящее время является восьмидорожечная перфокарта. Длина символа составляет 8 бит

или 1 байт. В системе реализовано помехоустойчивое кодирование с применением упомянутых методов защиты от структурных ошибок и декорреляции ошибок. Для обнаружения и исправления односторонних аддитивных ошибок используется циклический код Хэмминга (7,4). Длина кода определяется ограничением, накладываемым максимальной величиной циклического сдвига символов (длина кодовой последовательности $<$ длины слова в битах).

Информация на перфоленте располагается блоками, каждый из которых защищен помехоустойчивым кодированием. Каждый кодируемый блок разбивается на семь зон, из которых первые четыре — информационные, а следующие три — проверочные. Каждая зона имеет длину в 17 байтов. В информационных зонах каждый 17-й байт представляет собой циклическую контрольную сумму по модулю 256 предыдущих 16 байтов. Контрольная сумма используется для обнаружения пакетов ошибок, которые не удалось декоррелировать. В начале и конце блока, а также между зонами помещаются синхронизирующие последовательности, состоящие из трех одинаковых символов (c_{16}). Между блоками может быть помещена любая информация — так называемый "мусор", который при декодировании выбрасывается. Пример кодируемого блока изображен на рис. 1.

Кодирование. Кодовая последовательность организуется из i -х байтов всех зон, то есть для i -х символов информационных зон формируется i -е символы проверочных зон. Такая организация кодовых последовательностей позволяет декоррелировать пакеты ошибок, связанные с искажением нескольких последовательных символов, так как они разносятся по различным кодовым последовательностям.

Для декорреляции систематических ошибок одного из подканалов производится циклический сдвиг символов кодовой последовательности. При этом i -й информационный символ кодовой последовательности сдвигается циклически на i битов вправо. Следует заметить, что сами информационные зоны выдаются на перфорацию без сдвига. Фиктивный сдвиг осуществляется лишь на входе в декодер. Циклический сдвиг информационных символов, таким образом, влияет только на формирование проверочных зон и никак не отражается на их представлении на перфоленте. Сформированные проверочные символы также циклически сдвигаются: символ i -й проверочной зоны сдвигается на i битов вправо. Проверочные зоны выдаются на перфорацию движущимися. Схематическое формирование кодовой последовательности мож-



СП - синхронизирующая последовательность

КС - контрольная сумма

Рис. I Формат данных на перфоленге

но представить так:

Информационная часть
кодовой последовательности

j -я байты зон								
1	1	2	3	4	5	6	7	8
2	1	2	3	4	5	6	7	8
3	1	2	3	4	5	6	7	8
4	1	2	3	4	5	6	7	8

биты

Циклический сдвиг
информационных символов

j -я байты зон								
1	8	1	2	3	4	5	6	7
2	7	8	1	2	3	4	5	6
3	6	7	8	1	2	3	4	5
4	5	6	7	8	1	2	3	4

биты

Формирование проверочных
символов

j -я байты зон								
1	8	1	2	3	4	5	6	7
2	7	8	1	2	3	4	5	6
3	6	7	8	1	2	3	4	5
4	5	6	7	8	1	2	3	4
5	a	b	c	d	e	f	g	h
6	a	b	c	d	e	f	g	h
7	a	b	c	d	e	f	g	h

биты

Представление информации
на перфокарте

j -я байты зон								
1	1	2	3	4	5	6	7	8
2	1	2	3	4	5	6	7	8
3	1	2	3	4	5	6	7	8
4	1	2	3	4	5	6	7	8
5	h	a	b	c	d	e	f	g
6	g	h	a	b	c	d	e	f
7	f	g	h	a	b	c	d	e

биты

Цифрами показано начальное расположение соответствующих битов информационных символов и их расположение после сдвигов. Латинские буквы обозначают то же самое для проверочных символов.

Проверочные символы для кода Хемминга (7,4) формируются по схеме регистра сдвига, приведенного в [1, стр.254, фиг.8.3]. На алгоритмоподобном ячике этот алгоритм кодирования выглядит следующим образом. Пусть информационные биты кодовой последовательности образуют массив $COD [1 \cdot 4]$. А, В и С соответственно есть 1-й, 2-й и 3-й проверочные биты

$A := \text{ложь}; B := \text{ложь}; C := \text{ложь};$

для $i := 1$ шаг 1 до 4 цикл

начало

$X := COD[i];$

$$Y : = X \oplus A ;$$

$$A : = B ;$$

$$B : = Y \oplus A ;$$

$$C : = Y ;$$

конец ;

Здесь знак $\oplus$ означает оложение по модулю 2 .

Декодирование. Работа декодирующей программы состоит из трех этапов. На первом этапе восстанавливается структура информации, то есть выделяются блоки и зоны и проверяется длина каждой зоны. Поиск синхронизирующих последовательностей производится по двум байтам из трех, так как допускается выпадение одного из синхронизирующих символов. Если длина какой-либо зоны не совпадает с заданной, то структура зоны приводится к заданной. Зона дополняется нулевыми символами, если ее длина меньше заданной; если длина больше заданной, то последние символы отбрасываются. Таким образом, ошибки синхронизации превращаются в пакеты аддитивных ошибок. Затем вычисляются контрольные суммы информационных зон и сравниваются с контрольными суммами на перфоленге. Если контрольные суммы совпадают, то считается, что ошибок нет и информационные зоны передаются программе первичной обработки [4] . В противном случае управление передается на второй этап программы.

На втором этапе выбираются i -ые байты из всех зон блока. Они циклически сдвигаются на соответствующее число битов в направлении, противоположном первоначальному сдвигу и таким образом выделяются кодовые последовательности. Для каждой кодовой последовательности вычисляется синдром и производится коррекция одиночных ошибок. Биты синдрома вычисляются по той же схеме, по которой производится кодирование. Разница заключается лишь в том, что на вход подается вся кодовая последовательность (размерность массива $COD [1 : 7]$), а АВС есть двоичное представление синдрома.

На третьем этапе в направленном кодовом блоке снова сравниваются контрольные суммы информационных зон. Если они совпадают, то блок считается правильным. В противном случае блок считается неверным и выбраковывается, а на печать выдается сообщение об ошибке. Реализованная в системе САФРА программа декодирования позволяет исправить любую комбинацию ошибок в любой отдельно взятой зоне блока, а также любые комбинации одиночных ошибок в разных кодо-

вых последовательностях блока. Программа особенно критична к ошибкам синхронизации. Если ошибки синхронизации происходят одновременно в двух и более зонах кодируемого блока, то они не могут быть исправлены.

В заключение хотелось бы отметить, что реализованное в системе САФРА помехоустойчивое кодирование информации обладает большой избыточностью (из 143-х символов кодируемого блока только 64 являются информационными). Такая избыточность была продиктована необходимостью защиты от разного рода одиночных ошибок и пакетов ошибок при полном отсутствии информации о статистике ошибок. Предстоящий опыт эксплуатации системы позволит определить статистику ошибок, а это, в свою очередь, даст возможность воспользоваться наиболее экономным кодированием.

Л и т е р а т у р а

1. Питерсон У., Уэлдон Э. Коды, исправляющие ошибки. М., "Мир", 1976.
2. Мешковский К. А. Кодирование в технике связи. М., "Связь", 1966.
3. Гладких Б. А. Автоматизированная система регистрации, хранения и обработки экспериментальных данных САФРА - общее описание. Наст. сборник.
4. Буллер И. Я. Программы обслуживания архивов в системе САФРА, Наст. сборник.

ПРОГРАММЫ ОБСЛУЖИВАНИЯ АРХИВОВ В СИСТЕМЕ САФРА

И. Я. Буллер

I. Введение

Комплекс программ обслуживания архивов в системе САФРА [1] представляет собой ту часть подсистемы регистрации и хранения информации, которая реализуется на обрабатывающей ЭВМ. Этот комплекс выполняет следующие функции:

- преобразование данных из внешнего представления на носителях во внутреннее представление обрабатывающей ЭВМ;
- создание наборов данных, совместимых с САС [2], и запись их в архив;
- создание и обслуживание архива словарей.

Соответственно этим функциям комплекс программ обслуживания архивов состоит из трех основных блоков: блока анализа, блока создания наборов данных и блока обслуживания архива словарей. Взаимосвязь блоков между собой изображена на рис. 1.

Экспериментальные данные могут поступать на вход блока анализа либо в стандартном представлении канала передачи данных, либо в представлении для пятидорожечной перфокарты, либо в специальном представлении автономной регистрирующей аппаратуры [3]. В том и другом случае блок анализа производит обработку данных и запись их на промежуточный внешний носитель. Блок анализа организован как обычный загрузочный модуль, вызываемый в момент выполнения.

Совершенно иначе реализован блок создания наборов данных.

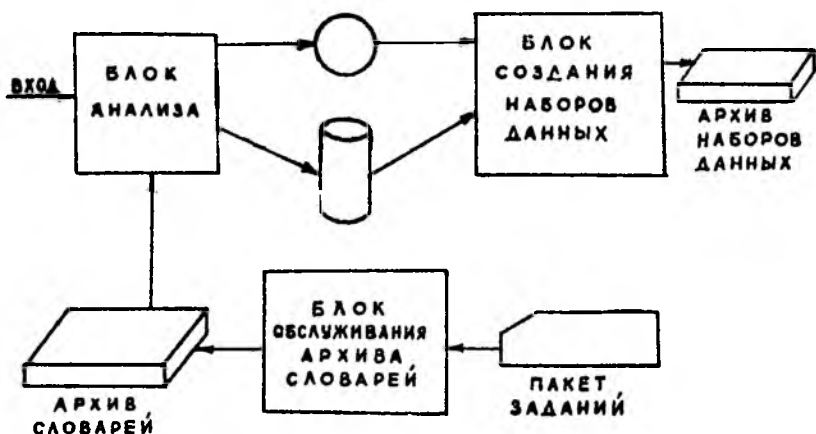


Рис. I

Так как структура записей, образующих наборы данных, определяется лишь в результате анализа, то естественно организовать блок создания наборов данных по принципу компиляции. Это означает, что в процессе выполнения задания исходя из структуры записей генерируется программа, которая представляет собой завершенный шаг задания, подсоединенный к выполняемому заданию. Цель генерации — создать каждый раз такой вариант блока создания наборов данных, который бы полностью отвечал структуре поступающих записей. Эта программа состоит из управляющих операторов и собственно программы, которую естественно сформировать из конструкций языка САС [2], поскольку в конечном счете должны быть созданы наборы данных, совместимые с системой САС, а в системе для этого имеются удобные средства. Генерация программы осуществляется в блоке анализа, таким образом, блок создания наборов данных существует только во время выполнения задания.

Блок обслуживания архива словарей переменных работает совершенно автономно. Этот блок создает архив словарей, добавляет в него новые словари, удаляет из архива ненужные словари или модифицирует их в случае необходимости.

2. Блоки анализа и создания наборов данных

Блок анализа выполняет следующие функции:

- преобразование данных из внешней формы представления во внутреннее представление обрабатывающей ЭВМ;
- формирование наблюдения;
- генерация блока создания наборов данных.

Преобразование данных из внешней формы представления. Если данные поступают в стандартной форме представления канала передачи данных, то выделение переменных происходит непосредственно по форматам, описание которых дано в словаре соответствующего набора данных. Несколько иной путь анализа информации, поступающей в представлении на пятидорожечной перфоленте либо специальном представлении. Поскольку структура данных для этих представлений значительно более сложная, то простое форматное распознавание переменных невозможно, требуется подробный синтаксический анализ входного текста. Для этой цели в блоке анализа имеется синтаксически управляемый анализатор автоматного типа. Формальный синтаксис входного языка описывается матрицей переходов автомата, поэтому, изменяя матрицу, можно настраивать анализатор на различные формы представления поступающих данных. Такая возможность представляется весьма полезной для работы с автономной регистрирующей аппаратурой: могут быть легко изменены алфавит входного языка, разделители, правила образования имен переменных, причем изменение входного языка приводит только к модификации синтаксической таблицы, а сама программа анализа не меняется.

Выделенные блоком анализа значения переменных переводятся во внутреннее представление обрабатывающей ЭВМ. Арифметические переменные представляются в плавающем формате двойной точности, а символьные переменные преобразуются по таблице кодировки из кода управляющей ЭВМ или из кода МТК-2 в код ДКОИ-8.

Формирование наблюдения. После того как запись выделена, проверяется, есть ли пропущенные значения переменных. Если таковые есть, то они подставляются в запись по принципу умолчания, указанному в словаре, и формируется наблюдение в логической форме представления. Сформированное текущее наблюдение записывается на

место предыдущего в оперативной памяти, а также на промежуточный внешний носитель.

Заметим, что экспериментальные данные могут поступать на блок анализа попеременно: сначала запись, относящаяся к одному набору данных, затем запись из другого набора данных и т.д.; блок анализа этого порядка не изменяет. Сортировку наблюдений по наборам данных и запись их в архив производит блок создания наборов данных.

Генерация блока создания наборов данных. Программа блока создания наборов данных состоит из текста на языке управления заданиями операционной системы ОС ЕС и текста на языке SAS. Текст на языке управления заданиями, в свою очередь, формируется из постоянной части, осуществляющей вызов супервизора SAS и переменной части, состоящей из DD - операторов для определения архивных наборов данных. Эта часть является переменной по той причине, что перед выполнением программы неизвестны имена наборов данных, запись которых поступают на обработку.

Каждое генерируемое DD предложение в случае, когда архив хранится на магнитной ленте, имеет следующий вид:

```
// имя_набора_данных DD UNIT=5010, VOL=SER=ARCHIV,  
// DISP=(,CATLG), LABEL=(PPP,SL,PASSWORD),  
// DSN=имя_набора_данных
```

Имя набора данных, заносимое в DD - оператор, формируется сцеплением символьной отроки 'FILE' с номером набора данных. Например, если номер набора данных есть 12, то его имя будет 'FILE12'. Кроме того, в DD - оператор заносится PPP - очередной доступный номер набора данных на архивной ленте. SAS - программа, генерируемая блоком анализа, предоставляет следующую последовательность операторов SAS:

```
PROC SASDS OUT=имя_набора_данных-1;  
PROC PRINT;  
.....  
PROC SASDS OUT=имя_набора_данных-N;  
PROC PRINT;
```

Эта программа выполняется под управлением супервизора SAS. Процедура SASDS представляет собой специальную служебную программу,

оформленную по правилам SAS - процедуры и включенную в SAS - библиотеку. Она написана на языке *PL/1* и содержит в себе обращения к подпрограммам SAS [4], с помощью которых создается и записывается в архив набор данных о имени, определенном в параметре *OUT*. Процедура *PRINT* осуществляет контрольную распечатку созданного набора данных. Процедуры *SASDS* и *PRINT* выполняются для каждого набора данных, записи которых поступили на вход блока анализа.

3. Блок обслуживания архива оловарей

Этот блок выполняет следующие функции:

- инициализация архива оловарей;
- запись в архив новых оловарей;
- модификация оловарей в архиве;
- удаление оловарей из архива;
- печать оловарей архива.

Словарь переменных заводится для каждого набора данных, номер словаря соответствует номеру набора данных *N*, взятому из имени *FILEN* [3].

Инициализация архива оловарей. При выполнении этой функции указывается режим *INPUT*, который отводит место на диске под архив оловарей и его оглавление.

Запись в архив оловарей. Для записи оловаря в архив указывается режим *ADD* и номер словаря, под которым он будет находиться в архиве. Если в архиве не найдется места для записи очередного словаря, то выдается сообщение: " В АРХИВЕ НЕТ МЕСТА ДЛЯ СЛОВАРЯ номер словаря ".

Модификация оловарей в архиве. При модификации словаря в архиве указывается режим *MOD* и номер словаря, который требуется модифицировать. В этом режиме указанные строки словаря заменяются на новые строки из пакета заданий. Если в архиве нет словаря с указанным номером, то выдается сообщение:

" В АРХИВЕ НЕТ СЛОВАРЯ номер словаря "

Удаление оловарей из архива. Эта функция выполняется при

указании режима *DELETE* и номера удаляемого оловаря. Если в архиве нет словаря, который требуется удалить, то выдается сообщение:

" В АРХИВЕ НЕТ СЛОВАРЯ номер словаря "

Печать оловарей архива. Печать архива оловарей выполняется при указании режима *PRINT* . Производится распечатка всех оловарей, находящихся в данный момент в архиве.

Обслуживание архива словаря осуществляется автономной программой *DICTVAR* , доступ к которой происходит через специальную каталогизированную процедуру *ARCHIV* . Пакет заданий для этой программы оформляется в виде колоды перфокарт. Каждое задание состоит из карты режима и для режимов *ADD* и *MOD* карт с текстами оловарей.

В карте режима в колонках I - 6 указывается режим, а в колонках 7 - 10 - номер обрабатываемого словаря (в режимах *INPUT* и *PRINT* номер словаря указывать не нужно). Карты с текстами словарей перфорируются так, что каждая строка словаря занимает одну карту, формат перфорации следующий:

колонки

I - 3	номер переменной ;
4 - 11	имя переменной ;
12 - 23	формат ;
24 - 39	принцип умолчания ;
40 - 43	единицы измерения ;
44 - 80	примечание.

Признаком конца словаря служит пустая карта.

П р и м е р. Пусть требуется занести в архив словарь переменных набора данных *FILE 2* , описанного в [3] , кроме того, указать новый принцип умолчания для переменной *TYPE* из набора данных *FILE 1* [1] , удалить некоторый словарь набора данных *FILE 3* и, наконец, распечатать скорректированный архив оловарей. Это может быть сделано в результате выполнения задания *EXAMPLE* , текст которого приведен на рис.2.

1	5	10	15	20	25	30	35	40	45	50	55
1. I. E. X. A. M. P. L. E.				J. O. B.	2. 4. 0. 4. 7. 7.		' B. Y. A. A. E. P. '				
1. I. A. R. C. H. I. V.				E. X. E. C.	A. R. C. H. I. V.		N. A. M. E = D. I. C. T. V. A. R.				
1. I. A. R. C. H. I. V.	3	Y. S. I. N.	D. D.	*							
A. D. D.	2										
1. D. A. T. E.		F. (8.)						Д. А. Т. А.			
2. N. A. M. E.		F. (10.)		И. Б. А. Н. О. В.				Ф. А. М. И. Л. И. Я.			
3. T.		E. (2, 4.)		M. I. S. S.				Г. Р. К. Т. Е. М. П. Е. Р. А. Т. У. Р. А.			
4. U.		E. (2, 4.)						В.	Н. А. П. Р. Я. Ж. Е. Н. И. Е.		
5. I.		E. (2, 4.)						А.	Т. О. К.		
0. R.		E. (2, 4.)		2. 0. 0. 0.				О. М.	С. О. П. Р. О. Т. И. В. Л. Е. Н. И. Е.		
M. O. D.	1										
1. T. Y. P. E.		F. (4.)		D. 2. 0. 2.				Т. И. Г. Д. И. Ю. Д. А.			
D. E. L. E. T. E. 3.											
P. R. I. N. T											
1. *											
1. /.											

Рис. 2

Л и т е р а т у р а

1. Гладких Б.А. Автоматизированная система регистрации, хранения и обработки экспериментальных данных "САФРА" - общее описание. Наст. сборник.
2. Матушевский В.В. Применение системы статистического анализа САС для обработки экспериментальных данных в системе САФРА. Наст. сборник.
3. Гладких Б.А., Матушевский В.В. Представление данных в системе САФРА. Наст. сборник.
4. Service J. SAS User's Guide. Published by Institute of Statistics North Carolina State University Raleigh, N.Carolina 27607, 1972.

ПРИМЕНЕНИЕ СИСТЕМЫ СТАТИСТИЧЕСКОГО АНАЛИЗА САС ДЛЯ ОБРАБОТКИ ЭКСПЕРИМЕНТАЛЬНЫХ ДАННЫХ В СИСТЕМЕ САФРА

В.В.Матушевский

1. Введение

Обработка экспериментальных данных требует развитого программного обеспечения, способного удовлетворить разнообразные запросы пользователей. За основу программного обеспечения в системе САФРА [3] принята система статистического анализа САС [1] .

Данные, переданные ЕС ЭВМ по каналу связи из подсистемы управления экспериментом, подвергаются первичной обработке с помощью специальной программы [5] . Эта программа декодирует поступающие данные и в соответствии со словарями переменных [4] преобразует их в архив наборов данных САС.

Система САС, разработанная в Институте статистики университета штата Северная Каролина под руководством А.Дж.Барра и Дж.Г.Гуднйта, предоставляет пользователю возможности создания и хранения наборов данных, их редактирования, поиска и сортировки, статистической обработки данных. САС ориентирована на работу на машинах серии IBM 360/370, которым соответствуют модели машин серии ЕС и работает в операционных системах ОС PCP, MFT и MVT. Минимальный объем оперативной памяти для САС составляет 120 К.

Система представляет собой компилятор с собственного языка САС и библиотеку процедур, написанных на языках ПЛ/1, Фортран и Ассемблер. Среди наиболее важных процедур САС имеются гибкие процедуры метода наименьших квадратов, процедуры регрессионного

анализа, дисперсионного и корреляционного анализа. Другие процедуры производят дискриминантный анализ, факторный анализ, служат для вычисления статистических характеристик, таблиц частот и их анализа, построения гистограмм и т.д. Часть процедур предназначена для представления результатов-распечатки таблиц и графиков на АЦПУ.

Пользователь может расширять библиотеку процедур, создавая нужные ему процедуры на языках ПЛ/I, Фортран, Ассемблер. Для включения процедур пользователя в САС предусмотрены специальные средства.

Ниже приводится краткое описание языка и процедур САС. Полное описание языка и процедур САС можно найти в справочнике, изданном разработчиками системы [1]. Средства включения процедур пользователя в САС описаны в руководстве для программиста САС [2].

2. Язык САС

Набор данных САС. САС оперирует над наборами данных, которые можно представить в виде двумерных матриц. Строкам матрицы соответствуют **наблюдения**, столбцам — **переменные**. Число переменных набора данных САС ограничено до 255, число наблюдений не ограничено.

Наблюдение состоит из **1 - 255 элементов данных**, каждый из которых является значением соответствующей переменной. Переменные могут быть двух типов — строковые и арифметические. Преобразование переменной из строковой в арифметическую и наоборот — недопустимо. Строковый элемент содержит до 64 символов (символ ";" нельзя включать в строковый элемент). Арифметический элемент данных представляется на перфокартах и АЦПУ в виде

[+][целая часть][.][дробная часть][E ± xx],

а на лентах и дисках — во внутреннем представлении ЕС ЭВМ в плавающем формате с двойной точностью.

Пользователь должен назначать уникальные **имена** наборам данных, переменным, макроопределениям и меткам. Имена содержат до 8 символов. Первым символом имени должна быть буква латинского алфавита или подчерк. Специальные символы (в том числе и буквы русского алфавита) и пробелы внутри имени недопустимы. Слова **MACRO** и **COMMENT** не могут быть именами. В наборе данных САС

могут встретиться пропущенные значения. Способы обработки пропущенных значений приводятся в описаниях операторов и процедур.

Операторы САС. При описании операторов САС будем использовать следующие обозначения и правила. Слово из прописных букв является именем оператора и не может быть изменено. Слова из строчных букв сообщают, какой вид информации пользователь должен поместить и соответствующие позиции. Выражение, заключенное в [], необязательно. При описании каждого оператора указывается, какие действия будут выполнены при опущенном необязательном выражении.

Каждый терм оператора САС отделяется от другого пробелом (пробелами) или специальным символом. Каждый оператор должен заканчиваться точкой с запятой.

САС - программа выполняет два вида работ: создание новых наборов данных, временных или постоянных, и обработку наборов данных процедурами. Новый набор данных можно создавать из уже существующих наборов данных САС, из данных, записанных в формате пользователя, а также генерировать самой программой. Обычно программа САС состоит из блоков, каждый из которых выполняет оба вида работ - создание и обработку набора данных. Блок начинается оператором *DATA*, который объявляет, что начинается создание нового набора данных. За оператором *DATA* может следовать только один из операторов: *SET*, *MERGE* или *INPUT*. С помощью операторов *SET* и *MERGE* программа получает доступ к уже существующим наборам данных САС, а с помощью оператора *INPUT* - к данным, записанным в формате пользователя. Это означает, что программе передаются имена переменных, их типы, форматы и элементы данных каждого наблюдения. Следующие затем программные операторы обрабатывают каждое наблюдение по отдельности. Наблюдение включается в создаваемый набор данных, когда будут выполнены все программные операторы или оператор *OUTPUT*. Программные операторы могут изменять значения переменных каждого наблюдения, могут создавать новые переменные и наблюдения и удалять существующие. Создание набора данных заканчивается, когда последний программный оператор обрабатывает последнее наблюдение. Когда создание набора данных завершено, его можно обработать процедурами САС или запомнить на внешнем носителе. Набор данных является входным для процедуры и не может быть ею преобразован. Процедура может создать другой набор данных, если это

предусмотрено в её описании.

Все операторы САС условно можно разделить на две группы по области их действия. Одна группа операторов действует на наборы данных в целом. Область действия операторов другой группы находится внутри набора данных (разработчики САС называют их *программными операторами*). Операторы этой группы воздействуют на переменные текущего наблюдения или на всё наблюдение.

Операции над наборами данных. К первой группе относятся операторы *DATA* , *SET* , *MERGE* , *INPUT* , *DRCP* и *PROCEDURE*

Оператор

DATA имя набора данных ;

предписывает САС создать новый набор данных с именем *имя_набора_данных*. Если *имя_набора_данных* опущено, то система сама назначает имена *DATA001* , *DATA002* и т.д. За оператором *DATA* всегда должен следовать один из операторов

SET, *MERGE* или *INPUT*

Оператор

SET имя_набора_данных ;

включает в создаваемый набор данных уже существующий набор данных с именем *имя_набора_данных*. Если *имя_набора_данных* опущено, то САС использует последний созданный набор данных для образования нового. Если *имя_набора_данных* имеет вид *FILEN* , где *N* - не более чем четырехразрядное целое число, то оператор *SET* возьмет набор данных из архива системы САФРА, при этом будут переданы не только значения переменных, но также их имена и форматы. В дальнейшем пользователь может обращаться к переменным этого набора данных просто по именам.

Пример. Пусть требуется считать из архива системы САФРА набор данных с именем *FILE15* и включить его в создаваемый набор данных *TREAT* . После преобразования *TREAT* , в свою очередь, включается в набор данных *TEST* .

DATA TREAT;

SET FILE15;

.....

DATA TEST;

SET;

.....

Первый оператор *SET* включит *FILE15* в набор данных *TREAT* ; второй оператор *SET* (без указания имени) включит *TREAT* в набор данных *TEST* (так *TREAT* был последним ранее созданным набором данных).

Оператор

MERGE набор_данных_1 набор_данных_2;

служит для включения в создаваемый набор объединения двух уже существующих. Переменными объединенного набора будут переменные из обоих наборов. Если оба набора данных содержат переменные с одинаковыми именами, то они должны быть одного типа (обе арифметические или обе строковые).

Оператор

INPUT [DDNAME =] Описание формата данных;

применяется тогда, когда у пользователя появляется необходимость ввести набор данных не из архива, а непосредственно с перфокарт, либо с другого носителя. В этом случае пользователь обязан сам определить формат вводимых данных. *DDNAME* - это имя *DD* (оператора определения наборов данных в языке управления заданиями операционной системы) вводимого набора данных. Если данные вводятся с перфокарт, то *DDNAME* опускается.

Оператор

DROP переменная-1 [переменная-2 ... переменная-*n*];

предписывает исключить из набора данных переменные, указанные в списке.

Обращение к процедурам *SAS* осуществляется с помощью оператора *PROCEDURE* (допускается сокращение *PROC* , который имеет вид

PROCEDURE имя-процедуры [опция-1 ... опция-*n*]
[параметр-1 ... параметр-*k*] [*DATA* =имя-набора-данных]
[*OUT* = имя-набора-данных] ;

Процедура может обрабатывать и создавать только по одному набору данных. Процедура обрабатывает набор данных, указанный в параметре *DATA* ; если этот параметр опущен, то обрабатывается последний созданный набор данных. Параметр *OUT* определяет имя набора данных, создаваемого процедурой. В каждой процедуре можно описать до 20 опций и 20 параметров. Опции - это просто имена. В описании каждой процедуры указано, что будет сделано, если опции указаны или опущены. Параметры записываются в виде "имя-".

Действие параметров также приводится в описании каждой процедуры. Опции и параметры можно задавать в любом порядке.

Пример *PROCEDURE REGR SIMPLE CORR;*

REGR - это имя процедуры, *SIMPLE* и *CORR* - опции. Процедура обрабатывает последний созданный набор данных.

PROC FACTOR TR N=4 DATA = QUEST;

Процедура *FACTOR* обрабатывает набор данных *QUEST*. Задана опция *TR*, параметр *N* принимает значение 4.

В SAS-программе за оператором *PROC* обычно следуют спецификации процедуры, другой оператор *PROC* или оператор *DATA*. При обращении к процедуре могут быть указаны её спецификации, которые определяют дополнительные возможности обработки. Имеется пять основных спецификаций, которые применимы ко многим или всем процедурам библиотеки SAS.

Спецификация *CLASSES* (допускается сокращение *CLASS*) указывает, что перечисленные в ней переменные считаются классификационными. Значения этих переменных используются только для различия групп наблюдений, над ними не производятся никакие вычисления. Общий вид этой спецификации:

CLASSES переменная-1 [переменная-2 ... переменная-*m*];

Спецификацию *VARIABLES* (допускается сокращение *VAR*) можно использовать во всех процедурах, за исключением *PLAN*, *GUTTMAN* и *SORT*. Процедура обрабатывает только те переменные, которые указаны в списке. Эта спецификация имеет вид
VARIABLES переменная-1 переменная-2 ... переменная-*m*);

Спецификация

DROP переменная-1 переменная-2 ... переменная-*n* ;
исключает из обработки (но не из набора данных) переменные, указанные в списке. Спецификацию *DROP* можно считать обратной к *VARIABLES*.

Действие спецификации

ID список переменных;

зависит от процедуры, в которой она используется. Обычно *ID* связана с распечаткой отдельных наблюдений.

Спецификация

BY список переменных;

имеет специальное назначение для процедуры *SORT*. Для остальных процедур *BY* предписывает обрабатывать данные по группам наблюдений с одинаковыми значениями переменных, указанных

в списке.

Программные операторы преобразуют значения переменных текущего наблюдения перед тем как оно будет включено в создаваемый набор данных. Прежде чем перейти к описанию этих операторов, рассмотрим пример, показывающий их действие.

Пусть набор данных *IN* с переменными *AL*, *SEA* и *JU* состоит из трех наблюдений

	<i>AL</i>	<i>SEA</i>	<i>JU</i>
Наблюдение 1	27	10	2
Наблюдение 2	33	6	29
Наблюдение 3	34	8	18

Следующая программа создает новый набор данных *ON* из входного *IN*, а затем обрабатывает его процедурой *MEANS*

```
DATA ON; SET IN;  
AL = AL - JU;  
IF AL < 5 THEN DELETE;  
FR = SEA / AL;  
PROC MEANS;
```

Оператор *DATA* предписывает создать новый набор данных с именем *ON*. Оператор *SET* включает набор данных *IN* в *ON*. Первый программный оператор предписывает присвоить *AL* значение, равное $AL - JU$. Второй оператор указывает, что если значение *AL*, вычисленное первым оператором, меньше 5, то это наблюдение нужно исключить из набора данных. Третий программный оператор предписывает создать новую переменную с именем *FR* и присвоить ей значение SEA / AL .

Рассмотрим поэтапно работу программных операторов. Сначала просматривается первое наблюдение, переменные которого имеют значения

<i>AL</i>	<i>SEA</i>	<i>JU</i>	<i>FR</i>
27	10	2	

После выполнения первого программного оператора наблюдение становится таким

<i>AL</i>	<i>SEA</i>	<i>JU</i>	<i>FR</i>
25	10	2	

Второй оператор проверяет условие и оставляет наблюдение в наборе данных, так как $AL > 5$. Третий оператор назначает *FR* значение 0.4. После выполнения всех трех программных операторов

наблюдение становится

AL	SEA	JU	FR
25	10	2	0.4

После этого просматривается второе наблюдение и для него повторяются те же операции. Так как при этом значение AL равно 4, то наблюдение не включается в набор данных ON. Третье наблюдение обрабатывается так же как и предыдущее. В результате получается набор данных

	AL	SEA	JU	FR
Наблюдение 1	25	10	2	0.4
Наблюдение 2	16	8	18	0.5

Перейдем теперь к описанию программных операторов.
Оператор присваивания

имя_переменной = выражение;

предписывает присвоить переменной значение выражения, которое должно соответствовать типу переменной; несоответствие типов недопустимо. Если имя переменной впервые встречается в правой части оператора присваивания, то его значение считается пропущенным. Если оно впервые встречается в левой части оператора (см. пример), то создается новая переменная и её значение включается в текущее наблюдение с именем имя_переменной. Выражение определяется как последовательность имен переменных, констант и/или функций, связанных знаками операций и круглыми скобками. Выражение вычисляется посредством выполнения арифметических и логических операций, операций сравнения и вычисления встроенных функций. Знаки всех операций соответствуют принятым в языке III/I, за исключением операции NO. Операция NO применяется для отыскания пропущенных значений: если NO применена к пропущенному значению, то результат есть 1, в противном случае 0. Порядок выполнения операций обычный. Для изменения порядка выполнения операций используются круглые скобки.

В САС имеется ряд встроенных функций, включающий арифметические и тригонометрические функции, генераторы псевдослучайных чисел, специальные функции, обрабатывающие пропущенные значения.

Оператор *DELETE* ; предписывает САС не включать текущее наблюдение в набор данных. Обычно *DELETE* выполняется в условном операторе.

Оператор *OUTPUT* ; предписывает немедленно включить теку-

щее наблюдение в набор данных, то есть образовать наблюдение из всех текущих значений всех переменных (за исключением перечисленных в *DROP*) и добавить это наблюдение к набору данных.

Условный оператор

IF выражение *THEN* оператор;

Оператор выполняется, если значение выражения есть 1.

Оператором может быть любой программный оператор, за исключением *IF* и *DROP*. Специальный оператор

IF выражение;

эквивалентен оператору *IF NOT* (выражение) *THEN DELETE*;

Любой оператор *CAS* может быть помечен меткой.

Метка: Оператор;

Оператор метка; передает управление на оператор, помеченный меткой.

Оператор

LINK метка;

передает управление на оператор, помеченный меткой. Различие в использовании *GO TO* и *LINK* определяется применением оператора *RETURN*

Результат применения оператора *RETURN*; в свою очередь, зависит от того, как используется *LINK*:

1. Если не выполнялся оператор *LINK* или если *RETURN*

уже был выполнен для всех *LINK*, то *RETURN* становится последним программным оператором.

Это означает, что *CAS* выполнит все операторы, предшествующие *RETURN*, включит наблюдение в набор данных и обратится к новому наблюдению, как это предписывают операторы *SET*, *MERGE* или *INPUT*.

2. Если был выполнен оператор *LINK*, то первый встретившийся после него *RETURN* возвращает управление на оператор, следующий за этим *LINK*.

Операторы *LIST*, *PUT* и *ERROR* служат для печати наблюдения или некоторых переменных наблюдений. Оператор *TITLE* служит для печати заголовков в начале каждой страницы, а оператор *COMMENT* — для печати комментариев в тексте программы.

Макроопределения. Для повторного использования одних и тех же операторов и описаний пользователь может применять макроопределения. Макроопределение задается следующим образом:

MACRO имя_макро информация %

Имя_макро должно быть правильным SAS-именем. Информация может содержать любые операторы SAS, другие *MACRO* (глубина вложенности *MACRO* - не более 5). Обращение к макроопределению производится по имени имя_макро.

Примеры.

1) *MACRO* *x* *x1* *x2* *x3* %

Здесь *x* - имя_макро. Например, оператор *DROP x*; эквивалентен *DROP x1 x2 x3*;

2) *MACRO* *rom* $x2 = x * x$; $y2 = y * y$;

$xy = x * y$ %

Обращение к такой программе - *rom*;

3. Библиотека процедур SAS

Полное описание процедур приводится в [1]. Здесь приводятся лишь имена и назначение процедур

Имя процедуры	Назначение
<i>PRINT</i>	Распечатка набора данных
<i>SORT</i>	Сортировка набора данных.
<i>RANK</i>	Ранжирование значений переменных.
<i>PLOT</i>	Построение графиков.
<i>MEANS</i>	Вычисление одномерных статистических характеристик.
<i>REGR</i>	Построение одно- и многомерных регрессионных моделей по методу наименьших квадратов; вычисление простых, множественных и частных коэффициентов корреляции.
<i>RSQUARE</i>	Построение "всех возможных" регрессий.
<i>STEPWISE</i>	Пять методов выбора независимых переменных для регрессионной модели.
<i>ANOVA</i>	Вариационный анализ.
<i>DUNCAN</i>	Тест Дункана для множественных рангов.
<i>NESTED</i>	Вариационный и ковариационный анализ - вложенные модели.
<i>LATTICE</i>	Вариационный и ковариационный анализ.
<i>RQUE</i>	Несмещенные квадратичные оценки.
<i>CANCOR</i>	Канонический корреляционный анализ.

DISCRIM	Дискриминантный анализ.
FACTOR	Факторный анализ; анализ главных компонент.
CORR	Коэффициенты корреляции.
SPEARMAN	Ранговые коэффициенты корреляции Спирмена; коэффициенты корреляции Кендалла.
FREQ	Таблицы частот; тест хи-квадрат для таблиц сопряженности признаков с двумя входами.
GUTMAN	Создание и оценка масштабной модели Гуттмана
PLAN	Рандомизированные планы экспериментов.
PRTPCN	Вывод наборов данных в заданном формате.
HARVEY	Вычисление по методам наименьших квадратов и максимума правдоподобия.
RENAME	Изменение имен переменных в наборе данных.
QUESTN	Таблицы частот и сопряженности признаков.
EXPLODE	Печать символов в увеличенном масштабе.
PROBIT	Пробит - анализ данных.
STANDART	Стандартизация экспериментальных данных.
HIST	Печать гистограмм.
KSLTEST	Проверка на нормальность.

В каждой процедуре определено несколько опций и параметров, поэтому кажущийся небольшим список процедур охватывает довольно широкую область статистической обработки данных.

4. Хранение наборов данных SAS и доступ к ним

Подсистема регистрации и хранения информации системы САФРА создает архив SAS-наборов данных на магнитной ленте или диске и обеспечивает доступ к ним. Наборы данных каталогизируются в каталоге операционной системы. Имена всех наборов данных архива состоят из слова *FILE* с добавлением не более чем четырехразрядного номера соответствующего файла (например *FILE 1*, *FILE 2035*).

Задание на выполнение SAS-программы, как и любое другое задание в ОС ЕС, должно содержать операторы языка управления заданиями [6]. Простейшее SAS-задание содержит следующую последовательность операторов:

```
" имя_задания JOB учетная_информация, фамилия_программиста
" EXEC SAS
" SAS.SYSIN DD .
```

САС-программа

/*

//

Используемая здесь каталогизированная процедура SAS дает возможность выполнить САС-программу, не требующую ввода данных из архива. Чтобы получить доступ к наборам данных из архива, пользователь должен применять каталогизированную процедуру SAFRA

Пример.

```
// READ JOB 037805, STUDENT
// EXEC SAFRA
//SAFRA.SYSTN DD *
  PROC MEANS DATA = FILE3;
  .....
```

/*

//

В этом примере набор данных FILE3, содержащийся в архиве, обрабатывается процедурой MEANS.

Тот же самый набор данных FILE3 можно использовать для создания нового набора данных, например

```
// SET JOB 037805, STUDENT
// EXEC SAFRA
//SAFRA.SYSTN DD *
  DATA FIX;
  SET FILE3;
  .....
  Операторы САС
  .....
```

/*

//

В этом примере FILE3, содержащийся в архиве, включается в создаваемый временный набор данных FIX.

Для создания постоянного набора данных на внешнем носителе в последовательности упорядочивающих операторов нужно включить оператор DD (определения данных), описывающий этот набор данных. Если набор данных записывается как последовательный файл, то имя должно быть именем набора данных.

Пример.

```
// STORE JOB 037805, STUDENT
```

```
// EXEC SAFRA
// SAFRA.RESULT DD DSN = dsn , UNIT = unit , VOL = SER = v ,
// DISP = (NEW, CATLC), SPACE = (2048, (a, b))
// SAFRA.SYSIN DD *
    DATA RESULT ;
    SET FILE 15 ;
    .....
Операторы САС
    .....
```

/*
//
Здесь набор данных *FILE15* из архива включается в создаваемый набор данных *RESULT*, который записывается на внешний носитель. *dsn* - это имя набора данных на томе *v*, *unit* - серийный номер внешнего устройства; *a* и *b* определяются числом наблюдений и числом переменных. Если набор данных записывается на магнитную ленту, то параметр *SPACE* нужно опустить.

Точно так же набор данных пользователя записывается в раздел библиотеки. При этом оператор *DATA* записывается в виде

DATA имя_DD (имя_раздела_библиотеки);

а параметр процедуры

OUT =имя_DD (имя_раздела_библиотеки).

5. Включение процедур пользователя в САС

Очевидно, что широкое применение средств автоматизации эксперимента вызовет необходимость в специальной обработке данных, не предусмотренной разработчиками САС. В связи с этим в САС имеются средства включения процедур пользователя, написанных на языках III/I, Фортран, Ассемблер. По существу, вызов САС-процедуры представляет собой вызов соответствующего загрузочного модуля с передачей ему опций и параметров и обеспечением обмена данными между набором данных САС и этим модулем.

Компилятор САС, просматривая обращение к процедуре и спецификации, составляет таблицы опций, параметров, атрибутов переменных. Информацию об атрибутах переменных компилятор получает из паспорта набора данных, который хранится вместе с набором. В паспорте содержится описание переменных - их имена, типы и длины в байтах. Анализируя спецификации, компилятор классифицирует пере-

менные по типу спецификации, в списке которой они перечислены. Это позволяет пользователю обрабатывать переменные в соответствии со спецификацией. Номер класса переменной входит во многие служебные подпрограммы как параметр. Эти служебные подпрограммы и обеспечивают связь между программой пользователя и компилятором САС. Назначение служебных программ таково:

- получение информации об опциях и параметрах САС-процедуры и атрибутах переменных ;
- динамическое распределение памяти ;
- чтение данных из САС-набора данных ;
- создание новых САС - наборов данных.

Следует отметить, что процедуре пользователя не нужна информация об имени и расположении входного набора данных, так как оператор *PROC* обращается к компилятору САС, который находит набор данных и считывает каждое наблюдение.

Определение опций и параметров. Для передачи опций и параметров оператора *PROC* программе пользователя используются функции *IOPT* и *PARM*. Функция *IOPT* определяет, указана ли соответствующая опция в обращении к процедуре. Функция *PARM* передает программе значения параметров процедуры, в том числе имена входного и создаваемого наборов данных.

Определение переменных. Подпрограмма *NOVAR* определяет число переменных соответствующего класса, а подпрограмма *NAMES* определяет имена и атрибуты переменных.

Динамическое распределение памяти позволяет избавиться от описания статических массивов в программе пользователя. В программе вычисляется необходимая суммарная размерность массивов. Затем подпрограмма *GETSTR* отводит необходимую память для базового массива, на который накладываются массивы программы.

Чтение данных из САС-набора данных. Подпрограмма *INPUT* осуществляет доступ программы пользователя к очередному наблюдению САС - набора данных. Подпрограммы *VAR*, *VARX*, *VARY* считывают из набора данных; соответственно элемент данных, вектор элементов данных и строку данных длиной до 80 байт. Вызов подпрограммы *RWWD* устанавливает доступ к первому наблюдению набора данных.

Создание набора данных процедурой. Процедура может создавать

только один набор данных САС. Переменные нового набора данных могут быть созданы процедурой или быть переменными из входного набора, либо теми и другими вместе. Подпрограмма *OPNOUT* открывает выходной набор данных и определяет его имя. Подпрограмма *TNAMES* передает имена переменных входного набора данных выходному. Подпрограмма *ONAMES* определяет имена и атрибуты выходных переменных. Подпрограмма *ENDNAM* завершает определение переменных выходного набора данных. Подпрограмма *PUTREC* устанавливает положение наблюдения в выходном наборе данных, *TRANSFER* передает текущее входное наблюдение выходному набору данных, *MOVEDA* передает атрибуты переменных текущему выходному наблюдению, *CLSOUT* закрывает выходной набор данных. Подпрограммы *SVAR* и *SVARX* передают соответственно элемент данных и вектор элементов данных выходному наблюдению. Комбинацией обращения к этим подпрограммам пользователь может создавать своей процедурой набор данных, состоящий из переменных входного набора, из всех новых переменных либо из тех и других вместе.

6. Заключение

Использование САС в качестве базового программного обеспечения в значительной степени определяет возможности обработки экспериментальных данных в системе САФРА.

Система обеспечивает стандартное представление данных, создание архива данных [5] и стандартный доступ к ним из обрабатываемых процедур.

Система позволяет с помощью относительно простого языка САС привести экспериментальные данные к виду, удобному для обработки.

Библиотека процедур САС дает широкие возможности статистической обработки данных.

Средства включения процедур пользователя в систему позволяют расширять возможности обработки за счет использования разработанных к настоящему времени систем и пакетов прикладных программ, написанных на разных языках (например, пакет *SSP* [7] на Фортране и III/I, система математического обеспечения графического вывода СМОГ [8] и другие) и программ пользователя.

Л и т е р а т у р а

1. Service J.SAS User's Guide. Published by Institute of Statistics North Carolina State University. Raleigh, N.Carolina 27607, 1972.
2. Barr Q.J., Goodnight J.H. SAS Programmer's Guide. Published by Institute of Statistics North Carolina State University. Raleigh, N.Carolina 27607, 1972.
3. Гладких Б.А. Автоматизированная система регистрации, хранения и обработки экспериментальных данных САФРА - общее описание. Наст.сборник.
4. Гладких Б.А., Матушевский В.В. Представление данных в системе САФРА. Наст.сборник.
5. Буллер И.Я. Программы обслуживания архивов в системе САФРА. Наст.сборник.
6. Система математического обеспечения ЕС ЭВМ. Под ред. А.М. Ларионова. М., "Статистика", 1974.
7. Сборник научных программ на Фортране. Статистика. Вып. I. Матричная и линейная алгебра. Вып. 2, М., "Статистика", 1974.
8. Костюк Д.Д. Система математического обеспечения графического вывода для ЕС ЭВМ. Томск, Изд-во Томского университета, 1977.

СИСТЕМА ВЫВОДА СЛОЖНЫХ ИЗОБРАЖЕНИЙ НА АППУ ЕС ЭВМ (на базе ПЛ/I ОС)

Ю.В.Потапов

Существующие в распространенных языках программирования (ПЛ/I, ФОРТРАН) средства вывода на АППУ реализуют лишь один способ формирования изображений – строчный. По-видимому, это связано с тем, что АППУ производит только построчную печать. Однако строчный принцип не всегда естествен для пользователя. При получении сложных изображений типа таблиц, рисунков, графиков удобнее конструировать их не построчно, а в произвольном порядке, так, как это диктуется структурой изображения.

В свое время для ЭВМ второго поколения была создана и завоевала достаточную популярность система [1], дающая такую возможность. В настоящей статье предлагается аналогичная система для ЕС ЭВМ (на базе языка программирования ПЛ/I ОС). Для краткости назовем ее системой ВАЦ (Вывод Алфавитно-Цифровой).

1. Описание системы

Принцип работы. В системе ВАЦ все изображение предварительно формируется в памяти ЭВМ и только потом распечатывается на АППУ. Для этого сначала вводится двумерный литерный массив, как мы будем говорить, задается лист необходимых размеров (т.к. на АППУ этому массиву соответствует прямоугольный лист бумаги длиной n строк АППУ и шириной 128 позиций АППУ, число строк N определяется пользователем). Каждый алфавитно-цифровой символ будущего изо-

бражения получает на листе две координаты – номер строки и номер позиции. Затем оредствами системы на лист по координатам записываются нужные фрагменты изображения. (Именно координатный принцип позволяет заносить фрагменты в произвольном порядке). Наконец, после того как весь лист с изображением оформирован, он выводится из памяти на печать.

Таким образом, ВАЦ обеспечивает, с одной стороны, любой удобный для пользователя порядок формирования изображения, а с другой стороны – нужный для АЦПУ построчный порядок вывода изображения на печать.

Организация системы. С точки зрения языка III/I ВАЦ представляет собой процедуру, ее распечатку см. в приложении I. (Заметим, что распечатка была получена на пультовой машинке с помощью оператора DISPLAY и потому является точной копией отлаженной программы). Эта процедура имеет четыре точки входа: LISTL, CIFL, TEXTL, PRINTL, которые предназначены для следующих целей. LISTL олужит для задания листа, TEXTL и CIFL – для записи на этот лист соответственно текстовых и числовых фрагментов, PRINTL – для распечатки листа на АЦПУ.

Для согласования атрибутов аргументов и параметров точек входа при конкретных обращениях к ВАЦ пользователь должен поместить в своей программе такой оператор:

```
DCL LISTL ENTRY(FIXED BIN, FIXED BIN, FIXED BIN),  
    CIFL ENTRY(FLOAT DEC(16), FIXED BIN, FIXED BIN, FIXED BIN), (I)  
    TEXTL ENTRY(CHAR(1000)VAR, FIXED BIN, FIXED BIN),  
    PRINTL ENTRY(FIXED BIN);
```

Впередь будем считать, что это требование выполнено.

Все преобразования арифметических аргументов при обращениях к ВАЦ происходят без округления (уочением дробных разрядов). Если требуется округление, пользователю следует применить к нужным аргументам встроенную функцию ROUND.

Рассмотрим более детально работу и формы обращения к системе.

Задание листа. Задание листа осуществляется путем обращения к точке входа LISTL:

```
CALL LISTL(N, IO, JO);
```

(2)

где N – число строк в листе;

IO, JO – номера соответственно строки и позиции (в абсолютной

системе координат листа, рис. I, а) точки начала относительной системы координат.

Все аргументы в (2) могут быть арифметическими выражениями.

Лист занимает в памяти $N \cdot 128$ байтов. В целях экономии память под лист выбрана управляемой.

Число строк в листе должно быть не меньше единицы, иначе ВАЦ выдаст сообщение:

ЛИСТ НЕ ЗАДАН: ЧИСЛО СТРОК <1
ПРОГРАММА ОСТАНОВЛЕНА

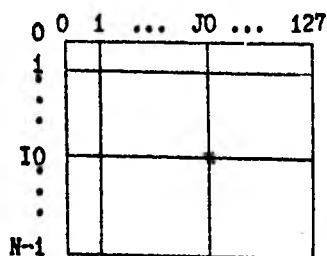
и прекратит выполнение программы пользователя.

Верхняя граница для числа строк заранее не известна. Она определяется имеющимся в момент обращения к LISTL резервом памяти ЭВМ. Если памяти под лист с нужным числом строк будет не хватать, операционная система выдаст сообщение:

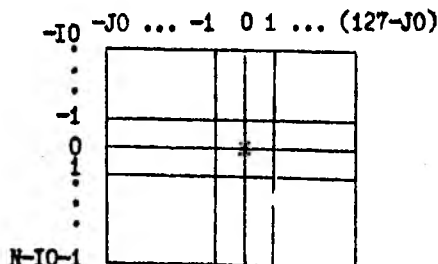
INEO12I <JOBNAME> ABENDED AT OFFSET 0001F6
FROM ENTRY POINT LISTL WITH CC 80A (SYSTEM)

и прекратит выполнение программы.

Во время задания листа на нем всегда вводится относительная система координат. Это делается для облегчения построения таблиц и графиков. В частном случае относительная система координат может совпадать с абсолютной. Нумерация строк и позиций листа в абсолютной (а) и относительной (б) системах координат показана на рис. I (крестиком отмечена точка начала относительной системы):



а)



б)

Рис. I. Нумерация строк и позиций листа

После размещения в памяти все поле листа заливается пробелами. В последующем ни один пробел на лист не пишется.

Время задания листа на машине ЕС 1020 составляет порядка

4 мсек на строку.

Занесение на лист текстовых фрагментов. Формирование изображения в системе ВАЦ осуществляется путем обращения к точкам входа TEXTL и CIPL. В любом случае, прежде чем обращаться к этим точкам, должен быть задан лист, иначе ВАЦ выдаст сообщение:

ВНИМАНИЕ ! НЕ ЗАДАН ЛИСТ
ПРОГРАММА ОСТАНОВЛЕНА

и прекратит выполнение программы.

Для записи на лист текстовых фрагментов в системе используется обращение к TEXTL:

CALL TEXTL(STRT,IS,JS); (3)

где STRT - текст (литерная строка), который требуется занести на лист;

IS,JS - номера соответственно строки и позиции (в относительной системе координат листа) первого символа из STRT. Первый аргумент в (3) может быть строчным выражением, второй и третий - арифметическими.

В качестве текста при обращении к TEXTL может выступать произвольная последовательность любых символов, имеющих в распоряжении программиста. Список литер, доступных пользователю, приведен в приложении 2. Однако следует знать, что в системе ВАЦ символ " (двойная кавычка) резервирован для особой цели: как только при обращении (3) знак " встретится внутри литерной строки STRT, часть текста, идущая за ним, будет перенесена на следующую строку листа в исходную позицию JS. (По действию это аналогично переводу строки с возвратом каретки на пишущей машинке). Сам символ " на лист не пишется.

Если при обращении к TEXTL часть фрагмента не попадает в поле листа, эта часть не будет записана, а ВАЦ выдаст сообщение:

<KT> ТЕКСТ УСЕЧЕН ,

где <KT> - порядковый номер обращения к TEXTL в программе пользователя, соответствующий этой диагностической печати.

Если же при обращении к TEXTL весь фрагмент не попадает в поле листа, он не будет записан, а ВАЦ выдаст сообщение:

<KT> ТЕКСТ ВНЕ ЛИСТА ,

где <KT> имеет тот же смысл, что и прежде.

Время одного обращения к TEXTL для машины ЕС 1020 составляет порядка 60 мсек.

Зачесение на лист числовых фрагментов. Для записи на лист числовых фрагментов в системе может быть использовано обращение к CIFL :

CALL CIFL(WC,LC,IC,JC); (4)

где WC - число, значение которого нужно записать на лист;

IC - точность представления числа;

IC,JC - номера соответственно строки и позиции (в относительной системе координат листа) явной или подразумеваемой десятичной точки числа.

Все аргументы в (4) могут быть арифметическими выражениями.

Обращение к CIFL позволяет в зависимости от величины точности IC выбрать не только число значащих цифр у WC, но также и формат записи числа WC на лист.

Если $LC > 0$, то WC будет записано в формате с фиксированной точкой, с числом цифр после точки равным IC. Если $LC = 0$, то WC будет записано в виде целого (без десятичной точки). Если же $LC < 0$, то WC будет записано в формате с плавающей точкой, с числом значащих цифр в мантиссе, равным абсолютной величине IC. Причем десятичная точка в мантиссе будет располагаться сразу за первой цифрой, а показатель степени будет выбран так, чтобы эта первая цифра была отлична от нуля.

В любом случае, если число WC отрицательно, перед первой его значащей цифрой будет записан на листе знак минус, в противном случае никакого знака не будет.

Нужно знать, что величина точности IC ограничена неравенствами:

$$-16 < LC < 15,$$

в противном случае, если окажется, что $IC < -16$, ВАЦ установит точность $IC = -16$ и выдаст сообщение

<K> ЧИСЛО НЕ ПРЕДСТАВИМО С УКАЗАННОЙ E-ТОЧНОСТЬЮ,

УСТАНОВЛЕНА ТОЧНОСТЬ - 16,

если же будет $LC > 15$, ВАЦ установит точность $LC = 15$ и выдаст сообщение:

<K> ЧИСЛО НЕ ПРЕДСТАВИМО С УКАЗАННОЙ F-ТОЧНОСТЬЮ,

УСТАНОВЛЕНА ТОЧНОСТЬ 15,

где <КС> везде означает порядковый номер обращения к C1FL в программе, соответствующий диагностической печати.

При использовании формата с фиксированной точкой нельзя забывать, что количество значащих цифр при таком представлении числа WC не должно превышать 15. Если это окажется не так, ВАЦ автоматически сменит F-формат представления числа WC на формат с плавающей точкой и запишет число с точностью - 10, причем так, чтобы оно уложилось в отведенное ранее поле. При этом будет выдано сообщение:

<КС> ЧИСЛО НЕ ДОПУСКАЕТ F - ПРЕДСТАВЛЕНИЯ,
ЗАПИСАНО В E-ФОРМЕ, ТОЧНОСТЬ -10,

где <КС> имеет прежний смысл.

Если при обращении к C1FL получится, что часть представления числа WC не попадает в поле листа, ВАЦ выдаст сообщение:

<КС> ЧИСЛО УСЕЧЕНО

и не запишет эту часть числа.

Если же все представление числа WC не попадет в поле листа, ВАЦ не запишет его и выдаст сообщение:

<КС> ЧИСЛО ВНЕ ЛИСТА.

Время одного обращения к C1FL для машины ЕС 1020 оставляет порядка 60 мсек.

Распечатка листа на ALPH. Вывод листа с изображением на печать происходит в режиме при обращении к PRINTL:

CALL PRINTL(KP); (5)

где KP - нужное количество экземпляров листа.

Аргумент в (5) может быть арифметическим выражением.

Обращаясь к PRINTL следует помнить, что после распечатки листа он перестанет быть заданным и сформированное на нем изображение окажется недоступным пользователю.

При попытке распечатать незаданный лист ВАЦ выдаст сообщение:

РАСПЕЧАТЫВАТЬ НЕЧЕГО: НЕ ЗАДАН ЛИСТ.

После этого произойдет выход из системы.

Время распечатки для машин ЕС 1020 оставляет порядка 150 мсек на отроку листа.

2. Пример использования

Для того чтобы показать возможности системы, а также рассмотреть некоторые тонкости работы с ней, разберем, как, используя ВАЦ, построить на АЦПУ график функции $Y = \sin(X)$, рис.2. Сделать это можно с помощью программы на языке ПИ/І ОС, представленной на рис.3. Результатом работы этой программы как раз и является рис.2, он получен на АЦПУ.

Не будем останавливаться на том, как рассчитывались в программе координаты на листе тех или иных фрагментов рис.2. Это требует просто внимательности. Интереснее заметить, что относительная система координат листа выбрана так, чтобы она совпадала с системой координат изображаемой функции. Однако нумерация строк листа идет сверху вниз из отрицательной области значений в положительную (см. рис.1,б). При построении графика это приводит к инвертированию кривой относительно оси абсцисс. Чтобы этого не случилось, в программе значения функции взяты с обратным знаком.

Время работы программы, использующей ВАЦ, зависит не столько от сложности аргументов при обращениях к системе, сколько от количества этих обращений. В этом смысле показательно, как заносятся на лист в программе рис.3 горизонтальные и вертикальные линии. Каждая линия формируется на листе с помощью только одного обращения к TEXTL. Достигается это использованием повторителя строки языка ПИ/І (для проведения горизонтальных линий) или комбинированным использованием повторителя строки и резервированной в ВАЦ литеры " переноса строки с возвратом каретки (для проведения вертикальных линий).

Л и т е р а т у р а

1. Гладких Б.А., Иваненков И.А. Альфа-процедуры для вывода алфавитно-цифровой информации на АЦПУ, В сб.: Стандартные программы и процедуры. Новосибирск, ВЦ СО АН СССР, 1971, вып.12.

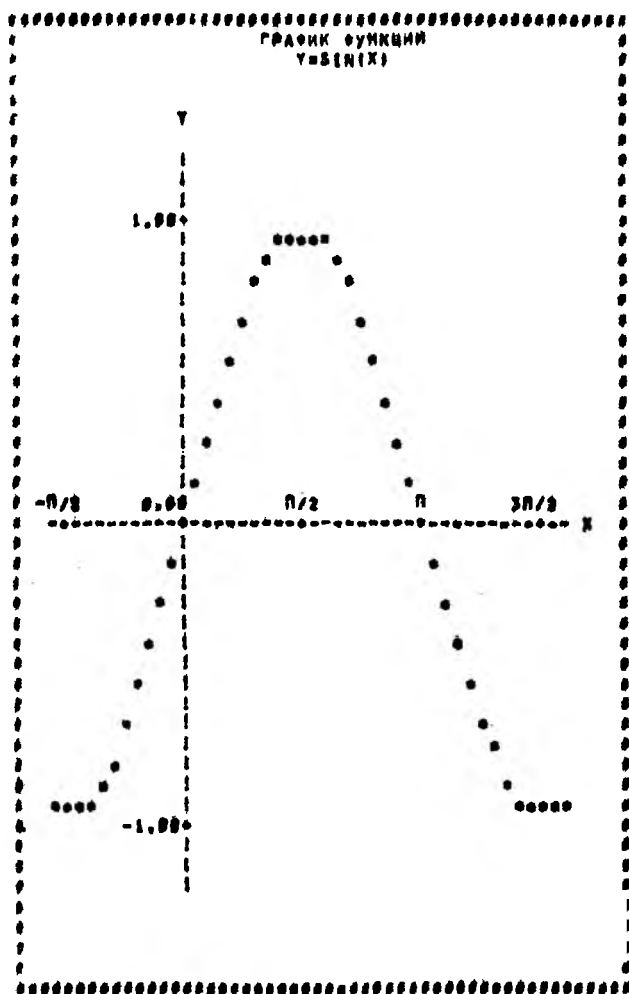


РИС.2. ПРИМЕР ИСПОЛЬЗОВАНИЯ ВАЧ

```

TRAN:PROC OPTIONS(MAIN);
/* ОБ'ЯВЛЕНИЯ АТРИБУТОВ ПАРАМЕТРОВ ТОЧЕК ВХОДА ВАЦ */
DCL LISTL ENTRY(FIXED BIN, FIXED BIN, FIXED BIN),
    CIFL ENTRY(FLOAT DEC(16), FIXED BIN, FIXED BIN, FIXED BIN),
    TEXTL ENTRY(CHAR(1000) VAR, FIXED BIN, FIXED BIN),
    PRINTL ENTRY(FIXED BIN);
/* ОБ'ЯВЛЕНИЕ ВОСПОМОГАТЕЛЬНЫХ ПЕРЕМЕННЫХ */
DCL (I,J) FIXED BIN;
/* ЗАДАНИЕ ЛИСТА ПОД ГРАФИК */
CALL LISTL(72,35,31);
/* ЗАНЕСЕНИЕ НА ЛИСТ РАМКИ */
CALL TEXTL((52)'#',-25,-14);CALL TEXTL((52)'#',23,-14);
CALL TEXTL((47)'#",-24,-14);CALL TEXTL((47)'#",-24,37);
/* ПРОВЕДЕНИЕ ОСЕЙ КООРДИНАТ */
CALL TEXTL((44)'-',0,-11);CALL TEXTL((37)'1",-18,0);
/* НАНЕСЕНИЕ НА ОСИ МАСШТАБНЫХ ДЕЛЕНИЙ */
DO J=-10 TO 30 BY 10;CALL TEXTL('+',0,J);END;
DO I=-15,15;CALL TEXTL('+',I,0);END;
/* ЗАПИСЬ ГРАФИКА ФУНКЦИИ Y=SIN(X) */
DO J=-11 TO 32;
CALL TEXTL('!',ROUND(-15*SIN(0.157*J),0),J);
END;
/* ОЦИФРОВКА И ПОДПИСЫВАНИЕ ОСЕЙ КООРДИНАТ */
CALL TEXTL('Y',-20,0);
DO I=-15,15;CALL CIFL(-I/15,2,I,-3);END;
CALL TEXTL('X',0,34);CALL TEXTL('П/2',-1,-12);
CALL TEXTL('0.00',-1,-3);CALL TEXTL('П/2',-1,9);
CALL TEXTL('П',-1,20);CALL TEXTL('3П/2',-1,28);
/* ПОДПИСЫВАНИЕ ГРАФИКА И РИСУНКА */
CALL TEXTL('ГРАФИК ФУНКЦИИ Y=SIN(X)',-27,7);
CALL TEXTL('РИС.2.ПРИМЕР ИСПОЛЬЗОВАНИЯ ВАЦ',25,-2);
/* РА.ПЕЧАТКА ЛИСТА С ИЗОБРАЖЕНИЕМ */
CALL PRINTL(1);
END TRAN;

```

РИС.3.ПРОГРАММА ДЛЯ ПОСТРОЕНИЯ РИС .2

ПРИЛОЖЕНИЕ 1
РАСПЕЧАТКА ПРОЦЕДУРЫ ВАИ

```

LISTL:PROC(N,I0,J0);
/* LISTL-ЗАДАНИЕ ЛИСТА В N СТРОК И 128 ПОЗИЦИЙ;
(I0,J0)-СТРОКА И ПОЗИЦИЯ ТОЧКИ НАЧАЛА ОТНОСИТЕЛЬНОЙ
СИСТЕМЫ КООРДИНАТ (ОСК) */
/* ПАРАМЕТРЫ ТОЧЕК ВХОДА */
DCL(N,I0,J0,LC,IC,JC,IS,JS,KP)FIXED BIN,
WC FLOAT DEC(16),
STRT CHAR(1000) VAR,
/* ВСПОМОГАТЕЛЬНЫЕ ПЕРЕМЕННЫЕ */
(I,J,J2,L,K,K1) FIXED BIN,
(I1,J1,N1)CTL FIXED BIN,
(KC,KT)CTL FIXED BIN(31,0),
W FLOAT DEC(16),
B CHAR(1),
STR CHAR(1000) VAR,
/* ПРИЗНАКИ ДЛЯ ПРЕДУПРЕДИТЕЛЬНОЙ ПЕЧАТИ */
TB BIT(1) INIT('1'B),/* ТЕКСТ */
TW BIT(1) INIT('1'B),/* ТЕКСТ ВНЕ ЛИСТА */
TY BIT(1) INIT('0'B),/* ТЕКСТ НЕ УСЕЧЕН */
/* ПОЛЕ ЛИСТА */
SP(N) CHAR(128) CTL;
IF N<1 THEN DO;/* ОСТАНОВ С ДИАГНОСТИКОЙ */
PUT SKIP EDIT('ЛИСТ НЕ ЗАДАН: ЧИСЛО СТРОК<1')(A);
PUT SKIP EDIT('ПРОГРАММА ОСТАНОВЛЕНА')(A);
STOP;
END;
ALLOCATE I1,J1,N1,WC,KT,SP;
/* ЗАГРУЗКА ПАРАМЕТРОВ LISTL */
I1=I0;
J1=J0;
N1=N;

```

```

I1=I1+1;
/* ЗАНИМАНИЕ ЛИСТА ПРОБЕЛАМИ */
SP=' ';
/* УСТАНОВКА НАЧАЛЬНОГО ЗНАЧЕНИЯ ДЛЯ СЧЕТЧИКОВ
   ЧИСЛА ОБРАЩЕНИЙ К CIFL-КС И TEXTL-КТ */
KC=0;
KT=0;
RETURN;
CIFL:ENTRY(WC,LC,IC,JC);
/* CIFL-ЗАПИСЬ ЧИСЛА WC НА ЛИСТ;
   LC-ТОЧНОСТЬ ЧИСЛА(ДОЛЖНО БЫТЬ:-16<=-LC<=15),
   ЕСЛИ LC<0,ТО WC ЗАНОСИТСЯ НА ЛИСТ В ФОР-
   МАТЕ E(6-LC,-1-LC,-LC),ЕСЛИ ЖЕ LC>=0,ТО
   В ФОРМАТЕ F(18,LC) (ПРИ LC=0 ДЕСЯТИЧНАЯ
   ТОЧКА ЧИСЛА WC НА ЛИСТ НЕ ЗАПИСЫВАЕТСЯ);
   (IC,JC)-СТРОКА И ПОЗИЦИЯ ДЕСЯТИЧНОЙ ТОЧКИ ЧИСЛА В ОСК */
IF -ALLOCATION(SP) THEN DO; /* ОСТАНОВ С ДИАГНОСТИКОЙ */
  PUT SKIP EDIT('ВНИМАНИЕ! НЕ ЗАДАН ЛИСТ')(A);
  PUT SKIP EDIT('ПРОГРАММА ОСТАНОВЛЕНА')(A);
  STOP;
                                END;
/* ЗАПИСКА ПАРАМЕТРОВ CIFL */
W=WC;
L=LC;
I=IC;
J=JC;
/* УВЕЛИЧЕНИЕ НА СЧЕТЧИКА
   ЧИСЛА ОБРАЩЕНИЙ К CIFL */
KC=KC+1;
/* ВЫРАВНИВАНИЕ ТОЧНОСТИ L */
IF L<-16 THEN DO; /* УСТАНОВКА МАКСИМАЛЬНО ДОПУСТИМОЙ ТОЧНОСТИ
   ДЛЯ E-ПРЕДСТАВЛЕНИЯ ЧИСЛА WC И ДИАГНОСТИКА */
  L=-16;
  STR=' ЧИСЛО НЕ ПРЕДСТАВИМО С УКАЗАННОЙ E-ТОЧНОСТЬЮ,';

```

```

STR=STR!!'УСТАНОВЛЕНА ТОЧНОСТЬ -16';
PUT SKIP EDIT(KC,STR)(F(6),A);
END;
IF L>15 THEN DO;/* УСТАНОВКА МАКСИМАЛЬНО ДОПУСТИМОЙ ТОЧНОСТИ
ДЛЯ F-ПРЕДСТАВЛЕНИЯ ЧИСЛА WC И ДИАГНОСТИКА*/
L=15;
STR=' ЧИСЛО НЕ ПРЕДСТАВИМО С УКАЗАННОЙ F-ТОЧНОСТЬЮ,';
STR=STR!!'УСТАНОВЛЕНА ТОЧНОСТЬ 15';
PUT SKIP EDIT(KC,STR)(F(6),A);
END;
/* ЗАЩИТА F-ПРЕДСТАВЛЕНИЯ ЧИСЛА WC(КОЛИЧЕСТВО ЦИФР ПЕРЕД ДЕСЯ-
ТИЧНОЙ ТОЧКОЙ ПРИ ТАКОМ ПРЕДСТАВЛЕНИИ ДОЛЖНО БЫТЬ =-(15-L))*/
IF L<=0 THEN
IF W>=1E1**(15-L) THEN DO;/* ФОРМИРОВАНИЕ ПОЗИЦИИ ДЕСЯТИЧ-
НОЙ ТОЧКИ И ЗАДАНИЕ ТОЧНОСТИ
-10 ДЛЯ ЧИСЛА WC, ТАК ЧТОБЫ ОНО
ИМЕЛО E-ПРЕДСТАВЛЕНИЕ И УЛОЖИ-
ЛОСЬ В ОТВЕДЕННОЕ РАНЕЕ ПОЛЕ */
IF L<0 THEN J=J-14; ELSE J=J+L-13;
L=-10;
/* ДИАГНОСТИКА */
STR=' ЧИСЛО НЕ ДОПУСКАЕТ F-ПРЕДСТАВЛЕНИЯ,';
STR=STR!!'ЗАПИСАНО В E-ФОРМЕ, ТОЧНОСТЬ -10';
PUT SKIP EDIT(KC,STR)(F(6),A);
END;
/* ФОРМИРОВАНИЕ СТРОКОВОГО ПРЕДСТАВЛЕНИЯ STR ЧИСЛА WC И
НАЧАЛЬНОЙ ПОЗИЦИИ J СТРОКИ STR В ЗАВИСИМОСТИ ОТ ТОЧНОСТИ L */
IF L<0 THEN DO;/* СТРОЧНОЕ ПРЕДСТАВЛЕНИЕ ЧИСЛА WC
В ВИДЕ С ПЛАВАЮЩЕЙ ТОЧКОЙ */
PUT STRING(STR) EDIT(W)(E(6-L,-1-L,-L));
J=J-2;
END;
ELSE DO;/* СТРОЧНОЕ ПРЕДСТАВЛЕНИЕ ЧИСЛА WC
В ВИДЕ С ФИКСИРОВАННОЙ ТОЧКОЙ */

```

```

        PUT STRING(STR) EDIT(W)(F(18,L));
        IF L=0 THEN J=J-18;ELSE J=J+L-17;
        END;
/* ПРИЗНАК ДЛЯ ПРЕДУПРЕДИТЕЛЬНОЙ ПЕЧАТИ */
T='0'B; /* НЕ ТЕКСТ */
/* ПЕРЕДАЧА СТРОЧНОГО ПРЕДСТАВЛЕНИЯ STR ЧИСЛА WS И КООРДИНАТ
(I,J) ПЕРВОЙ ЛИТЕРЫ СТРОКИ STR КАК АРГУМЕНТОВ ДЛЯ TEXTL */
GO TO M1;
TEXTL: ENTRY(STRT,IS,JS);
/*TEXTL-ЗАПИСЬ СТРОКИ STRT НА ЛИСТ;
(IS,JS)-СТРОКА И ПОЗИЦИЯ ПЕРВОЙ ЛИТЕРЫ ИЗ STRT В ОСК */
IF -ALLOCATION(SP) THEN DO; /* ОСТАНОВ С ДИАГНОСТИКОЙ */
    PUT SKIP EDIT('ВНИМАНИЕ! НЕ ЗАДАН ЛИСТ')(A);
    PUT SKIP EDIT('ПРОГРАММА ОСТАНОВЛЕНА')(A);
    STOP;
                                END;
/* ЗАЩИТА ПАРАМЕТРОВ TEXTL */
STR=STRT;
I=IS;
J=JS;
/* УВЕЛИЧЕНИЕ НА 1 СЧЕТЧИКА
    ЧИСЛА ОБРАЩЕНИЙ К TEXTL */
KT=KT+1;
/* ПЕРЕХОД К АБСОЛЮТНОЙ СИСТЕМЕ КООРДИНАТ */
M1: I=I+1;
    J=J+J;
/* ЗАПОМИНАНИЕ ПОЗИЦИИ ПЕРВОЙ ЛИТЕРЫ СТРОКИ STR */
J2=J;
K1=LENGTH(STR);
/* ПРОВЕРКА НА ПОПАДАНИЕ СТРОКИ STR В ПОЛЕ ЛИСТА */
IF I>F1!J>127!(J+K1)<1 THEN DO; /* ВЫХОД С ДИАГНОСТИКОЙ */
    IF T THEN PUT SKIP EDIT(KT,' ТЕКСТ ВНЕ ЛИСТА')(F(6),A);
    ELSE PUT SKIP EDIT(KC,' ЧИСЛО ВНЕ ЛИСТА')(F(6),A);
    RETURN;

```



```

END;

/* ЗАНЕСЕНИЕ СТРОКИ STR ПОЛИТЕРНО В ПОЛЕ ЛИСТА */
DO K=1 TO K1;
J=J+1;
/* ВЫРЕЗАНИЕ K-ТОЙ ЛИТЕРЫ СТРОКИ STR */
B=SUBSTR(STR,K,1);
IF B=' ' THEN GO TO M2;
/* ПРОВЕРКА В ПОЛЕ НЕ ЗАНОСИТСЯ */
IF B='.' THEN DO; /* ЛИТЕРА "." В ПОЛЕ НЕ ЗАНОСИТСЯ И ОЗНАЧАЕТ
                    ПЕРЕХОД НА СЛЕДУЮЩУЮ СТРОКУ (I+1) ЛИСТА В
                    НАЧАЛЬНУЮ ПОЗИЦИЮ J2 СТРОКИ STR */
I=I+1;
J=J2;
GO TO M2;
END;

/* ПРОВЕРКА НА ПОПАДАНИЕ ЛИТЕРЫ ИЗ В В ПОЛЕ ЛИСТА */
IF I<111I>N11J<11J>128 THEN DO; /* ЛИТЕРА В ПОЛЕ НЕ ЗАНОСИТСЯ */
/* ПРИЗНАК ДЛЯ ПРЕДУПРЕДИТЕЛЬНОЙ ПЕЧАТИ */
TY='.'B; /* ТЕКСТ УСЕЧЕН */
GO TO M2;
END;

/* ЗАПИСЬ ЛИТЕРЫ В ПОЛЕ */
SUBSTR(SP(I),J,1)=B;
/* ПРИЗНАК ДЛЯ ПРЕДУПРЕДИТЕЛЬНОЙ ПЕЧАТИ */
TB='0'B; /* ТЕКСТ НЕ ВНЕ ЛИСТА */
M2: END;

/* ПРЕДУПРЕДИТЕЛЬНАЯ ПЕЧАТЬ */
IF TB THEN DO;
IF T THEN PUT SKIP EDIT(КТ, ' ТЕКСТ ВНЕ ЛИСТА')(F(6),A);
ELSE PUT SKIP EDIT(КС, ' ЧИСЛО ВНЕ ЛИСТА')(F(6),A);
END;
ELSE IF TY THEN DO;
IF T THEN PUT SKIP EDIT(КТ, ' ТЕКСТ УСЕЧЕН')(F(6),A);
ELSE PUT SKIP EDIT(КС, ' ЧИСЛО УСЕЧЕНО')(F(6),A);

```

```

END;

RETURN;
PRINTL:ENTRY(KP);
/* PRINTL-РАСПЕЧАТКА ЛИСТА В KP ЭКЗЕМПЛЯРАХ */
IF ~ALLOCATION(SP) THEN DO; /* ВЫХОД С ДИАГНОСТИКОЙ */
  PUT SKIP EDIT('РАСПЕЧАТЫВАТЬ НЕЧЕГО: НЕ ЗАДАН ЛИСТ')(A);
  RETURN;

END;

DO K=1 TO KP;
OPEN FILE(SYSPRINT) PAGESIZE(N1) LINESIZE(128);
PUT FILE(SYSPRINT) EDIT(SP)(A);
CLOSE FILE(SYSPRINT);
END;
FREE I1,J1,N1,NC,KT,SP;
END LISTL;

```

ПРИЛОЖЕНИЕ 2

СПИСОК ЛИТЕРАТУРЫ ДЛЯ АЦПУ ЕС 7032 И ПЕРФОРАТОРА ЕС 9011

```

0 1 2 3 4 5 6 7 8 9
В Г Д Ж З И Й Л П У Ф Ц Ч Ш Щ Ъ Э Ю Я
А В С Д Е Ф Г Н И К Л М О Р Q R S T U V W X Y Z
. : , ; " ' ! ? % _ * @ [ \ ] ^ & ~ + - = < > ( ) [ ]

```

ОБЗОР СИСТЕМ ГРАФИЧЕСКОГО ВЫВОДА

Ю.Л.Костюк

I. Введение

Машинная графика возникла совсем недавно, но уже можно выделить в ней три направления, соответствующие трем сферам ее применения. Это автоматизация проектирования, автоматизация обработки геодезических, картографических и геологических данных, автоматизация научных исследований. Для каждого из направлений характерен определенный набор задач. Так, в автоматизации проектирования решаются задачи ввода чертежей в ЭВМ, организации графического диалога с ЭВМ (например, с помощью дисплея), а также манипулирование с трехмерными объектами и их проецирование, рисование готовых чертежей (на графопостроителе). При обработке геодезических данных возникают задачи построения карт изолиний, штриховки контуров. Наконец, в автоматизации научных исследований существуют задачи изображения графиков, гистограмм, сглаживания кривых.

К настоящему времени создано большое количество графических систем, ориентированных на различные сферы применения, разнообразные ЭВМ, графопостроители и дисплеи. Тем не менее можно выделить ряд функций графического вывода, которые обязательны для любой системы (они являются средствами построения простейших изображений). К этим графическим функциям относятся следующие: 1) раскрой бумаги на отдельные рисунки (листы), 2) координатные преобразования изображений на плоскости, 3) изображение надписей и чисел, 4) изображение линий (прямых и кривых, сплошных и штриховых и т.п.), 5) рисование стандартных графических объектов.

Если в базе графической системы эти функции реализованы достаточно универсальным образом, то на его основе можно создавать специализированные проблемно-ориентированные надстройки для любых областей применения машинной графики.

В обзоре рассматриваются системы, ориентированные на вывод результатов инженерных и научных расчетов в графической форме. Кроме перечисленных возможностей базиса такие системы должны обладать еще, как минимум, средствами изображения графиков. Графические функции большинства рассматриваемых систем ограничиваются пределами базиса и вывода графиков, поэтому подробно мы остановимся на реализации именно этих возможностей.

Обзор составлен по "двухкоординатному" принципу: в п.2 перечислены рассматриваемые системы и кратко перечислены их графические возможности в целом, а в п.3 рассматриваются отдельные графические функции и описывается их реализация в различных системах.

2. Краткая характеристика систем

Будут рассмотрены следующие системы.

1) Система ГРАФОР [2-5], реализованная в виде набора подпрограмм Фортрана для ЭВМ БЭСМ-6, SDS-910, ЕС ЭВМ.

2) Комплекс программ ВЦ СО АН СССР [8-11,15,16], реализованный в виде набора подпрограмм Алгола (транслятор Альфа для М-220 и Альгибр для БЭСМ-6), Фортрана и автокода (БЭСМ-6).

3) Набор подпрограмм языка Фортран для графопостроителя ДИГИГРАФ (ЕС-7054), реализованный в ДОС ЕС ЭВМ [17].

4) Набор подпрограмм для графопостроителя АТЛАС [1,12,18-20], реализованный для ЭВМ типа М-20, Минск-22, Минск-32, ЕС ЭВМ. Вызов из машинных языков или Фортрана, ПЛ/I.

5) Подпрограммы вывода на графопостроитель ДРП-3 [21] из ЭВМ типа М-20. Вызов из Алгол-60 (транслятор ТА-1М).

6) Система ВЦ АН СССР [23,24], реализованная в Алголе-60 ЭВМ БЭСМ-6.

7) Программы фирмы *Calcomp* [22,26]. Вызов из Фортрана.

8) Программы фирмы *Venson* [25,30]. Вызов также из Фортрана.

9) Система *NPL* [27,31] . Вызов из Алгол-60 или автокода.

10) Система *AMESPLOT* [29] , реализованная в виде набора подпрограмм Фортрана.

11) Система *EASYPLOT* [28] . Вызов из Фортрана.

12) Система *CMOT*, реализация для ЭВМ типа М-20 [6] . Вывод на графопостроитель ДРП-3, обращение из Алгола-60 (трансляторы Альфа и ТА-1М).

13) Система *CMOT*, реализация для ЕС ЭВМ [14] . Обращение из языков Фортран, ПЛ/I.

Эти системы включают программы проведения отрезков и ломаных линий, позволяют подвергать линии и тексты координатным преобразованиям. В системах (1), (13) возможны вложенные координатные преобразования. В системах (1), (2), (9)-(11) предусмотрен раскрой бумаги на листы с произвольными размерами, а в системах (12), (13) - на листы стандартных (по ГОСТу) размеров (форматы). В системах (1), (4), (5), (9) - (13) можно изобразить произвольную гладкую кривую по узловым точкам, а в системах (2), (4), (6), (9) - кривую, заданную в функциональном виде (в виде подпрограммы). В системах (3)-(5), (8)-(10), (12), (13) линии можно изображать в виде штриховых и штрихпунктирных. В системе (3) предусмотрены специальные операторы изображения таких фигур, как окружности и их дуги, в системе (8) к ним добавлены эллипсы, в системе (9), кроме того, прямоугольники, параболы, гиперболы, а в системе (1) - окружности, спирали, эллипсы, прямоугольники, многоугольники, сетки. В системы (12), (13) входит оператор изображения фрагмента-рисунка, описанного на специальном графическом языке, и который может подвергаться различным координатным преобразованиям. Эта возможность сближает системы (12), (13) с системами, имеющими входной графический язык (например, [7,13]).

Минимальные возможности изображения графиков, которыми обладают все перечисленные системы, заключаются в изображении координатных осей и собственно графиков. В системах (1), (6), (9)-(13) возможна автоматическая разметка осей. В системах (1), (10), (11) возможен логарифмический масштаб по осям графика, а в системах (6), (12), (13) возможен произвольный функциональный масштаб (задается подпрограммой). В системах (1), (10) предусмотрены графики в полярной системе координат.

В дополнение к указанным возможностям в системах (10), (13) можно задавать бланки - прямоугольные области в поле чертежа,

где в дальнейшем будет производиться блокировка изображения линий. В системе (I) эти области могут описываться произвольной замкнутой ломаной линией, кроме того, в этой системе имеются программы штриховки контуров, можно запоминать в памяти ЭВМ след пера (для его последующей обработки).

Часть систем обладает и другими надотрочными возможностями: изображение изолиний – системы (2), (4), (9), изображение пространственных поверхностей – системы (I), (2), (6), векторных полей (2), изображение пространственных кривых и манипулирование с трехмерными графическими объектами (I), сглаживание функций (I), (I2), изображение номограмм (6), гистограмм (I), (IO), (I3).

Рассматриваемые графические системы реализованы как наборы подпрограмм, к которым пользователь может обращаться из какого-либо языка программирования. Такой способ реализации типичен для систем, ориентированных на вывод результатов вычислений на графопостроитель (вычисления ведутся на основном языке программирования). Существует и другой способ – реализация графической системы, как компилятора со специального графического языка. Этот способ чаще применяется в системах, ориентированных на автоматизацию проектирования (например, языки ОГРА-I [I3], ГРАФИК [7]), когда описание чертежа на таком языке получается более компактным, чем на Фортране или Алголе-60. Однако из-за большой трудоемкости в реализации этот способ получил меньшее распространение (впрочем, для вывода результатов вычислений первый способ более удобен).

3. Реализация графических функций базиса

Выше мы выделяли базисные графические функции: раскрой бумаги, координатные преобразования, изображение линий, надписей, стандартных графических объектов.

Раскрой бумаги. Эта функция необходима в системе по нескольким причинам. Как правило, размер физического листа бумаги графопостроителя слишком велик для отдельного рисунка и даже для нескольких рисунков, изображаемых в одной программе. Для программиста, использующего систему, удобнее всего размещать изображения внутри листа, абстрагируясь от его положения на бумаге графопостроителя: тогда его программа будет пригодна для

графопостроителей с различными размерами физического листа бумаги и при произвольном положении листа на физическом листе бумаги.

При этом система должна автоматически выделять незанятую часть бумаги под очередной лист, осуществлять пересчет координат рисунка к физическому листу, а также производить защиту формата — блокировать выход пера за лист, чтобы не испортить соседние рисунки.

В системах (I), (2), (9)–(II) раскрой осуществляется на листы произвольного размера, задаваемого программистом, а в системах (I2), (I3) — стандартного размера по ГОСТу (форматы I1, I2 и т.д.). Второй способ удобнее как для программиста (не нужно придумывать размеры листа), так и для системы (проще организовать экономный раскрой бумаги). Впрочем, в системах (I2), (I3) можно использовать и физический лист бумаги.

Координатные преобразования. Они необходимы для того, чтобы было удобнее использовать программы изображения отдельных графических элементов и для облегчения создания проблемно-ориентированных надстроек. Для первой цели достаточно частичное аффинное преобразование: сдвиг начала координат, поворот координат относительно начала на некоторый угол, одинаковое по обеим осям увеличение (уменьшение) масштаба. В общем же случае необходимо полное аффинное преобразование (например, при построении проекций трехмерных фигур): в дополнение к перечисленным преобразованиям изменение угла раскрытия осей координат и отдельно по каждой оси изменение масштаба. Кроме того, для возможности применения координатных преобразований в подпрограммах, необходима вложенность систем координат.

В большинстве графических систем координаты реализованы таким образом, что если задано некоторое преобразование, то все изображения линий рисуются только в новой системе координат. В системах (2) и (8) можно рисовать в новой (аффинной или декартовой) системе координат и в абсолютной системе координат, используя для этого различные операторы рисования. В системе (I2) внутри листа определены декартовы координаты, а внутри них — аффинные. В системе (I) возможна многократная вложенность аффинных координат, а в (I3) аффинные координаты могут быть параллельными и вложенными (т.е. можно создавать набор систем координат в виде дерева).

Линии. Основными элементарными графическими объектами являются линии: любое изображение, нарисованное графопостроителем, состоит в конечном счете из линий.

В большинстве систем простейшими линиями, о которых имеет дело программист, являются отрезки прямых и холостые движения пера (программист, как правило, должен предусматривать подвод пера к начальной точке изображения очередного графического элемента). В системах (12), (13) перо подводится в нужную точку рисунка автоматически.

В общем случае линия задается набором узловых точек, которые могут соединяться ломаной или гладкой кривой. Множество кривых, проходящих через набор узловых точек, неограничено, в различных системах применяются разные методы. В системах (4), (5) для этого используются интерполяционные кубические полиномы Лагранжа, а в (1) — классические сплайны. Эти методы не универсальны (абсциссы узловых точек должны быть упорядочены), поэтому применяют и другие методы: в системе (1) еще и полные кубические сплайны в параметрической форме, а в системе (12) — простейший вариант локального параметрического сплайна. В системе (13) применены 4 вида локальных параметрических сплайнов, предназначенных для изображения разных кривых, которые исчерпывают всевозможные случаи. В системах (2), (6) кривые задаются в функциональном виде, т.е. проблема гладкой интерполяции сводится к системе и перекладывается на программиста (который должен написать соответствующую программу, вычисляющую кривую).

В некоторых системах штриховые линии можно получить только при изображении отдельных фигур (например, в (1)). В системе (3) предусмотрен один вид штриховой и один вид штрихпунктирной линии в соответствии с аппаратными возможностями графопостроителя ЕС-7054. Часто этого недостаточно, поэтому в системах (12), (13) определено несколько видов штриховых и штрихпунктирных линий в соответствии с ГОСТом, а в системах (5), (8) — (10), (13) можно задавать произвольную разрывность линий в виде длин штрихов и пробелов.

При изображении графиков иногда требуется помечать узловые точки маркерами. В большинстве систем такая возможность предусмотрена. В системах (12), (13) около узловых точек линия не проводится и маркер получается более наглядным.

Для возможности использования различных пишущих элементов

во многих системах также предусмотрена возможность смены пера.

В большинстве систем каждая из перечисленных возможностей в изображении линий реализуется одной или несколькими подпрограммами (или дополнительными параметрами в программах рисования). В системах (I2), (I3) осуществлен другой способ: набор характеристик, называемый типом линии, задается специальной подпрограммой, после чего может быть задано необходимое число узловых точек (другой подпрограммой). Накопленная таким образом информация о линии (или совокупности линий) изображается далее в необходимой системе координат (возможно, неоднократно).

Символы. Наряду с линиями символы являются простейшими графическими объектами (хотя, конечно, символ состоит из линий). Набор символов у некоторых систем совпадает с набором символов в АЦПУ — (3), (4), (6)–(8); в такой набор входит минимальный набор специальных символов, цифр, заглавных латинских и (возможно) русских букв (до 60–90 символов). В других системах имеются и греческие буквы, а также строчные буквы и дополнительные математические и др. специальные символы. В системах (I), (2) число символов до 200, а в (I3) — 250. При таком большом числе символов важное значение приобретает экономичность их внутренней кодировки.

Каждый символ обычно представляется на условной целочисленной решетке. Величина основных целочисленных символов на такой решетке составляет 2x3 клетки (6), 4x6 клеток (I3), 4x7 или 14x14 клеток (4). В большинстве систем применяют размер 4x7 (хотя размер 4x6 ввиду большей симметрии оставляет лучшее эстетическое впечатление). При таких размерах решетки 200–250 символов можно закодировать в 2–3 Кбайтах памяти.

Фактически размеры символов должны быть различны, их высота задается либо специальной подпрограммой, либо параметром в подпрограмме рисования символов. Во многих системах можно задавать наклонный шрифт (как того требует ГОСТ), а в системах (I2), (I3) — широкий или узкий шрифт.

В большинстве систем в подпрограммах изображения символов и текстов специальным параметром задается угол поворота отрезка символов относительно горизонтали. В системах (I), (I2), (I3) символы считаются вложенными в систему координат. В системах (9), (I2), (I3) имеются средства размещения символов по отрезкам и позициям относительно заданной точки рисунка. Для этого в

системе (9) введено понятие текущей строки и позиции символа, а в (12), (13) — понятие трафарета.

Символы задаются либо целочисленными номерами, либо символическими строками — способ задания зависит от возможностей языка программирования. Так как набор символов обычно превышает набор символов АЦПУ, то в системах (1), (9) весь набор разделен на поднаборы и имеется оператор переключения поднабора. В системах (12), (13) для этих целей заглавные буквы алфавита кодируются совокупностью надчерк — буква, а ряд специальных символов кодируется в виде надчерк — номер символа. В системе (13), кроме того, введено понятие регистра (русский, латинский, греческий).

В системах (1), (3) некоторые специальные символы (маркеры) изображаются отдельным оператором.

Кроме того, в большинстве систем предусмотрены программы вывода чисел в различных форматах (плавающим, фиксированном).

Стандартные графические объекты. Средства изображения более сложных, чем линии и символы, графических объектов необходимы при создании специализированных надстроек.

В системе (3) имеются программы изображения окружностей и их дуг (с использованием аппаратных возможностей графопостроителя), их нельзя подвергать координатному преобразованию. В других системах применяются более гибкие программные средства изображения окружностей, эллипсов и их дуг — (1), (8) — (10), а также спиралей, многоугольников и сеток — (1), прямоугольников — (1), (9), (10), парабола и гипербола — (9).

Такого набора фигур недостаточно для конкретных приложений, поэтому важное значение имеют средства системы, с помощью которых пользователь сам может расширять набор таких графических объектов. Используя в системе (1) вложенные координаты, пользователь может наращивать набор объектов в виде подпрограмм, их изображающих (в других системах это более трудоемко — в таких подпрограммах нельзя применять преобразования координат). Однако такой способ громоздок. В системах (12), (13) для описания фигур предусмотрен специальный графический язык, описание на нем более компактно. Готовые описания фигур можно хранить на внешних носителях (магнитофонах, дисках) для многократного использования. В системе (12) язык очень прост — фигура описывается совокупностью линий. В системе (13) при описании фигур можно по-

пользовать другие готовые графические объекты, подвергая их координатному преобразованию. В обоих случаях готовые фигуры можно видоизменять с помощью систем координат.

Графики. Минимальными возможностями для изображения графиков обладают все рассматриваемые системы. Программы изображения осей требуют задания информации о начальных и конечных значениях, шаге разбиения, длине осей. График изображается программами рисования линий.

Системы (2), (7), (8) производят автоматическое масштабирование графиков, системы (1), (6), (9)–(13) автоматически выбирают шаг разбиения оси.

В системах (1), (10), (11) возможен не только равномерный, но и логарифмический масштаб, а в системах (6), (12), (13) – произвольный функциональный масштаб, задаваемый программистом в виде подпрограммы. В системе (6) программист должен задать пределы изменения аргумента и функции, а в системах (12), (13) они находятся и округляются полностью автоматически, обеспечивая наиболее "красивое" разбиение сетки графика и его наиболее рациональное размещение.

4. Сравнение графических систем

В таблице возможности систем ограничены средствами описания и изображением графиков. Наличие некоторой функции обозначено следующим образом: + – реализована в виде частного случая; ++ – реализована в целом, но с некоторыми ограничениями; +++ – реализована без ограничений.

Удобство применения системы (от чего зависит трудоемкость ее использования) зависит от многих факторов – наличие ограничений в различных функциях, опосредованная способность системы строить изображения по минимуму исходной информации, общее количество подпрограмм в системе и их сложность и т.п. В первой строке таблицы приведено число подпрограмм в системе, реализующих рассматриваемые функции. Оно очень сильно различается в разных системах. В тех системах, где их немного, каждая подпрограмма содержит очень много параметров, поэтому применять такие системы трудоемко. С другой стороны, в системах с большим числом подпрограмм каждая из них более проста. Наилучшим вариантом, по-видимому, следует

Таблица I

Сравнение графических систем

Графические функции	1. ГРАФОР	2. Куртуков	3. ЛИТИГРАФ	4. АТЛАС	5. ДРП-3	6. Числов	7. Calcomp	8. Benson	9. NPL	10. AMESPLOT	11. EASYPLOT	12. CROG M-20	13. CROG EC
Кол-во операторов базиса и изображения графиков	54	18	27	12	5	17	7	40	77	71	5	22	27 (15)
Раскрой бумаги	++	++							++	++	++	++	++
Аффинные координаты	+++	++	++	++			+++	++	++	++		++	++
Блочные координаты	++											+	++
Прямые линии	+++	+++	+++	+++	+++	+++	+++	+++	+++	+++	+++	+++	+++
Кривые линии	++	+		+	+	+			++	++	++	++	++
Штриховые и штрих-пунктирные линии	+	+	++	+	+++			+++	+++	+++		++	+++
Маркировка узлов	++	+	++	+			++	++	++	++	++	+++	+++
Полнота набора символов	+++	+++	++	++	++	+	+	+	++	++	+	+++	+++
Варианты шрифта по ГОСТу	++	++	+	+	++	+	+	+	++	++	+	+++	+++
Координатные преобразования символов	+++	+++	++	+++	+++	++	++	++	++	++		+++	+++
Управление размещением текстов									+++	+++		+++	+++
Числа	+++	+++	++	++		++	++	++	+++	+++		+++	+++
Стандартные объекты	++		+					+	+	+		+++	+++
Графики с равномерным масштабом	+++	++	++	++	++	++	++	++	+++	+++	+++	+++	+++
Графики с логарифмическим масштабом	+++					++		++	+++	+++	+++	+++	+++
Графики с произвольным функциональным масштабом						++						+++	+++

считать небольшое число простых подпрограмм, но для этого необходимо, чтобы в системе были лишь наиболее универсальные средства с минимумом ограничений.

В системе (I3) для облегчения ее использования применены следующие меры: подпрограммы разделены на основные (наиболее простые), которые достаточны в большинстве случаев и реализуют наиболее стандартные варианты (их всего 15) и дополнительные (посложнее), реализующие максимум возможностей.

Л и т е р а т у р а

1. Айпанов Ш.А., Жиленкова Н.М., Перфильев Л.Г. Подпрограммы вычерчивания осей координат и графиков на ЭВМ Минок-32 и ЕС ЭВМ с помощью устройства АТЛАС. Инструктивные указания, сер.ХІ, машинная графика. Вып.І4. КОМЭ Мин-геологии Каз.ССР, Алма-Ата, 1975.
2. Баяковский Ю.М., Михайлова Т.Н., Мишакова С.Т. ГРАФОР: комплекс графических программ на ФОРТРАНе. Вып.І. Основные элементы и графики. М., препринт ИИМ АН СССР, № 4І, 1972.
3. Баяковский Ю.М., Гринберг Г.С., Зиман Ю.Л. ГРАФОР: комплекс графических программ на ФОРТРАНе. Вып.2. М., препринт ИИМ АН СССР, № 52, 1973.
4. Баяковский Ю.М., Лебедева Т.С., Мамаева А.И. ГРАФОР: комплекс графических программ на ФОРТРАНе. Вып.3. М., препринт ИИМ АН СССР, № 88, 1974.
5. Баяковский Ю.М., Топалов Н.Н. ГРАФОР: комплекс графических программ на ФОРТРАНе. Вып.4. М., препринт ИИМ АН СССР, № 79, 1974.
6. Гладких Б.А., Костюк Ю.Л., Позолотин В.А. СМОГ - система математического обеспечения графопостроителя. Томск, Изд-во Томского университета, 1974.
7. ГРАФИК - система математического обеспечения ЭВМ и графопостроителей. Под ред. Е.Л.Оуенко. Кишинев, "Штиинца", 1975.
8. Дворжац В.И. Процедуры вычерчивания изолиний.-В об.:Машинная графика и ее применение. Новосибирск, 1973.

9. Дворжец В.И. Комплект процедур вывода графиков. -В сб.:Машинная графика и ее применение. Новосибирск, 1973.
10. Дворжец В.И., Куртуков А.Я. Система для М-220. -В сб.:Машинная графика и ее применение. Новосибирск, 1973.
11. Дебелов В.А. Процедуры изображения поверхностей. -В сб.:Машинная графика и ее применение. Новосибирск, 1973.
12. Захарченко В.В., Перфильев Л.Г. Стандартные программы вычерчивания прямой линии и надписей на устройстве "Атлас" для ЭВМ БЭСМ-4. Инструктивные указания, сер.ХІ, машинная графика. Вып.І. КОМЭ Мингеологии Каз.ССР, Алма-Ата, 1971.
13. Зозулевич Д.М. Машинная графика в автоматизированном проектировании. М., "Машиностроение", 1976.
14. Костюк Ю.Л. Система математического обеспечения графического вывода для ЕС ЭВМ. Томск, Изд-во Томского университета, 1977.
15. Куртуков А.Я., Горин С.В., Дворжец В.И., Дебелов В.А. Система для БЭСМ-6. -В сб.:Машинная графика и ее применение. Новосибирск, 1973.
16. Математическое обеспечение для графопостроителей. I уровень. Под ред. А.Я.Куртукова. ВЦ СО АН СССР, Новосибирск, 1971.
17. Основная система математического обеспечения графопостроителя ДИГИГРАФ. Прага, 1974.
18. Перфильев Л.Г. Модернизированная программа вычерчивания карт изолиний на графопостроителе "Атлас" для ЭВМ БЭСМ-4, М-222. Инструктивные указания, сер.ХІ, машинная графика. Вып.9. КОМЭ Мингеологии Каз. ССР, Алма-Ата, 1974.
19. Перфильев Л.Г., Трофимов А.К., Шабалдин В.Н., Шипицына Л.М. Подключение графопостроителя "Атлас-3" к ЭВМ ЕС-1020 и программное обеспечение нижнего уровня. Инструктивные указания, сер.ХІ, машинная графика. Вып.12. КОМЭ Мингеологии Каз. ССР, Алма-Ата, 1975.
20. Перфильев Л.Г., Шабалдин В.Н., Шипицына Л.М. Программа "SYMBOL" для вычерчивания надписей с помощью графопостроителя "Атлас-3", подключенного к ЭВМ ЕС-1020. Инструктивные указания, сер.ХІ, машинная графика.

Вып. 13. КОМЭ Мингеологии Каз. ССР, Алма-Ата, 1975.

21. Программы вывода информации на ДРП-3.
22. Фараджев И.А. Математическое обеспечение графопостроителя. Математическое описание системы 4. Вып. 14. М., Институт проблем управления, 1972.
23. Чибиков В.В. Вычерчивание и оформление номограмм на графопостроителе, подключенном к машине БЭСМ-6. "Номографический сборник", № 9. М., ВЦ АН СССР, 1973.
24. Чибиков В.В. Автоматизация построения номограмм и графиков устройством АППУ и графопостроителем. - В сб.: Проблемы повышения эффективности БЭСМ-6. Вильнюс, 1974.
25. Benson Basic Program. Level I. Ref: 4/400/02/022/00, 1973.
26. Calcomp Software Reference Manual. California Computer Products, 1968.
27. Heap B.R., Laws M. Construction of the Characters Available in the NPL Graphical Output System. National Physical Laboratory Central Computer Unit Report CCU-1, 1968.
28. Hedrick G.B., Forrest I.R. A High Level Plotting System. "Software-Practice and Experience", v. 3, N4, 1973.
29. Hirschsohn I. AMESPLOT. A High Level Data Plotting Software System. "Comm. ACM", v. 13, N9, 1970.
30. Library of Benson. Drafting Subroutines. Level II. Ref: 4/400/02/023/00, 1973.
31. Nott C.M. A Brief Guide to the NPL Graphical Output System. National Physical Laboratory Central Computer Unit Report CCU-2, 1968.

СИСТЕМА ГРАФИЧЕСКОГО ВЫВОДА СМОГ И ЕЁ РЕАЛИЗАЦИЯ НА ЕС ЭВМ

Б.А.Гладких, Ю.Л.Костюк

I. Введение

От системы машинной графики требуются различные способности при решении различных задач автоматизации проектирования и вывода результатов вычислений в графической форме. Поэтому такая система должна содержать универсальный базис и ряд проблемно-ориентированных надстроек. В базис входят графические функции, необходимые при выводе любых изображений, а каждая из надстроек рассчитана на построение изображений определенного класса с минимальными затратами на программирование.

Система СМОГ построена именно по такому принципу. Она предназначена в первую очередь на вывод результатов вычислений в графической форме, однако на её основе могут создаваться любые надстройки. Первая реализация СМОГ [1], рассчитанная на ЭВМ типа М-20, содержит базис и надстройку для изображения графиков. В основу реализации для ЕС ЭВМ [2] положены более обобщенные понятия, благодаря чему возможности базиса системы стали более универсальными. Чтобы система не была громоздкой, её структура выбрана с учетом принципа "ортогональности", а для облегчения использования системы (особенно в наиболее употребительных стандартных ситуациях) применяется принцип "умолчания".

Принцип ортогональности здесь заключается в следующем: средства базиса разделены на непересекающиеся графические функции, каждая из которых реализована наиболее универсальным образом. Принцип умолчания заключается в том, что если не задано не-

которое действие (или значение), то выполняется стандартное действие (или значение полагается стандартным).

Пользователь обращается к системе СМОГ с помощью операторов вызова подпрограмм в языке Фортран или ПЛ/I (в ОС или ДОС ЕС ЭВМ), сами же подпрограммы хранятся в виде объектных и загрузочных модулей и подключаются к программе пользователя на этапе редактирования и выполнения.

Операторы базиса СМОГ разделены на подготовительные, выполняющие графические функции без вывода конкретных изображений, и исполнительные, осуществляющие построение рисунков. Исполнительные делятся на уровни. Операторы первого уровня рисуют элементарные графические объекты (линии и символы), которые могут (хотя бы частично) рассчитываться в аппаратуре графопостроителя. Операторы второго уровня изображают более сложные объекты (тексты, числа, фрагменты-рисунки), которые исполняются программно, и которые обязательны для базиса графической системы. Наконец, операторы третьего уровня реализуют проблемно-ориентированные надстройки. Реализация надстройки для изображения графиков описана в [3].

В статье кратко описываются возможности базиса и особенности его реализации.

2. Базис системы СМОГ

В операторах применяются параметры следующих видов: слова с фиксированной точкой (целые), слова с плавающей точкой (вещные), символьные строки. Некоторые параметры (строки) являются составными: каждый символ в них имеет самостоятельное значение. Такие символы всегда являются цифрами, после них помещается звездочка, указывающая конец строки. Нулевые символы в конце строки (до звездочки) можно опускать, поэтому нулевое значение всей строки можно изобразить одной звездочкой (это соответствует принципу умолчания).

Первым оператором СМОГ в программе пользователя должен быть оператор

```
CALL SMOG (A);
```

который подготавливает систему к работе: открывает выходной файл, приводит графопостроитель в исходное положение. Параметр A , являющийся оставшим, первой цифрой задает модель графопостроителя (если их несколько). В зависимости от этой цифры в память ЭВМ загружается один из загрузочных модулей, каждый из которых реализует обращение к "своему" графопостроителю.

Для закрытия выводного файла необходим оператор

CALL SMOG ('END');

Перед построением изображения необходимо выделить участок листа бумаги графопостроителя. Оператором

CALL FORM ('/200/');

выделяется, например, полоса бумаги длиной 200 мм и шириной, определяемой размерами графопостроителя (графопостроитель может быть как планшетным, так и рулонным). Оператором

CALL FORM ('11Г');

задается горизонтальный формат II по ГОСТу. Его фактическое расположение на листе для программиста не играет роли. В обоих случаях после этого осуществляется защита формата: блокировка проведения линий, случайно выходящих за его границы.

Внутри формата считается заданной прямоугольная система координат с именем '@'. Новая аффинная система координат (с именем '*') получается из имеющейся (она обозначается ниже именем M) путем применения одного из координатных преобразований:

CALL SHIFT (M, DX, DY); сдвиг на DX, DY мм;

CALL SCALE (M, KX, KY); изменение масштаба в KX, KY

раз вдоль осей координат;

CALL ROT (M, W); поворот на угол W градусов;

CALL ANGLE (M, F); изменение угла раскрытия осей координат на F градусов.

Чтобы получившуюся (текущую) систему координат можно было в дальнейшем использовать, ей необходимо задать "настоящее" имя (строка букв и цифр со звездочкой в конце) оператором

CALL NAMSC (M);

Этими операторами можно построить дерево координат ("выросшее" из системы координат '@'). При задании нового формата дерево ликвидируется.

Операторами I-го уровня изображаются линии, они состоят из типа и совокупности узловых точек. Оператор

CALL TYPE (T);

задает начало линии и указывает её тип (набор характеристик, задаваемый составным параметром *T*). Первая характеристика - форма линии. 0 - ломаная; 1 - ломаная замкнутая; 2 - незамкнутая гладкая кривая (сплайн I); 3 - то же, но замкнутая; 4 - незамкнутый сплайн 2 и т.д. Всего в системе 4 вида сплайнов, предназначенных для изображения различных фигур 4. Вторая характеристика: 0 - сплошная линия, 1 - штриховая, 2 - штрихпунктирная и т.д. Третья задает величину, а четвертая - номер маркера в узловых точках. Пятая указывает номер пера. Координаты каждой из узловых точек задаются оператором

CALL POINT (X,Y);

Одна или несколько линий (их совокупность называется текущим фрагментом) рисуется на листе бумаги в системе координат *M* оператором

CALL DRAW (M);

Операторы 2-го уровня изображают тексты, числа и фрагменты-рисунки. Фрагменты-рисунки описываются на специальном графическом языке, допускающим компактную запись изображений. Его реализация описана в [5].

Тексты и числа состоят из символов. Символы рисуются на целочисленной решетке (трафарете), её размеры задаются оператором

CALL TRAF (M,R);

Составной параметр *R* задает характеристики трафарета: высоту шрифта, наклон, ширину, номер пера. *M* - имя системы координат, к которой привязан трафарет. Нулевая строка и нулевая позиция трафарета совпадают с нулем этой системы координат. Текст *S* изображается оператором

CALL TEXT (C,I,J);
начиная с J -й позиции I -й строки.

3. Внутренняя структура базиса

Операторы I-го уровня и подготовительные обмениваются между собой информацией, поэтому они реализованы как набор точек входа объектного модуля *SMOG*. Имена точек входа следующие:

SMOG, NAMSC, DELSC (этот оператор удаляет часть систем координат), *SHIFT, ANGLE, SCALE, ROT, FORM, BLANK* (для задания бланков на поле чертежа), *TYPE, POINT, DRAW, DECTL* (объявление внешнего типа линии), *DELTL* (удаление объявленных внешних типов линий), *TRAF, DELTR* (удаление трафаретов), *TEXT, SMOOTE* (служебная точка входа). Другие операторы 2-го, а также 3-го уровней реализуются как отдельные объектные модули, в которых имеется вызов модуля *SMOG*.

При включении системы оператором *SMOG* задается номер графопостроителя, и соответствующий загрузочный модуль подключается к программе пользователя. Благодаря такому механизму система СМОГ может работать с самыми разными типами графопостроителей.

В [6] приведены параметры ряда советских и зарубежных шаговых графопостроителей, многие из них могут подключаться к ЕС ЭВМ. Минимальные способности графопостроителя состоят в выполнении команд элементарного перемещения (на 1 шаг), поднятия и опускания пера. Следующей ступенью является наличие линейного интерполятора (шаговая интерполяция отрезка прямой производится аппаратурой). Другие аппаратные возможности, имеющиеся в некоторых графопостроителях (интерполяция окружностей, изображение штриховых линий символов), реализованы с такими неудобными ограничениями, что их практически невозможно использовать в универсальной графической системе, и в СМОГ они не применяются.

Загрузочный модуль моделирует "идеальный" графопостроитель, в котором "реализованы" следующие функции: слежение за текущим положением пера, задание и защита формата, защита бланков, рисование отрезков прямых и кубических сплайнов, изображение как сплошных, так и различных штриховых линий (в том числе с разры-

вами около узловых точек), изображение символов, координатные преобразования изображений. Модуль *SMO* непосредственно выполняет следующие функции: управление памятью, раскрой бумаги, формирование и запоминание дерева координат, формирование и хранение информации с линий, расчет коэффициентов прямых и отрезков сплайнов при изображении линий, запоминание трафаретов, преобразование текстов из входной кодировки во внутреннюю, диагностика ошибочных значений в операторах СМОГа и ряд других сервисных действий.

4. Реализация подпрограмм базиса

В модуле *SMO* имеется буферный массив. Операторами модуля в нем формируется информация, которой эти операторы обмениваются (см. рис. I). Указатели (целые переменные) отмечают в массиве соответствия: *PP* - конец узловых точек текущего фрагмента, *PC* - конец систем координат, *P* - граница раздела массива (начало систем координат и начало внешних типов линий), *PT* - конец внешних типов линий, *PL* - конец типов линий текущего фрагмента, *PB* - начало типов линий текущего фрагмента.

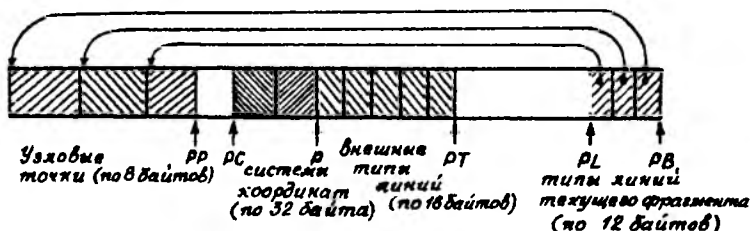


Рис. I. Буферный массив

По мере заполнения массива может случиться, что указатели *PP* и *PC* (или *PT* и *PL*) перекроют друг друга. В этом случае производится сдвиг информации, расположенной в центральной части массива (систем координат и внешних типов линий) вправо или влево. Если же сдвиг невозможен, то система прекращает работу из-за нехватки памяти. Величина буферного массива задается при включе-

ложение начала системы координат внутри формата, задает загрузочному модулю параметры формата и формирует в буферном массиве координаты формата и текущую систему координат (с одинаковыми параметрами).

проц BLANK = (лит N , вещ X,Y,A,B):
если ПРОЛОГ ; НОМЕР БЛАНКА ВЕРЕН
то ПЕРЕВОД В ФИЗИЧЕСКУЮ СИСТЕМУ КООРДИНАТ ;
ЗАДАНИЕ БЛАНКА
иначе ОШИБКА
илисе

проц NAMSC = (строк M):
если ПРОЛОГ ; M = '*' VM = '@' то ОШИБКА
иначе АНАЛИЗ ИМЕНИ M ;
РС минус 32 ;
если PP > PC то СДВИГ БУФЕРА НАПРАВО илисе
если не хватило памяти
ЗАПИСЬ ИМЕНИ В ТЕКУЩУЮ СИСТЕМУ КООРДИНАТ ;
КОПИЯ ТЕКУЩЕЙ СИСТЕМЫ КООРДИНАТ
илисе

Процедура NAMSC записывает в текущую систему координат имя M и копирует ее параметры, создавая новую текущую систему координат. Координатные преобразования осуществляются над текущей системой координат без копирования. Для примера рассмотрим процедуру ROT .

проц ROT = (отрок M, вещ W):
начало ПРОЛОГ ;
ЧТЕНИЕ ПАРАМЕТРОВ СИСТЕМЫ КООРДИНАТ M ;
если W НЕ СЛИШКОМ БЕЛИКО
то ПРЕОБРАЗОВАНИЕ ПОВОРОТА ;
если ПАРАМЕТРЫ НЕ СЛИШКОМ БЕЛИКИ
то ЗАПИСЬ ПАРАМЕТРОВ В ТЕК СИСТ КООРД
иначе ОШИБКА илисе
иначе ОШИБКА илисе
конец

Остальные операторы задания координат имеют аналогичную структуру.

```
проц DELSC = (строка M):  
  если ПРОЛОГ ; M ≠ '*' то  
    если M = @ то PC := P-64 ;  
    КОПИЯ ТЕКУЩЕЙ СИСТЕМЫ КООРДИНАТ  
  иначе АНАЛИЗ ИМЕНИ M ;  
    цел указатель := PC ;  
    ПОИСК ИМЕНИ M ;  
    если ИМЯ НАЙДЕНО то PC := указатель ;  
    УДАЛЕНИЕ ИМЕНИ M  
  иначе ОШИБКА илсе  
  илсе
```

Процедура DELSC делает текущей систему координат, имевшую ранее имя M.

```
проц TYPE = (строка T):  
  начало ПРОЛОГ ; АНАЛИЗ ТИПА ЛИНИИ ;  
    если БЫЛО ИЗОБРАЖЕНИЕ то СБРОС ТЕК ФРАГМЕНТА илсе ;  
    PL минус 12 ;  
    если PT > PL то СДВИГ БУФЕРА ВЛЕВО илсе ;  
    ‡ если не хватило памяти ‡  
    ЗАПИСЬ ТИПА ЛИНИИ  
  конец
```

```
проц POINT = (вещ X,Y):  
  если ПРОЛОГ ; КООРДИНАТЫ НЕ СЛИШКОМ ВЕЛИКИ  
    то ЧТЕНИЕ ПРЕДЫДУЩЕЙ ТОЧКИ ;  
    если РАЗНОСТЬ КООРДИНАТ ТОЧЕК СУЩЕСТВЕННА  
      то PP плюс 8 ;  
      если PP > PC то СДВИГ БУФЕРА ВПРАВО илсе ;  
      ‡ если не хватило памяти ‡  
      ЗАПИСЬ ТОЧКИ илсе  
  иначе ОШИБКА  
  илсе
```


При занесении в массив очередной точки проверяется, чтобы ее координаты не были слишком велики по абсолютной величине и чтобы разница между соседними точками была больше некоторого малого числа, так как в противном случае при интерполяции линий по этим точкам были бы возможны аварийные ситуации.

проц DRAW = (строка M):

если ПРОЛОГ ; ПРИЗНАК ИЗОБРАЖЕНИЯ; M = '#'

то ЧТЕНИЕ ПАРАМЕТРОВ СИСТЕМЫ КООРДИНАТ M ;

если PP > 0 то

ЧТЕНИЕ НАЧАЛЬНОЙ ТОЧКИ ;

если КООРДИНАТЫ НЕ СОВПАДАЮТ

то УСТАНОВКА НОВЫХ КООРДИНАТ илсе ;

для указатель от PB-12 через -12 до PL цикл

ЧТЕНИЕ ТИПА ЛИНИИ ;

если число точек > 0 то

ПОДВОД ПЕРА К НАЧАЛЬНОЙ ТОЧКЕ ;

если НОМЕР ПЕРА НОВЫЙ

то УСТАНОВКА ПЕРА илсе ;

если число точек = 1 то ТОЧКА ;

если ЕСТЬ МАРКЕР то

ИЗОБРАЖЕНИЕ МАРКЕРА илсе

иначе если число точек = 2 то

ИНТЕРПОЛЯЦИЯ ЛИНЕЙНАЯ илсе ;

если ШТРИХИ НОВЫЕ то

УСТАНОВКА ШТРИХОВ илсе ;

если РАЗРЫВЫ В УЗЛАХ НОВЫЕ то

УСТАНОВКА РАЗРЫВОВ илсе ;

если МАРКЕР ЕСТЬ то

УСТАНОВИТЬ РАЗМЕР ШРИФТА илсе ;

если СПЛАЙН то

ИЗОБРАЖЕНИЕ СПЛАЙНА

иначе ИЗОБРАЖЕНИЕ ЛОМАННОЙ илсе

илсе

илсе ;

лкци

илсе

илсе

Процедура перед изображением текущего фрагмента устанавливает действующей систему координат, в которой этот фрагмент изображается. Затем просматривает типы линий текущего фрагмента (где содержатся также указатели на начальные узловые точки линий) и устанавливает, если это необходимо, действующими такие параметры, как разрывность по длине линии, разрывы в узлах, величина маркера, номер пера. После этого изображается ломаная или сплайн. Более подробный алгоритм изображения сплайнов, применяемых в системе СМОГ, приведен в [4].

проц DECTL = (строка M, N, S, L, T, R, C, P):

начало ПРОЛОГ ;

АНАЛИЗ ИМЕНИ ТИПА ЛИНИИ M ;

если ИМЯ ВЕРНО то

АНАЛИЗ ТИПА ЛИНИИ N И ЕГО ХАРАКТЕРИСТИК ;

АНАЛИЗ ХАРАКТЕРИСТИК ТИПА ЛИНИИ M ;

если ХАРАКТЕРИСТИКИ ВЕРНЫ

то PT целое 16 ;

если PT > PL то СДВИГ БУФЕРА ВЛЕВО илсе ;

⌘ если не хватило памяти ⌘

ЗАПИСЬ ТИПА ЛИНИИ

иначе ОШИБКА илсе

иначе ОШИБКА илсе

конец

Процедура вначале формирует характеристики из типа линии N, а затем некоторые из них заменяет характеристиками, указанными как параметры процедуры (те, значение которых отличается от '#').

проц DECTL = (строка M):

начало ПРОЛОГ ;

АНАЛИЗ ИМЕНИ ТИПА ЛИНИИ M ;

если ИМЯ ВЕРНО то цел указатель := PT ;

ПОИСК ТИПА ЛИНИИ ;

если ТИП ЛИНИИ M НАЙДЕН то PT := указатель

иначе ОШИБКА илсе

иначе ОШИБКА илсе

конец

Процедура удаляет типы линий, начиная с М.

проц TRAF = (строка М, А):

начало ПРОЛОГ;

АНАЛИЗ ИМЕНИ М;

АНАЛИЗ ХАРАКТЕРИСТИК;

СДВИГ ВНИЗ СТЕКА ТРАФАРЕТОВ;

ЗАПИСЬ ТРАФАРЕТА

конец

Процедура TRAF задает новый трафарет в вершине стека трафаретов. При этом может быть уничтожен самый старый заданный трафарет, если ему не хватило места. Обнаружено это может быть лишь при попытке его восстановить процедурой DELTA.

проц DELTA = (цел L):

если ПРОЛОГ; L > 0

то СДВИГ ВВЕРХ СТЕКА ТРАФАРЕТОВ;

если ТРАФАРЕТ ОТСУТСТВУЕТ то ОШИБКА;

ЗАПИСЬ НУЛЕВОГО ТРАФАРЕТА иначе

иначе

проц TEXT = (отрок А, цел I, J):

начало ПРОЛОГ;

ЧТЕНИЕ ТРАФАРЕТА;

ЧТЕНИЕ ПАРАМЕТРОВ СИСТЕМЫ КООРД ТРАФАРЕТА;

если ШРИФТ НАКЛОННЫЙ ИЛИ УЗКИЙ ИЛИ ШИРОКИЙ

то ПРЕОБРАЗОВАНИЕ ПАРАМЕТРОВ КООРДИНАТ

иначе;

если КООРДИНАТЫ НЕ СОВПАДАЮТ

то УСТАНОВКА НОВЫХ КООРДИНАТ иначе;

если РАЗМЕРЫ ШРИФТА НЕ СОВПАДАЮТ

то УСТАНОВКА ШРИФТА иначе;

если НОМЕР ПЕРА НОВЫЙ то УСТАНОВКА ПЕРА иначе;

ПЕРЕСЧЕТ СТРОКИ И ПОЗИЦИИ В КООРДИНАТЫ;

ПОДВОД ПЕРА К НАЧАЛУ;

цел i := 1;

пока А [i] ≠ '*' цикл

цел повторитель: = 1 ;
 ПЕРЕВОД СИМВОЛА ВО ВНУТРЕННЮЮ КОДИРОВКУ ;
до повторитель цикл
 если СИМВОЛ УПРАВЛЯЮЩИЙ
 то ПОДВОД ПЕРА
 иначе РИСОВАНИЕ СИМВОЛА илсе
 лкиц
лкиц
конец

Каждый символ закодирован таким образом, что после его изображения перо передвигается в начальную позицию для изображения следующего символа. Этот порядок может быть изменен управляющим символом, тогда производится подвод пера к другой точке.

Описанные процедуры обращаются к загрузочному модулю при установке новых координат, новых видов разрывностей, при задании формата, бланков и при других установочных действиях, а также при изображении отрезков прямых, сплайнов и символов.

Л и т е р а т у р а

1. Гладких Б.А., Костюк Ю.Л., Позолотин В.А. СМОГ - система математического обеспечения графопостроителя. Томск, Изд-во Томского университета, 1974.
2. Костюк Ю.Л. Система математического обеспечения графического вывода для ЕС ЭВМ. Томск, Изд-во Томского университета, 1977.
3. Костюк Ю.Л. Универсальная процедура рисования графиков. Наст. сборник.
4. Костюк Ю.Л. Применение сплайнов для изображения линий в машинной графике. Наст. сборник.
5. Костюк Ю.Л., Фуко А.Л. Язык представления рисунков в системе СМОГ. Наст. сборник.
6. Зозулевич Д.М. Машинная графика в автоматизированном проектировании. М., "Машиностроение", 1976.

ПРИМЕНЕНИЕ СПЛАЙНОВ ДЛЯ ИЗОБРАЖЕНИЯ ЛИНИЙ В МАШИННОЙ ГРАФИКЕ

В.Л.Костык

I. Введение

Элементарными объектами в графической системе являются линии: изображение, построенное графопостроителем на бумаге или лучом на экране дисплея, состоит из линий. Сама же линия задается узловыми точками, через которые она должна проходить. Если соседние точки соединить отрезками прямых, получится ломаная (единственная). Если же известно, что линия является гладкой (с непрерывной касательной), то изобразить такую линию можно не единственным образом. А если точки таковы (например, являются отсчетами со случайными ошибками), что линия должна проходить "в среднем" близко ко всей их совокупности, то такую линию можно изобразить только исходя из конкретной физической природы ошибок.

Будем считать, что гладкая линия проходит через узловые точки (которые, быть может, сами являются сглаженными). Известно немало методов гладкого восполнения кривых, однако в графической системе следует применять только такие, которые обладают рядом полезных свойств: устойчивостью при добавлении новых узловых точек; универсальностью — возможностью изображать кривую при любом расположении точек; инвариантностью относительно координатных преобразований.

Из всех интерполирующих функций сплайны (кусочно-непрерывные полиномы) являются самыми устойчивыми, их устойчивость тем выше, чем меньше степень (нечетная). Таким образом, наиболее

подходящи кубические сплайны. Однако сплайны $Y(X)$ (см., например, [1,2]) не универсальны — последовательность координат X точек должна быть возрастающей —, поэтому для вычерчивания кривых необходимы сплайны в параметрической форме $X(t)$, $Y(t)$, где t принимает возрастающую последовательность значений $t_0 < t_1 < \dots < t_n$ в узловых точках (x_0, y_0) , (x_1, y_1) , ..., (x_n, y_n) (см. раздел 2.6 в [1]).

Отрезок сплайна между соседними точками (x_{i-1}, y_{i-1}) и (x_i, y_i) представляется кубическими полиномами

$$\begin{aligned} X_i(t) &= a_{0i} + a_{1i}t + a_{2i}t^2 + a_{3i}t^3, \\ Y_i(t) &= b_{0i} + b_{1i}t + b_{2i}t^2 + b_{3i}t^3. \end{aligned} \quad (1)$$

Если положить $t = 0$ в точке (x_{i-1}, y_{i-1}) и $t = 1$ в точке (x_i, y_i) , то коэффициенты полинома $X_i(t)$ определяются из соотношений

$$\begin{aligned} a_{0i} &= x_{i-1}, \\ a_{1i} &= x'_{i-1}, \\ a_{2i} &= 3 \cdot \Delta x_i - 2 \cdot x'_{i-1} - x'_i, \\ a_{3i} &= x'_i - x'_{i-1} + 2 \cdot \Delta x_i, \end{aligned} \quad (2)$$

где x'_{i-1} , x'_i — коэффициенты X производных сплайна в $(i-1)$ -й и i -й точках соответственно, $\Delta x_i = x_i - x_{i-1}$. Для полинома $Y_i(t)$ соотношения аналогичны.

Таким образом, для вычисления коэффициентов отрезка сплайна между соседними узловыми точками необходимо вычислить производные в этих точках. Однако производные могут вычисляться не единственным образом, поэтому можно получать различные виды сплайнов, обладающих теми или иными свойствами.

В [1, 2] производные в узловых точках вычисляются из условия непрерывности кривизны в узловых точках, то есть приравниваются вторые производные соседних отрезков сплайна в узловых точках, а в [3] — приравниваются кривизны соседних отрезков сплайна. В результате для n точек получаются 2 системы n линейных или система $4n$ нелинейных уравнений, для решения которых в ЭВМ

требуется много времени и оперативной памяти.

Так как незначительные разрывы кривизны незаметны на глаз, то для изображения кривых можно применять локальные сплайны, в которых для вычисления производной в некоторой узловой точке используются координаты только нескольких соседних узловых точек. При этом производные можно вычислять не все сразу, а последовательно, по мере изображения кривых, и тем самым сэкономить время и память. В [4] описаны программы вычисления локальных 5-точечных сплайнов $Y(x)$, в которых производная в i -й точке рассчитывается по координатам пяти точек: $i-2$, $i-1$, i , $i+1$, $i+2$.

В статье рассматриваются самые простые - 3-точечные локальные сплайны в параметрической форме. Для них удалось выяснить ряд полезных свойств, на основании которых можно считать, что такие сплайны наиболее удобны для изображения кривых.

2. Простейший локальный сплайн

Вторая производная полинома (I) с коэффициентами (2):

$$X_i''(t) = 2(3\Delta x_i - 2x_{i-1}' - x_i') + 5(x_{i-1}' + x_i' + 2\Delta x_i)t. \quad (3)$$

Здесь и далее будем рассматривать только полином $X(t)$, для полинома $Y(t)$ соотношения симметричны.

Для вычисления производной в точке i рассмотрим сплайн, проходящий через три точки: $i-1$, i , $i+1$. Условие непрерывности второй производной в i -й точке

$$X_i''(t) = X_{i+1}''(0). \quad (4)$$

Добавив к нему краевые условия $X_i''(0) = 0$ и $Y_{i+1}''(1) = 0$, обеспечивающие равенство кривизны нулю на концах такого сплайна, получим систему трех уравнений:

$$\left. \begin{aligned} 2x_{i-1}' + x_i' &= 3\Delta x, \\ x_{i-1}' + 4x_i' + x_{i+1}' &= 3(\Delta x_i + \Delta x_{i+1}), \\ x_i' + 2x_{i+1}' &= 3\Delta x_{i+1}. \end{aligned} \right\} \quad (5)$$

Её решение

$$x'_i = \frac{\Delta x_i + \Delta x_{i+1}}{2}. \quad (6)$$

Простейший локальный сплайн, производные в узловых точках которого вычисляются по формуле (6), применяется в некоторых системах графического ввода (например, в [5]), там эти формулы получены из других соображений. Однако этот сплайн неустойчив: при неравномерно расположенных узловых точках могут образоваться нежелательные петли и изгибы.

3. Нормирование производных по длине хорды

Петли и изгибы получаются из-за того, что на коротких отрезках сплайна, с которыми соседствуют длинные, производные имеют чрезмерно большую величину. Избавимся от этого, проинтерполировав производные сплайна по длине отрезка. Тогда производная в точке i на отрезке $i-1, i$ будет равна $L_i x'_i$, а на отрезке $i, i+1$ соответственно $L_{i+1} x'_i$. Производная при этом терпит разрыв, однако наклон (т.е. касательная) остается непрерывным:

$$\frac{L_i y'_i}{L_i x'_i} = \frac{L_{i+1} y'_i}{L_{i+1} x'_i}.$$

Здесь L_i , L_{i+1} — длины отрезков сплайна $i-1, i$ и $i, i+1$. В качестве их оценок можно взять длины хорд между соответствующими узловыми точками:

$$L_i = S_i + \sqrt{\Delta x_i^2 + \Delta y_i^2}, \quad L_{i+1} = S_{i+1} + \sqrt{\Delta x_{i+1}^2 + \Delta y_{i+1}^2} \quad (7)$$

Уравнение сплайна на отрезке $i-1, i$ будет таким:

$$\begin{aligned} X_{Ai}(t) = & x_{i-1} + x'_{i-1} S_i t + (3\Delta x_i - 2x'_{i-1} S_i - x'_i S_i) t^2 + \\ & + (x'_{i-1} S_i + x'_i S_i - 2\Delta x_i) t^3. \end{aligned} \quad (8)$$

Вычисляя x'_i аналогично соотношениям (4)-(6), получим

$$x'_i = \frac{\Delta x_i + \Delta x_{i+1}}{S_i + S_{i+1}}. \quad (9)$$

До сих пор предполагалось, что параметр t изменяется от 0 до 1 на отрезке $i-1, i$. Естественнее считать, что параметр t является длиной сплайна ([1], раздел 2.6). В качестве оценки длины отрезка сплайна можно снова взять длину хорды S_i , параметр t при этом изменяется от 0 до S_i . Уравнение сплайна

$$\begin{aligned} \chi_{Si}(t) = & x_{i-1} + x'_{i-1} S_i \frac{t}{S_i} + (3\Delta x_i - 2x'_{i-1} S_i - x'_{i-1} S_i) \frac{t^2}{S_i^2} + \\ & + (x'_{i-1} S_i + x'_i S_i - 2\Delta x_i) \frac{t^3}{S_i^3}. \end{aligned} \quad (10)$$

Система уравнений для вычисления x'_i

$$\left. \begin{aligned} \chi''_{Si}(S_i) &= \chi''_{Si+1}(0), \\ \chi''_{Si}(0) &= 0, \\ \chi''_{Si+1}(S_{i+1}) &= 0. \end{aligned} \right\}$$

Её решение

$$x'_i = \frac{S_i S_{i+1}}{S_i + S_{i+1}} \left(\frac{\Delta x_i}{S_i^2} + \frac{\Delta x_{i+1}}{S_{i+1}^2} \right). \quad (11)$$

Наконец, рассмотрим промежуточный вариант, полагая, что на отрезке $i-1, i$ сплайна параметр t изменяется от 0 до $\sqrt{S_i}$. Тогда уравнение сплайна

$$\begin{aligned} \chi_{Si}(t) = & x_{i-1} + x'_{i-1} S_i \frac{t}{\sqrt{S_i}} + (3\Delta x_i - 2x'_{i-1} S_i - x'_{i-1} S_i) \frac{t^2}{S_i^2} + \\ & + (x'_{i-1} S_i + x'_i S_i - 2\Delta x_i) \frac{t^3}{S_i^2 \sqrt{S_i}}. \end{aligned} \quad (12)$$

Вычисляя x'_i аналогичным образом, получим

$$x'_i = \frac{1}{2} \left(\frac{\Delta x_i}{S_i} + \frac{\Delta x_{i+1}}{S_{i+1}} \right). \quad (13)$$

Полученные сплайны более устойчивы, чем простейший сплайн, однако они значительно уступают полным сплайнам в тех случаях, когда требуется начертить "выпуклую" кривую, заданную небольшим количеством узловых точек. Так, изображение окружности по 4 узловым точкам с помощью полного сплайна дает максимальное отклонение около 1%, а одним из описанных локальных сплайнов - около 10% по радиусу (см.рис.1).

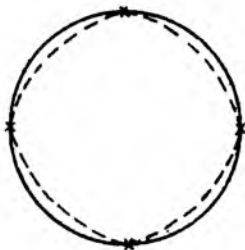


Рис. 1. Полный сплайн-сплошная линия, локальный - штриховая

4. Нормализация сплайна на сегменте

Улучшить свойства локальных сплайнов можно следующим образом: производную сплайна, вычисленную по формуле (9), (11) или (13), нормировать на единичную длину и использовать как направляющий вектор производной. Длину же производной вычислить, оценив более точно длину отрезка сплайна, для чего можно воспользоваться приемом нормализации сплайна на сегменте.

Рассмотрим отрезок сплайна, совпадающий с окружностью в точках А, В, С (см. рис.2) и образующий в точках А, С угол α с хордой S. Как показано в [3], абсолютная длина производной сплайна в точках А и С

$$L = \frac{2S}{1 + \cos \alpha} \quad (14)$$

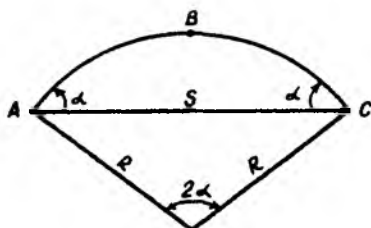


Рис.2. Оценка длины дуги сплайна

Мы же оценим длину дуги сплайна из других соображений. Длина дуги окружности, проходящей через точки А, В, С, равна

$$L = S \frac{\pi}{\sin \alpha} \approx \frac{3S}{2 + \cos \alpha} \quad (15)$$

Оценка (14) дает чрезмерно "выпуклый" сплайн, который оказывается неустойчив: $L \rightarrow \infty$ при $\alpha \rightarrow \pi$. Такой сплайн, в частности, дает несколько большую ошибку, чем сплайн по (15) при изображении окружности по 4 и более узловым точкам.

Длину производной сплайна при одинаковых углах между касательной и хордой в точках $i-1, i$ положим равной длине дуги (15), а при различных углах (аналогично [3]):

$$L_{i-1} = \frac{3S_i}{2 + \frac{2}{3} \cos \alpha_{i-1} + \frac{1}{3} \cos \alpha_i},$$

$$L_i = \frac{3S_i}{2 + \frac{1}{3} \cos \alpha_{i-1} + \frac{2}{3} \cos \alpha_i} \quad (16)$$

Таким образом, после вычисления производных $x'_{i-1}, x'_i, y'_{i-1}, y'_i$ по формуле (9), (11) или (13), их нормализованные значения вычисляются по формулам:

$$x'_{i-1} = \frac{9S_i^2 \cdot u_{i-1}}{6S_i + 2(u_{i-1} \Delta x_i + w_{i-1} \Delta y_i) + (u_i \Delta x_i + w_i \Delta y_i)}, \quad (17)$$

$$x_{n_i}' = \frac{9S_i^2 u_i}{6S_i + (u_{i-1} \Delta x_i + W_{i-1} \Delta y_i) + 2(u_i \Delta x_i + W_i \Delta y_i)},$$

где

$$u_i = \frac{x_i'}{\sqrt{x_i'^2 + y_i'^2}}; \quad W_i = \frac{y_i'}{\sqrt{x_i'^2 + y_i'^2}}.$$

— нормированные на единичную длину производные.

Такие нормализованные сплайны практически не уступают полным сплайнам при изображении "выпуклых" кривых по небольшому числу точек. Например, окружность по 4 точкам они изобразят с погрешностью 1%, как и полные сплайны.

5. Краевые условия

При изображении незамкнутых кривых, чтобы вычислить производные в начальной (x_0, y_0) и конечной (x_n, y_n) точках, необходимо в этих точках задать краевые условия. В большинстве случаев хорошим краевым условием является равенство нулю кривизны сплайна (или его вторых производных) в крайних точках. Локальные трехточечные сплайны наиболее устойчивы по отношению к краевым условиям, которые влияют только на начальный и конечный отрезки сплайна.

Для вычисления производных в точках (x_0, y_0) и (x_n, y_n) , достаточно решить соответствующие системы уравнений (5) и им подобные для других сплайнов) при $i = 1$ и $i = n-1$ относительно x_0' и x_n' .

Для простейшего сплайна

$$x_0' = \frac{3}{2} \Delta x, -\frac{1}{2} x_1', \quad x_n' = \frac{3}{2} \Delta x_n - \frac{1}{2} x_{n-1}'. \quad (18)$$

Для нормированных сплайнов А, Б, В:

$$x_0' = \frac{3}{2} \frac{\Delta x_1}{S_1} - \frac{1}{2} x_1', \quad x_n' = \frac{3}{2} \frac{\Delta x_n}{S_n} - \frac{1}{2} x_{n-1}'. \quad (19)$$

В некоторых случаях указанные краевые условия не годятся, например, при изображении дуги окружности по нескольким узловым точкам. В этом случае более естественным краевым условием можно считать постоянство кривизны на конечных отрезках сплайна. Производная сплайна x'_0, y'_0 вычисляется из условия равенства скалярных произведений векторов (x'_0, y'_0) , $(\Delta x, \Delta y)$ и (x'_1, y'_1) , $(\Delta x, \Delta y)$, а также равенства для векторов (x'_0, y'_0) и (x'_1, y'_1) : (см. рис.3)

$$\left. \begin{aligned} x'_0 \Delta x + y'_0 \Delta y &= x'_1 \Delta x + y'_1 \Delta y, \\ x_0'^2 + y_0'^2 &= x_1'^2 + y_1'^2. \end{aligned} \right\} \quad (20)$$

Одно из двух решений системы (20): $x'_0 = x'_1$, $y'_0 = y'_1$ следует отбросить. Второе решение

$$\begin{aligned} x'_0 &= \frac{1}{\Delta s} [x'_1 (\Delta x_1^2 - \Delta y_1^2) + 2y'_1 \cdot \Delta x_1 \cdot \Delta y_1], \\ y'_0 &= \frac{1}{\Delta s} [y'_1 (\Delta y_1^2 - \Delta x_1^2) + 2x'_1 \cdot \Delta y_1 \cdot \Delta x_1] \end{aligned} \quad (21)$$

дает искомые значения производных x'_0, y'_0 .

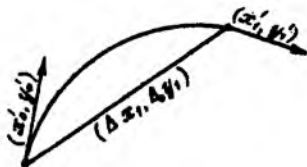


Рис.3. Краевое условие постоянства кривизны на конечных отрезках сплайна

Выражения для производных x'_0, y'_0 симметричны.

6. Свойства локальных трехточечных сплайнов

Рассмотрим инвариантность полученных сплайнов относительно координатных преобразований. Общее аффинное преобразование записывается в виде

$$x_n = d_1 + d_2 x + d_3 y, \quad y_n = d_4 + d_5 x + d_6 y, \quad (22)$$

где x, y - старые, x_n, y_n - новые координаты, а $d_1, \dots, d_6$ - коэффициенты преобразования.

Сплайн инвариантен относительно преобразования, если преобразованный сплайн от исходных узловых точек совпадает со сплайном от преобразованных узловых точек.

Рассмотрим сплайн (1) с коэффициентами (2), производные у которого можно представить в виде

$$x'_i = \sum_{k=1}^n \kappa_{ik} \Delta x_k, \quad y'_i = \sum_{k=1}^n \kappa_{ik} \Delta y_k, \quad (23)$$

где коэффициенты κ_{ik} таковы, что не меняются при преобразовании (22). Такой сплайн инвариантен относительно преобразования (22), так как сплайн, подвергнутый преобразованию

$$X_{ni}(t) = d_1 + d_2 X_i(t) + d_3 Y_i(t),$$

равен сплайну от преобразованных узловых точек

$$X_{ni}(t) = x_{ni-1} + x'_{ni-1} t + (3\Delta x_{ni} - 2x'_{ni-1} - x_{ni}) t^2 + \\ + (x'_{ni-1} + x'_{ni} - 2\Delta x_{ni}) t^3,$$

где

$$x_{ni-1} = d_1 + d_2 x_{i-1} + d_3 y_{i-1};$$

$$\Delta x_{ni} = d_2 \Delta x_i + d_3 \Delta y_i;$$

$$x'_{ni} = d_2 \cdot x'_i + d_3 \cdot y'_i.$$

Простейший локальный сплайн с производными (6) и (18), как частный случай сплайна (23), инвариантен относительно преобразования (22). Нормированные сплайны А, Б, В не инвариантны, так как κ_{ik} у них изменяются при преобразовании (22).

Рассмотрим частичное аффинное преобразование

$$x_n = d_1 + m \cdot \cos \varphi \cdot x - m \cdot \sin \varphi \cdot y, \\ y_n = d_4 + m \cdot \sin \varphi \cdot x + m \cdot \cos \varphi \cdot y, \quad (24)$$

включающее сдвиг начала координат на d_1 , d_2 , поворот системы координат на угол φ и изменение масштаба в m раз. Коэффициенты h_{ik} рассмотренных сплайнов при таком преобразовании не изменяются (так как являются дробями вида $\frac{S_i}{S_{i-1} + S_{i+1}}$, $\frac{S_{i+1}}{S_i + S_{i+1}}$ и т.д.), т.е. сплайны инвариантны относительно преобразования (24).

Нетрудно показать также, что нормализация сплайнов по формулам (17) и применение краевых условий (19), (21) также сохраняет инвариантность сплайнов относительно (24).

Рассмотренными свойствами инвариантности обладают не только локальные трехточечные, но и соответствующие полные сплайны в параметрической форме. Далее рассмотрим свойства, которыми обладают только локальные трехточечные сплайны.

При неравномерном расположении точек сплайны А, Б и В ведут себя совершенно по-разному. Для иллюстрации рассмотрим рис. 4.

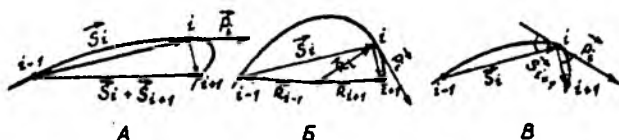


Рис. 4. Поведение различных сплайнов при неравномерном расположении узловых точек

Производная сплайна А в точке i параллельна хорде, соединяющей $(i-1)$ -ю и $(i+1)$ -ю точки, что следует из формулы (9):

$$x'_i = \frac{x_{i+1} - x_{i-1}}{S_i + S_{i+1}}.$$

Таким образом, на более коротком отрезке сплайн А более выпуклый, чем на более длинном соседнем.

Производная сплайна Б в точке i касательна к окружности, проведенной по точкам $i-1$, i , $i+1$. Действительно, для этой окружности выполняются соотношения

$$\begin{aligned}
|\vec{R}_{i-1}| &= |\vec{R}_i| = |\vec{R}_{i+1}|, \\
\vec{R}_i - \vec{R}_{i-1} &= \vec{S}_i, \\
\vec{R}_{i+1} - \vec{R}_i &= \vec{S}_{i+1}.
\end{aligned}
\tag{25}$$

Возведя два последних соотношения в квадрат и подставив в них первые два соотношения, получим

$$\begin{aligned}
(\vec{R}_i, \vec{S}_i) &= \frac{\vec{S}_i^2}{2}, \\
(\vec{R}_i, \vec{S}_{i+1}) &= -\frac{\vec{S}_{i+1}^2}{2}.
\end{aligned}
\tag{26}$$

Запишем выражение для производной сплайна В в точке i (II) в виде

$$\vec{p}_i = \frac{S_i S_{i+1}}{S_i + S_{i+1}} \left(\frac{\vec{S}_i}{S_i^2} + \frac{\vec{S}_{i+1}}{S_{i+1}^2} \right).
\tag{27}$$

Используя (26), (27) получим

$$(\vec{R}_i, \vec{p}_i) = \frac{S_i S_{i+1}}{S_i + S_{i+1}} \left(\frac{(\vec{R}_i, \vec{S}_i)}{S_i^2} + \frac{(\vec{R}_i, \vec{S}_{i+1})}{S_{i+1}^2} \right) = 0,$$

откуда следует исходное предположение. Таким образом, сплайн В более выпуклый на длинном отрезке, чем на соседнем коротком.

Производная сплайна В в точке i образует одинаковые углы с хордами $\vec{S}_i$, $\vec{S}_{i+1}$, так как скалярное произведение вектора производной (I3) с вектором $\vec{S}_i/S_i$ равно скалярному произведению вектора производной с $\vec{S}_{i+1}/S_{i+1}$. На этом основании можно говорить, что сплайн В "равномерно" выпуклый, и что он наиболее близок ломаной, соединяющей узловые точки.

Эти свойства останутся справедливыми и для нормализованных сплайнов.

Для ненормализованного сплайна В можно доказать еще одно свойство, иллюстрирующее его устойчивость при любом расположении узловых точек. Свойство заключается в том, что угол между

касательной в любой точке сплайна на промежутке $i, i+1$ и хордой, соединяющей точки $i, i+1$, меньше прямого. Из этого следует, что в сплайне В заведомо отсутствуют резкие изгибы и петли, при существовании которых найдется точка с отрицательной проекцией касательной на хорду.

Для доказательства повернем сплайн так, чтобы хорда была параллельна оси абсцисс. Покажем, что $X'_{\delta i}(t) > 0$ при $0 < t < \sqrt{S_i}$. Дифференцируя (12), подставив $\tau = \frac{t}{\sqrt{S_i}}$ и учитывая, что $\Delta x_i = S_i$, получим

$$\begin{aligned} \frac{X'_{\delta i}(t)}{\sqrt{S_i}} &= 1 + 2(3 - 2x'_{i-1} - x'_i)\tau + 3(x'_{i-1} + x'_i - 2)\tau^2 = \\ &= \tau(1-\tau)(6 - 4x'_{i-1} - 2x'_i) + x'_i\tau + 1 - x'_{i-1}\tau^2 > 0, \end{aligned}$$

так как $0 \leq x'_{i-1} \leq 1$, $0 \leq x'_i \leq 1$, $0 < \tau < 1$.

7. Вычисление сплайнов на ЭВМ

Из всех видов полученных сплайнов для системы графического вывода необходимо выбрать такие, которые были бы наиболее удобны для построения изображений различных классов. В системе СМОГ для ЕС ЭВМ [6] применены следующие 4 вида сплайнов: сплайн 1 - ненормализованный сплайн В, сплайн 2 - нормализованный сплайн В, сплайн 3 - нормализованный сплайн Б, сплайн 4 - простейший сплайн. Краевые условия в сплайнах 1, 3, 4 - вида (18)-(19) (равенство - нулю кривизны в конечных точках), а в сплайне 2 - вида (21) (постоянство кривизны на конечных отрезках сплайна).

Сплайн 1 удобен для изображения графиков, с помощью сплайна 2 можно изображать окружности, овалы и их дуги, сплайном 3 можно изобразить траекторию движения тела по неравномерным точкам, а сплайном 4 можно без искажения нарисовать проекции пространственной кривой. В сомнительных случаях рекомендуется применять наиболее устойчивый сплайн 1.

Процедура СПЛАЙН вычисляет 4 вида сплайнов в двух

вариантах - замкнутом и незамкнутом.

Параметры процедуры (она написана на Алголе - 68):

замкн - при значении истина задает замкнутый сплайн, при значении ложь - незамкнутый;

номер - задает номер сплайна;

N - количество узловых точек;

X, Y - координаты узловых точек.

Процедура состоит из трех последовательных частей. В первой части изображается отрезок сплайна между точками 1, 2 (для замкнутого и незамкнутого сплайнов по разным формулам). Во второй части в цикле изображаются отрезки между точками 2 и 3, 3 и 4, ..., $N-2$ и $N-1$. В третьей части изображается отрезок $N-1, N$ (для замкнутого сплайна два отрезка: $N-1, N$ и $N, 1$).

ПОЛИНОМ изображает отрезок сплайна. ПРОИЗВОДНАЯ вычисляет ненормализованные производные в очередной точке сплайна.

НОРМАЛИЗАЦИЯ их нормализует, если номер сплайна 2 или 3. Крайние производные вычисляют ЛЕВКРАЙ и ПРАВКРАЙ.

ОТРЕЗОК 1 2, ОТРЕЗОК 2 3 и т.д. вычисляют разности координат и длины хорд на соответствующем отрезке сплайна.

проц СПЛАЙН = (лог замки, цел номер, N , имя [] вещ X, Y):
начало ОПИСАНИЯ;

первая часть:

ОТРЕЗОК 12;

если замки то

ОТРЕЗОК $N1$; ПРОИЗВОДНАЯ;

ОТРЕЗОК 23 ; ПРОИЗВОДНАЯ;

НОРМАЛИЗАЦИЯ

иначе

ОТРЕЗОК 23; ПРОИЗВОДНАЯ;

ЛЕВКРАЙ

конец ;

ПОЛИНОМ;

вторая часть:

для i от 3 до $N-1$ цикл

ОТРЕЗОК i ; ПРОИЗВОДНАЯ;

НОРМАЛИЗАЦИЯ; ПОЛИНОМ

конец;

третья часть:

если замки то

ОТРЕЗОК $N18$; ПРОИЗВОДНАЯ;

НОРМАЛИЗАЦИЯ; ПОЛИНОМ;
ОТРЕЗОК 128 ; НОРМАЛИЗАЦИЯ
ПРАВКРАЙ илсе;

иначе

ПОЛИНОМ

конец

Л и т е р а т у р а

1. Алберг Дж., Нильсон Э., Уолш Дж. Теория сплайнов и её приложения. М., "Мир", 1972 .
2. Завьялов Д.С. Интерполирование кубическими многозвеньниками. "Вычислительные системы", № 38, Новосибирск, 1970 .
3. Manning I.R. Continuity Conditions for Spline Curves. "Computer Journal", v.17, N2, 1974.
4. Баяковский Ю.М., Лебедева Т.С., Мамаева А.И. ГРАФОР: комплекс графических программ на ФОРТРАНе. Вып.3., М., препринт ИПМ АН СССР, № 88, 1974 .
5. ГРАФИК - система математического обеспечения ЭВМ и графопостроителей. Под ред. Е.Л.Ощенко. Кишинев, "Штиинца", 1975 .
6. Костяк Ю.Л. Система математического обеспечения графического вывода для ЕС ЭВМ, Томск, Изд-во Томского университета, 1977 .

ЯЗЫК ПРЕДСТАВЛЕНИЯ РИСУНКОВ В СИСТЕМЕ СМОГ

Ю.Л.Костык, А.Л.Фуко

I. Введение

Системы машинной графики можно разделить на два класса: системы с входным графическим языком и системы с процедурным языком, реализованные как набор подпрограмм универсального языка программирования (Алгола-60, Фортрана, ПЛ/I). Хотя реализация систем с входным графическим языком более трудоемка, чем систем в виде набора графических подпрограмм (необходим компилятор со специальным языком), их применение имеет определенные преимущества при формировании и наращивании большого набора стандартных графических рисунков.

Если система, реализованная в виде набора подпрограмм, обладает достаточно широкими и универсальными графическими функциями, то с её помощью также можно наращивать набор графических фрагментов (в виде подпрограмм, их изображающих), однако этот способ слишком громоздкий.

Система СМОГ [1], являющаяся набором процедур, содержит достаточно возможностей для формирования и наращивания графических подпрограмм изображения отдельных рисунков, в то же время в ней имеются средства, позволяющие описывать эти рисунки на специальном графическом языке. Сопрежение процедурного языка СМОГ с языком представления рисунка производится оператором

CALL FRAG (PIC, M, T);

где в строке *PIC* записан текст фрагмента-рисунка, который изображается в системе координат, имеющей имя *M*, с использованием типа линии *T*. Программным изменением системы координат

и типа линии фрагмент-рисунок можно изобразить в любом виде и месте на поле чертежа, поэтому сам графический язык может быть при этом достаточно простым.

Язык представления рисунка (ЯПР) в системе СМОГ намного проще входных графических языков, применяемых в других системах (которые обычно являются расширением Фортрана или Алгол-60), однако его сопряжение с процедурным языком создает достаточно гибкую и универсальную систему. В ЯПР используются операторы координатных преобразований, операторы цикла, линии и имена других фрагментов-рисунков, а также тексты. Особенно от ЯПР является то, что все значения в языке записываются константами (переменные отсутствуют), поэтому фрагмент-рисунок представляет собой устойчивое сочетание линий.

Возможности системы СМОГ позволяют легко расширять имеющуюся библиотеку стандартных фрагментов-рисунков (описанных на ЯПР), ориентируя тем самым систему СМОГ на определенную область приложения.

2. Язык представления рисунка

Текст фрагмента-рисунка записывается в виде строки символов. Его структуру можно описать следующим образом:

фрагмент-рисунок: последовательность фрагментов,
символ * ,
последовательность фрагментов: фрагмент ;
фрагмент, точка с запятой,
последовательность фрагментов.
фрагмент: имя стандартного рисунка;
текстовый фрагмент;
линейный фрагмент;
оператор, фрагмент;
символ (, последовательность фрагментов,
символ).

Фрагмент-рисунок состоит из фрагментов, разделенных точкой с запятой, и заканчивается звездочкой. Фрагмент может быть тек-

отом, например:

TEXT (0,0) "Рис. 1"

либо линейным фрагментом, в котором перечисляются координаты узловых точек, например:

(0,0 / 10,0 / 10,10 / 0,10)

Воздействием координатного оператора можно видоизменить фрагмент, а оператором цикла — изобразить его несколько раз в различных видах. В обоих случаях получится новый фрагмент, который можно подвергнуть новому преобразованию и т.д.

Оператор цикла имеет формат

DO (N, OPER)

где *N* — число повторений, *OPER* — набор координатных операторов, подвергающих фрагмент преобразованию в цикле.

Координатные операторы:

ROT (W) — поворот на *W* градусов;

ANGLE (F) — изменение угла раскрытия осей на *F* градусов;

SHIFT (X,Y) — сдвиг на *X,Y* мм;

SCALE (KX,KY) — изменение масштаба в *KX,KY* раз.

Например, фрагмент-рисунок

*DO (5, SHIFT (15,0))(0,0/10,0/10,10/0,10) **

изобразит рис. I.



Рис. I. Пример фрагмента-рисунка

Оператор типа позволяет задавать характеристики типа линии, которой изображается фрагмент. Например, оператор

TYPE (1,0,2)

задает замкнутую ломаную штрихпунктирную линию.

Некоторые характеристики в типе линии могут отсутствовать, тогда они берутся либо из другого оператора типа линии, действующего на данный, либо (если нет такого оператора) из это-

рого параметра подпрограммы *FRAG*.

Оператор трафарета задает характеристики шрифта, которым изображаются тексты внутри фрагмента. Например:

TRAF (18/01)

Этот оператор задает шрифт высотой 8 мм, нормальной ширины, с наклоном.

Если трафарет в фрагменте-рисунке не задан, то по умолчанию применяется нулевой трафарет.

Фрагмент может быть именем стандартного рисунка, помещенного в библиотеку. Имя всегда начинается с символа # и состоит до 7 букв или цифр.

3. Реализация языка

Применение языка представления рисунка в системе СМОГ предполагает его интерпретацию-анализ и выполнение фрагмента рисунка производится во время работы программы *FRAG*. В данном случае интерпретация (по сравнению с компиляцией) практически не замедляет построение изображений, так как основную трудоемкость составляет расчет элементарных графических объектов (линий и символов).

Основная структурная единица в ЯПР - фрагмент. Всего может быть 11 видов "начал" фрагмента, начинающихся с символов:

- "+" , "-" , "." , "(" цифра" - линейный фрагмент;
- "TE" - текстовой фрагмент;
- "*" - имя стандартного рисунка ;
- "T(", "TV" - оператор типа линии ;
- "TR" - оператор трафарета;
- "D" - оператор цикла ;
- "SH" - оператор сдвига;
- "SC" - оператор масштаба;
- "R" - оператор поворота ;
- "A" - оператор изменения угла раскрытия осей ;
- "(" - последовательность фрагментов в скобках (последние скобки должен начинаться фрагмент).

Таким образом, для определения соответствующего "начала"

достаточно проанализировать 1-2 символа.

После определения вида "начала" необходимо анализировать соответствующую конструкцию, после которой может быть всего несколько видов символов: " ; ", ") ", " * ". После точки с запятой должен опять начинаться фрагмент, а скобка и звездочка завершают фрагмент.

Текст анализируемого фрагмента-рисунка может быть либо параметром процедуры *FRAG*, либо может находиться в буфере для считанных с диска стандартных рисунков. Информация в буфере располагается в следующем виде (см. рис.2): в начале буфера-строки

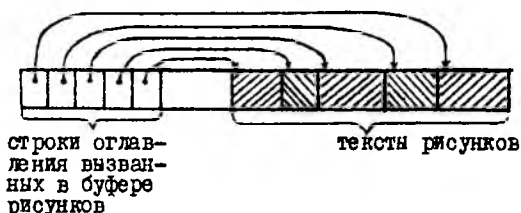


Рис.2. Буфер стандартных рисунков

оглавления вызванных стандартных рисунков, в конце буфера - тексты рисунков, а между ними свободное место.

Если для очередного рисунка и строки в оглавлении не хватает места, буфер полностью очищается. Такой алгоритм заполнения приводит к тому, что в буфере находятся те рисунки, к которым было наиболее позднее обращение.

При выполнении цикла необходим многократный просмотр, анализ и выполнение части текста фрагмента-рисунка, включающего совокупность координатных операторов, закрывающую скобку, фрагмент.

В укрупненном виде процедура *FRAG* записывается (на языке Алгол-68):

```

ПРОЦ FRAG (СТРОК, PIC, M, T):
  НАЧАЛО ПРОЦ;
    СТРОК ТЕКСТЫ: = : ЦЕЛ НАЧ: = 1, i: = 1;
    ФРАГМЕНТ (PIC);
  Е: ЭПИЛОГ
  КОНЕЦ
  
```


Переменная тексия содержит имя анализируемого рисунка. Фрагменту-рисунку *PIC* условно приписывается пустое имя. Переменная *нач* указывает на начало рисунка в буфере или просматриваемой строке, а *i* — номер очередного просматриваемого символа. Рекурсивная процедура ФРАГМЕНТ анализирует фрагмент-рисунок, содержащийся в строке (парамetre процедуры), начиная с *i*-го символа.

проц ФРАГМЕНТ = (строк *P*):

начало

пока НЕ КОНЕЦ ФРАГМЕНТА цикл

если ИМЯ СТАНДАРТИСГО РИСУНКА

то строк *имяс* := тексия;

АНАЛИЗ ИМЕНИ;

цел *K* := *i* - нач;

ПОИСК В БУФЕРЕ; *i* := *i*;

ФРАГМЕНТ (БУФЕР);

тексия := *имяс*;

ПОИСК В БУФЕРЕ; *i* := нач + *K*;

инес ТЕКСТ то РИСОВАНИЕ ТЕКСТА

инес ЛИНИЯ то РИСОВАНИЕ ЛИНИИ

инес ПОСЛЕДОВАТЕЛЬНОСТЬ ФРАГМЕНТОВ В СКОБКАХ

то *i* плюс 1; ФРАГМЕНТ (*P*);

если *P*[*i*-1] ≠ ')' то ОШИБКА илсе

инес ЦИКЛ то цел *K, n*;

АНАЛИЗ ЦИКЛА; НАЧАЛО ЦИКЛА;

если *n* < 1 то ОШИБКА илсе;

до *n* цикл

ФРАГМЕНТ (*P*); *i* := *K*;

АНАЛИЗ И ВЫПОЛНЕНИЕ КООРД ОПЕРАТОРОВ

лкий;

КОНЕЦ ЦИКЛА

инес СДВИГ то ВЫПОЛНЕНИЕ СДВИГА;

ФРАГМЕНТ (*P*); КОНЕЦ СДВИГА

инес ПОВОРОТ то ВЫПОЛНЕНИЕ ПОВОРОТА;

ФРАГМЕНТ (*P*); КОНЕЦ ПОВОРОТА

инес МАСШТАБ то ВЫПОЛНЕНИЕ МАСШТАБА;

ФРАГМЕНТ (*P*); КОНЕЦ МАСШТАБА

инес УГОЛ то ВЫПОЛНЕНИЕ УГЛА РАСКРЫВА;
 ФРАГМЕНТ (P) ; КОНЕЦ УГЛА
инес ТИП то ВЫПОЛНЕНИЕ ТИПА ;
 ФРАГМЕНТ (P) ; КОНЕЦ ТИПА
инес ТРАФАРЕТ то ВЫПОЛНЕНИЕ ТРАФАРЕТА ;
 ФРАГМЕНТ (P) ; КОНЕЦ ТРАФАРЕТА
инес P [i] = ' ; ' то i плюс 1
иначе ОШИБКА
илсе
лкнц
 i плюс 1
конец

Таким образом, выполнение фрагмента рисунка осуществляется за один просмотр, за исключением циклов, в которых количество просмотров равно количеству повторений.

Выполнение выделенных конструкций осуществляется путем обращений к подпрограммам 1-2 уровней СМОГ. Например, ВЫПОЛНЕНИЕ ПОВОРОТА включает операторы *NAMSC* и *ROT*, а КОНЕЦ ПОВОРОТА - оператор *DEISC*.

Работа с буфером стандартных рисунков происходит, лишь если в тексте фрагмента-рисунка есть имена стандартных рисунков, поэтому, при отсутствии последних, файл библиотеки стандартных рисунков не нужен, и он даже не открывается.

Выполнение процедуры ОШИБКА происходит при обнаружении в тексте фрагмента-рисунка синтаксической ошибки. При этом производится печать диагностического сообщения и окончание работы процедуры *FRAG* (выход на метку *E*).

Процедура *FRAG* в системе СМОГ реализована на языке АС-СЕМБЛЕР, поэтому рекурсивный вызов процедуры ФРАГМЕНТ реализован с использованием явно описанного в программе стека, в котором запоминаются необходимые промежуточные переменные.

4. Библиотека стандартных рисунков

Библиотека реализована в виде файла прямого доступа с фиксированной длиной физических записей. Логически она состоит из

оглавления и текстов рисунков. Строка оглавления имеет длину 14 байтов, тексты рисунков состоят из целого числа записей по 14 байтов, таким образом, вся библиотека состоит из логических записей длиной по 14 байтов. В первой записи содержится служебная информация о состоянии библиотеки. При первом обращении к библиотеке из процедуры *FRAG* эта информация считывается, и процедура настраивается на работу с данной библиотекой. Оглавление заполняется от начала (со 2-й записи), а тексты рисунков — с конца библиотеки (см. рис.3).

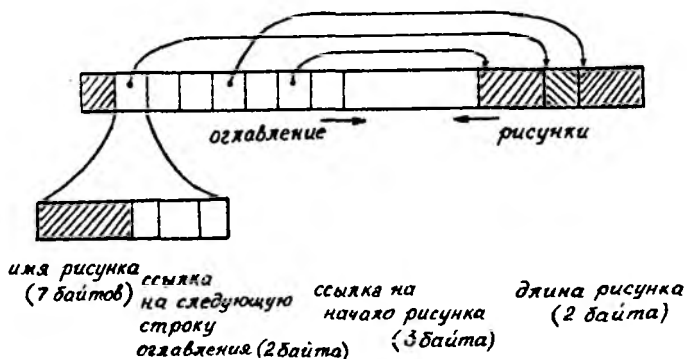


Рис.3. Структура библиотеки рисунков

В библиотеке применяется рандомизированный поиск, обеспечивающий нахождение адреса расположения рисунка в большинстве случаев за одно чтение оглавления. Алгоритм состоит в следующем. Имя рисунка делится как положительное двоичное число на специально подобранную константу p (простое число), и остаток (увеличенный на 1) считается номером строки оглавления этого рисунка. Так как возможны случаи, что из нескольких разных имен получатся одинаковые номера, то второе имя (с одинаковым номером) записывается в конец оглавления (которое при этом удлиняется), а в первой строке делается ссылка. В этом случае поиск удлиняется. Впрочем, такие случаи сравнительно редки, особенно при слабой заполненности библиотеки. Значение p подбирается в зависимости от размера библиотеки — количества рисунков, на которое она в среднем рассчитана. Например, при библиотеке на 125 рисунков $p = 127$. Число p записано в 1-й записи библиоте-

ки, так как размеры библиотеки могут меняться (при генерации системы СМОГ).

Служебная программа *SMOGLSF*, изменяющая состав библиотеки и выполняющая ряд других сервисных функций, включает алгоритм анализа фрагментов-рисунков, помещаемых в библиотеку. Этот алгоритм отличается от алгоритма процедуры *FRAG* лишь отсутствием обращений к процедурам I-2 уровней системы СМОГ.

Л и т е р а т у р а

- I. Костяк Ю.Л. Система математического обеспечения графического вывода для ЕС ЭВМ. Томск, Изд-во Томского университета, 1977.

УНИВЕРСАЛЬНАЯ ПРОЦЕДУРА РИСОВАНИЯ ГРАФИКОВ

Ю.Л.Костюк

I. Введение

Системы графического вывода состоят, как правило, из базиса и ряда проблемно-ориентированных надстроек. Базис включает основные графические функции, с помощью которых в принципе можно построить любое изображение. Однако при создании изображений специальных классов применение только средств базиса оказывается трудоемким, для облегчения программирования и увеличения эффективности использования системы необходимы специализированные надстройки.

Важнейшей надстройкой графической системы, применяемой в автоматизации научных и инженерных расчетов, является изображение графиков. Наиболее употребителен способ рисования графиков в виде функции $Y(X)$ на прямоугольной координатной сетке. При этом задача изображения графика разделяется на две - изображение сетки и изображение собственно графика на сетке. Вторая задача состоит в проведении гладкой или ломаной линии и решается средствами базиса графической системы.

Наибольшую сложность представляет изображение сетки для графика. Даже в случае равномерного или логарифмического масштаба по осям координат алгоритм автоматического разбиения (когда заданы лишь границы изменения аргумента X и функции Y) нетривиален, так как эти границы предварительно необходимо округлить, чтобы разбиение получилось приемлемым. Применение же произвольного функционального масштаба (особенно удобного, например, при про-

верке совпадения экспериментальной зависимости о теоретической) делает эту задачу весьма непростой.

Пусть задана монотонно возрастающая функция $S(u)$ и крайние значения $u_{\min}, u_{\max}$. Последовательность значений $\{u_0, u_1, \dots, u_n\}$ такая, что $u_0 \leq u_{\min} < u_1, u_{n-1} \leq u_{\max} < u_n$ представляет разбиение аргумента u . Требуется, чтобы числа в последовательности были "красивыми" (удобными для их рассмотрения), а расстояния между штрихами разбиения $S(u_{i+1}) - S(u_i)$ были ни слишком малы, ни слишком велики.

Предложенный в работе [1] эвристический алгоритм разбиения требует, чтобы значения u_0 и u_n были заданы. Этот алгоритм строит одно из возможных разбиений, которое не всегда может быть наилучшим.

В [2] определены красивые последовательности чисел и на их основе получены два алгоритма красивого разбиения функциональных шкал. Один из этих алгоритмов (глобальный) позволяет получать наилучшие разбиения при любых непрерывных масштабных функциях. Он применен в системе СИОГ [3] для разметки координатных сеток. В статье описывается реализация процедуры изображения сетки с применением глобального алгоритма.

2. Глобальный алгоритм красивого разбиения

Рассмотрим сначала некоторые определения и результаты из [2]. Там они получены для произвольной системы счисления, здесь мы их дадим для десятичной системы.

Определение 1. Возрастающая последовательность $\{u_0, u_1, \dots, u_n\}$ называется красивой, если:

а) разность любых двух соседних чисел $\Delta_i = u_i - u_{i-1}$ принадлежит множеству H красивых разностей;

б) любое число u_i кратно своей левой Δ_i и правой Δ_{i+1} разностям (для u_0 и u_n рассматриваются только правая Δ_1 и левая Δ_n разности);

в) элементы множества H являются числами вида $k \cdot 10^p$, где p - целое, а $k = 1, 2$ или 5 .

Теорема. Пусть $\{u_0, u_1\}$ и $\{u_{n-1}, u_n\}$ - такие двухэлементные красивые последовательности, что $u_1 < u_{n-1}$. Тогда суще ст-

вует красивая последовательность $\{v_0, v_1, \dots, v_n\}$ с $v_0 = u_0$, $v_n = u_n$, такая, что любые красивые разности в ней находятся в пределах: $u_n - u_{n-1} \geq v_i - v_{i-1} > u_1 - u_0$ либо $u_n - u_{n-1} \leq v_i - v_{i-1} < u_1 - u_0$.

Свойство 1. Количество значащих цифр в любом элементе красивой последовательности не превосходит количества цифр в $\frac{u}{10^p}$, где p - показатель степени в левой или правой разности h числа u ($h = k \cdot 10^p$).

Свойство 2. Если красивая последовательность $\{u_0, u_1, \dots, u_n\}$ включает интервал $[a, b]$, т.е. $u_0 \leq a$, $b \leq u_n$ и хотя бы одно число $u_i \in [a, b]$, то все самые красивые числа из интервала $[a, b]$ (т.е. такие, что при их представлении в виде $x = \ell \cdot 10^p$; ℓ, p - целые, максимальное p больше, чем у остальных чисел) принадлежат последовательности.

Определение 2. Пусть задана монотонно возрастающая функция $S(u)$, минимальное $u_{\min}$ и максимальное $u_{\max}$ значения её аргумента и минимальное разбиение шкалы S . Красивым разбиением функциональной шкалы называется такая красивая последовательность $\{u_0, u_1, \dots, u_n\}$, для которой выполняются условия:

- 1) $S(u_i) - S(u_{i-1}) \geq S$, $i = 1, \dots, n$;
- 2) $u_0 \leq u_{\min} < u_1$, $u_{n-1} < u_{\max} \leq u_n$;
- 3) n максимально.

Если к тому же задана геометрическая длина шкалы L , то в первом условии S необходимо заменить на $S \cdot \frac{L}{S(u_n) - S(u_0)}$.

Рассмотрим кратко работу глобального алгоритма.

Алгоритм состоит из трех этапов. На первом этапе находим два таких числа u_a, u_b с красивой разностью между ними, что $u_a \leq u_{\min}$, $u_{\max} \leq u_b$. Если $u_{\min} \geq 0$, то число $u_a = 0$, если $u_{\max} \leq 0$, то $u_b = 0$. Если же $u_{\min} \leq 0 < u_{\max}$, то находим две пары таких чисел: $u_a, 0$ и $0, u_b$.

Полученные u_a, u_b могут сказаться таким, что $S(u_a)$ или $S(u_b)$ могут не существовать, хотя $u_{\min}$ и $u_{\max}$ находятся внутри области определения функции $S(u)$. В этом случае необходим второй этап, на котором левый и (или) правый интервалы последовательным делением на 2 или 5 частей (и отбрасыванием тех полученных интервалов, которые находятся вне диапазона $u_{\min}, u_{\max}$) не будут получены крайние значения u_a, u_b , находя-

щиеся внутри области определения $S(u)$.

Наконец, на третьем этапе полученные интервалы подвергаются дальнейшему делению на 2 или 5 частей до тех пор, пока ни один из интервалов нельзя будет разделить из-за нарушения условия I.

Нарушение этого условия может произойти лишь при выполнении 2-го этапа (при неудачно заданных условиях), однако лучшее разбиение при этом в принципе невозможно.

3. Программная реализация алгоритма разбиения шкалы

Функцию $S(u)$ необходимо задавать (в виде процедуры) таким образом, чтобы вне области определения процедура вырабатывала очень большое число (например, 10^{37} в системе СМОГ).

Процедура, реализующая глобальный алгоритм, формирует список точек разбиения шкалы. Каждая точка представляет собой структуру (в Алголе-68):

вид точка = струк (цел k, l, n, m , вещ u , имя точка след),

где $k = 1, 2$ или 5 ; n - порядок правой красивой разности (разность $k = k \cdot 10^n$; u - значение точки ($u = l \cdot 10^n$); m - количество значащих цифр в данной точке; след - ссылка на следующую точку.

Параметры процедуры:

- u_{min}, u_{max} - исходные крайние значения аргумента u ;
- S - минимальное разбиение шкалы (длина L шкалы полагается равной 1);
- SF - масштабная функция, вне области определения вырабатывает максзначение;
- PH, PK - указатели на начальный и конечный элементы последовательности чисел, полученной алгоритмом.

проц ГЛОБРАЗБ = (вещ u_{min}, u_{max} , S , проц (вещ) вещ SF ,
имя точка PH, PK):

начало вещ $S1, SA$, имя точка PH1, PK1, P, PI;

НАЧАЛЬНАЯ ПОСЛЕДОВАТЕЛЬНОСТЬ; ϕ из двух или трех точек,
на которые ссылаются указатели PH, PH1, PK1, PK ϕ

пока $SF(u \text{ от } PH) = \text{максзначение}$ цикл
 ЛЕВРАЗБИЕНИЕ ϕ на 2 или 5 частей ϕ лици;
пока $SF(u \text{ от } PK) = \text{максзначение}$ цикл
 ПРАВРАЗБИЕНИЕ ϕ на 2 или 5 частей ϕ лици;
лог билоразбиение: = ИСТИНА ;
пока билоразбиение цикл
 $S1 := S / (SF(u \text{ от } PK) - (SA := SF(u \text{ от } PH)))$;
 ϕ вычисление минимального разбиения с учетом
 u, u_1 ; запоминание значения функции от u, ϕ
 билоразбиение : = ложь, $R := PH$;
пока $R \neq PK$ цикл
 $PI := \text{след от } R$;
 ШАГ РАЗБИЕНИЯ; ϕ на 2 или 5 частей ϕ
 $R := PI$
лици
лици
конец

Элементы последовательности процедура генерирует с помощью глобального генератора.

4. Изображение координатной сетки

Используя глобальный алгоритм разбиения шкалы, можно создать универсальную процедуру изображения сетки. Вызов такой процедуры в системе СМОГ:

CALL GRID (FOR, B, TEX, XMIN, YMIN, XMAX, YMAX, FX, FY, LX, LY);

Параметры процедуры.

FOR – номер задаваемого формата, на котором рисуется сетка; если значение параметра '0', то формат читается заданным до вызова процедуры, и сетка рисуется внутри текущей системы координат с длинами осей **LX, LY**.

B – параметры оформления сетки.

TEX – тексты для надписей.

XMIN, YMIN, XMAX, YMAX – диапазон изменения абсцисс и

ординат графика, который будет изображаться на данной сетке. После работы программы эти значения округляются до крайних элементов красивых разбиений.

FX, FY - процедуры масштабных функций.

LX, LY - длины осей сетки. Задаются программистом (если $FOR = '0'$) либо вычисляются программой.

Рассмотрим программу **GRID** в укрупненном виде (на Алгоде-68).

проц GRID = (строк FOR, B, TEX имя вещ $XMIN, YMIN, XMAX, YMAX$,

проц (вещ) вещ FX, FY , имя вещ LX, LY):

начало цел NX, NY , вещ SX, SY :

если АНАЛИЗ FOR ; ФОРМАТ ВЕРЕН

то $FORM(FOR)$; $\neq$ задание формата $\neq$

ЧТЕНИЕ ПАРАМЕТРОВ СЕТКИ ДЛЯ ФОРМАТА FOR ;

ЗАДАНИЕ ТЕКУЩЕЙ СИСТЕМЫ КООРДИНАТ ДЛЯ СЕТКИ

иначе $FOR \neq '0'$, то ОШИБКА илсе ;

АНАЛИЗ В И ЗАДАНИЕ ТРАФАРЕТА ;

АНАЛИЗ В И ВЫБОР ВАРИАНТА РАЗБИЕНИЯ СЕТКИ NX ;

АНАЛИЗ В И ВЫБОР ВАРИАНТА РАЗБИЕНИЯ СЕТКИ NY ;

АНАЛИЗ В И ВЫБОР РАЗБИЕНИЯ ШКАЛЫ SX ;

АНАЛИЗ В И ВЫБОР РАЗБИЕНИЯ ШКАЛЫ SY ;

АНАЛИЗ В И ВЫЧИСЛЕНИЕ ТИПА ЛИНИИ С УКАЗАННЫМ ПЕРЛОМ ;

ИЗОБРАЖЕНИЕ НАДПИСЕЙ TEX ;

имя число PH, PK ;

если $XMIN, XMAX, YMIN, YMAX$ ВЕРНЫ то

ГЛОБАЛЬ($XMIN, XMAX, SX/LX, FX, PH, PK$);

$XMIN := \text{" от } PH$; $XMAX := \text{" от } PK$;

выб NX из РАЗМЕТКА СЕТКИ X ВАРИАНТ 0 ;

РАЗМЕТКА СЕТКИ X ВАРИАНТ 1,

РАЗМЕТКА СЕТКИ X ВАРИАНТ 2,

РАЗМЕТКА СЕТКИ X ВАРИАНТ 3 оув ;

ГЛОБАЛЬ($YMIN, YMAX, SY/LY, FY, PH, PK$);

$YMIN := \text{" от } PH$, $YMAX := \text{" от } PK$;

выб NY из РАЗМЕТКА СЕТКИ Y ВАРИАНТ 0,

РАЗМЕТКА СЕТКИ Y ВАРИАНТ 1,

РАЗМЕТКА СЕТКИ Y ВАРИАНТ 2,

РАЗМЕТКА СЕТКИ Y ВАРИАНТ 3 оув

иначе ОШИБКА илисе ;
СБРОС ТРАФАРЕТА

конец

5. Изображение графиков

После работы процедуры *GRID* текущая система координат совмещена началом с левым нижним углом сетки. Процедура *GRAPH* рисует на сетке график, заданный массивами узловых точек X, Y (размерностью от 1 до N , типом линии T . Обращение к ней

CALL GRAPH (T,N,X,Y,XMIN,YMIN,XMAX,YMAX,FX,FY,LX,LY);

Остальные параметры должны иметь точно такое же значение, что и в процедуре *GRID*, тогда график разместится точно на сетке.

В некоторых системах графического вывода графики задаются в функциональном виде, но этот способ обычно менее удобный (и менее универсальный), в то же время, имея, например, функцию $Y(X)$, можно легко получить массивы координат точек X и Y .

Многократным употреблением процедуры *GRAPH* можно нарисовать несколько графиков на одной координатной сетке.

Процедура на Алголе-68:

```
проц GRAPH = (строка  $T$ , цел  $N$ , имя [ ] вещ  $X, Y$ ,  
                  вещ  $XMIN, YMIN, XMAX, YMAX$ ,  
                  проц (вещ) вещ  $FX, FY$ , вещ  $LX, LY$ ):  
начало вещ  $SX := LX / (FX(XMAX) - FX(XMIN))$ ,  
          вещ  $SY := LY / (FY(YMAX) - FY(YMIN))$ ;  
  т  $BE(T)$ ;  $\nless$  задание типа линии графика  $\nless$   
  для  $i$  до  $N$  цикл  
     $POINT(FX(X[i]) * SX, FY(Y[i]) * SY)$   
  кцикл ;  
   $\nless$  задание координат линии  $\nless$   
   $DRAW('*')$   $\nless$  изображение графика в текущей сис-  
    теме координат, т.е. на сетке  $\nless$ 
```

конец

Пользоваться процедурами *GRID* и *GRAPH* хотя и не сложно, но довольно громоздко. Для изображения одного графика на сетке в системе СМОГ предусмотрена процедура *GRASP*, используя которую, программист должен задавать минимум информации:

```

проц GRASP = (строки FOR, B, TEX, T, цел N, имя [ ] вещ X, Y,
               проц (вещ) вещ FX, FY):
начало вещ XMIN, YMIN, XMAX, YMAX, LX, LY;
  MINMA (N, X, XMIN, XMAX);      ¢ вычисление гра-
                                   ниц массива X ¢
  MINMA (N, Y, YMIN, YMAX);      ¢ вычисление гра-
                                   ниц массива Y ¢

  GRID (FOR, B, TEX, XMIN, YMIN, XMAX, YMAX, FX, FY, LX, LY);
  GRAPH (T, N, X, Y, XMIN, YMIN, XMAX, YMAX, FX, FY, LX, LY)

```

конец

Л и т е р а т у р а

1. Борисов С.И. Автоматизация расчета шкал номограмм. "Номографический сборник", № 2. М., ВЦ АН СССР, 1964.
2. Костяк Ю.Л. Красивые разбиения функциональных шкал. Сб. "Прикладная математика и кибернетика", Вып. I. Томск, Изд-во Томского университета, 1976.
3. Костяк Ю.Л. Система математического обеспечения графического вывода для ЕС ЭЕМ. Томск, Изд-во Томского университета, 1977.

УДК 007.5 : 681.5

Автоматизированная система регистрации, хранения и обработки экспериментальных данных "САФРА" - общее описание.

Г л а д к и х Б.А. Сб.: Автоматизация эксперимента и машинная графика. Томск, Изд-во ТГУ, 1977, 3-21.

Система САФРА представляет собой аппаратно-программный комплекс, предназначенный для комплексной автоматизации экспериментов - от регистрации электрических сигналов до получения готовых результатов в виде таблиц, графиков функций, диаграмм. Технические средства системы состоят из аппаратуры сопряжения с экспериментом, выполненной в стандарте КАМАК, управляющей мини-ЭВМ со структурой команд, аналогичной РДР-8 (Электроника - 100, Саратов), обрабатывающей ЭВМ Единой системы и устройств отображения. Математическое обеспечение включает подсистему управления экспериментом, подсистему регистрации и хранения информации, подсистему обработки информации и подсистему отображения.

В статье дается общее описание системы с точки зрения пользователя. Реализация отдельных подсистем более подробно описана в других статьях сборника.

Библ. 9, ил.3, табл.2.

УДК 007.5 : 681.321.0

Представление данных в системе САФРА. Г л а д к и х Б.А., М а т у ш е в с к и й В.В. Сб.: Автоматизация эксперимента и машинная графика. Томск, Изд-во ТГУ, 1977, 22-30.

Описываются принятые в системе САФРА представления структур данных - логические и физические. Логическое представление ориентировано на пользователей, занимающихся обработкой данных, оно стандартизовано и абстрагировано от технических деталей представления, обращение к данным производится просто по именам. Физические представления связаны с процессами передачи данных и их регистрации на носителях. Для связи между логическими и физическими представлениями служат словари, хранящиеся в ме-

те с данными.

Библ.3, ил.1, табл.2.

УДК 0075 : 681.3.069

Математическое обеспечение для управления экспериментом в системе САФРА. Золотенков В.В., Осипова Л.Б. Сб. : Автоматизация эксперимента и машинная графика. Томск, Изд-во ТГУ, 1977, 31-42.

Описывается реализация подсистемы управления экспериментом и частично (в той части, которая ложится на управляющую ЭВМ) подсистемы регистрации и хранения информации. Программы организованы в виде интерпретаторов макрокоманд, они обеспечивают управление модулями КАМАК, первичную арифметическую и логическую обработку данных, преобразование во внешнее представление канала передачи данных, диалог с экспериментатором. Язык программирования - Ассемблер РДР-8.

Библ.5, табл.6.

УДК 007.5 : 681.3.045

Помехоустойчивое кодирование данных на перфолене. Золотенков В.В., Матушевский В.В. Сб.: Автоматизация эксперимента и машинная графика. Томск, Изд-во ТГУ, 1977, 43-50.

Для повышения надежности хранения информации на перфолене предлагается использовать коды, исправляющие ошибки. Описывается реализация системы, обеспечивающей защиту как от одичных ошибок перфорации, так и от ошибок синхронизации. Применяется код Хемминга (7,4) и синхронизирующие последовательности Баркера. Изыточность около 100%.

Библ.3, ил.1.

УДК 007.5 : 681.3.069

Программы обслуживания архивов в системе САФРА. Буллер И.Я. Сб.: Автоматизация эксперимента и машинная графика. Томск, Изд-во ТГУ, 1977, 51-58.

Описывается реализация на ЕС ЭВМ подоистемы регистрации и хранения информации в системе САФРА, осуществляющей преобразование данных из физического представления на носителе во

Система графического вывода СМОГ и её реализация на ЕС ЭВМ. Г л а д к и х Б.А., К о о т ю к Ю.Л. Сб.: Автоматизация эксперимента и машинная графика. Томск, Изд-во ТГУ, 1977, 103-115.

Система математического обеспечения графопостроителей СМОГ ориентирована на научные и инженерные расчеты. Она представляет собой иерархическую структуру операторов (подпрограмм), содержащую универсальный базис и проблемно-ориентированные надстройки. В статье кратко описываются графические возможности базиса и особенности его реализации. Приведены укрупненные алгоритмы программ базиса на языке Алгол-68.

Библи.6, ил.1.

Применение сплайнов для изображения линий в машинной графике. К о о т ю к Ю.Л. Сб.: Автоматизация эксперимента и машинная графика. Томск, Изд-во ТГУ, 1977, 116-130.

Для интерполяции линий в графических системах весьма перспективно использование сплайнов. В статье рассматриваются простые 3-точечные локальные сплайны в параметрической форме. Для них устанавливается ряд полезных свойств, на основании которых можно считать, что такие сплайны наиболее удобны для изображения кривых. Приводится алгоритм вычисления сплайнов на ЭВМ.

Библи.6, ил.4.

Язык представления рисунков в системе СМОГ. К о о т ю к Ю.Л., Ф у к с А.Л. Сб.: Автоматизация эксперимента и машинная графика. Томск, Изд-во ТГУ, 1977, 131-139.

Система математического обеспечения графопостроителей СМОГ имеет продуманную организацию, в то же время в ней имеются средства описания фрагментов рисунков на специальном графическом языке. В нем используются операторы координатных преобразований, операторы цикла, линии и имена других рисунков, а также тексты. В статье описываются структура языка и особенности его реализации на ЭВМ.

Библи.1, ил.3.

внутреннее представление обрабатывающей ЭВМ, а также создание архивных наборов данных, доступных подсистеме обработки.

Библ.4, пл.2.

УДК 007.5 : 681.3.068

Применение системы статистического анализа САС для обработки экспериментальных данных в системе САФРА. М а т у ш е в с к и й В.В. Сб.: Автоматизация эксперимента и машинная графика. Томск, Изд-во ТГУ, 1977, 59-74.

За основу программного обеспечения подсистемы обработки данных принята система статистического анализа САС, разработанная в университете штата Северная Каролина (США) под руководством А.Барра и Дж.Гуднайт. В статье кратко описывается входной язык САС и особенности её использования в системе САФРА.

Библ.8.

УДК 007.5 : 681.3.069

Система вывода сложных изображений на АЦПУ ЕС ЭВМ (на базе ПЛ/І ОС). П о т а п о в Д.В. Сб.: Автоматизация эксперимента и машинная графика. Томск, Изд-во ТГУ, 1977, 75-89.

Описывается система программы, реализующая двухкоординатный принцип построения изображений на АЦПУ. Такой способ значительно удобнее построения при печати сложных таблиц, графиков, рисунков. Приведена распечатка текстов программ.

Библ.1, ил.3.

УДК 007.5 : 681.3.068

Обзор систем графического вывода. К о с т я к Ю.Д. Сб.: Автоматизация эксперимента и машинная графика. Томск, Изд-во ТГУ, 1977, 90-102.

Аналитический обзор 13 систем (отечественных и зарубежных), ориентированных на вывод результатов инженерных и научных расчетов на графоэкранные устройства. Обзор составлен по "двухкоординатному" принципу: сначала системы описываются в целом, затем рассматриваются отдельные графические функции и сравниваются их реализация в различных системах.

Библ.31, табл.1.

Универсальная процедура рисования графиков.

К о о т в к Ю.Л. Сб.: Автоматизация эксперимента и машинная графика. Томск, Изд-во ТГУ, 1977, 140-147.

Наиболее сложной процедурой при автоматическом построении графиков является разметка координатной сетки. Даже в случае равномерного или логарифмического масштаба алгоритм автоматического разбиения нетривиален. Применение же произвольного функционального масштаба (особенно удобного, например, при проверке совпадения экспериментальной зависимости с теоретической) делает эту задачу весьма непростой. В статье описывается алгоритм разбиения при любых непрерывных масштабных функциях, основанный на использовании предложенных автором "красивых" последовательностях чисел. Приводятся алгоритмы разбиения шкал и построения графиков, реализованные в системе СМОГ (на языке Алгол-68).

Библ.3.

С о д е р ж а н и е

Г л а д к и х Б.А. Автоматизированная система регистрации, хранения и обработки экспериментальных данных "СМФРА" - общее описание	3
Г л а д к и х Б.А., М а т у ш е в с к и й В.В. Представление данных в системе СМФРА	22
З о л о т е н к о в В.В., О с и п о в а Л.Б. Математическое обеспечение для управления экспериментом в оистеме СМФРА	31
З о л о т е н к о в В.В., М а т у ш е в с к и й В.В. Помехоустойчивое кодирование данных на перфоленте	43
Б у л л е р И.Я. Программы обслуживания архивов в системе СМФРА	51
М а т у ш е в с к и й В.В. Применение системы статистического анализа САС для обработки экспериментальных данных в системе СМФРА	59
П о т а п о в Ю.В. Система вывода сложных изображений на АШУ ЕС ЭЕМ (на базе III/I ОС)	75
К о с т ю к Ю.Д. Обзор систем графического вывода	90
Г л а д к и х Б.А., К о с т ю к Ю.Д. Система графического вывода СМОГ и ее реализация на ЕС ЭЕМ	103
К о с т ю к Ю.Д. Применение сплайнов для изображения линий в машинной графике	116
К о с т ю к Ю.Д., Ф у к с А.Д. Язык представления рисунков в системе СМОГ	131
К о с т ю к Ю.Д. Универсальная процедура рисования графиков	140
РЕЗЬЕРАТЫ НА ОПУБЛИКОВАННЫЕ СТАТЬИ	148

Автоматизация эксперимента и машинная графика

Редактор Л.И.Диканова

КЗ 03178

Подписано к печати 1/xII - 77

**Формат 60 x 84 1/16, бумага типографская № 2. Печ.л. 9,5 ;
уч.-изд.л.8 ; уол.печ.л.8,8. Заказ 107**

Тираж 400. Цена I руб.23 коп.

Издательство ТГУ, Томск-29, ул.Никитина, 17

Ротапринт ТГУ, Томск-29, ул.Никитина, 17