УДК 681.3.06

А. Н. ГИНЗБУРГ, Ю. И. РОДИОНОВ
(Новосибирск)

СТРУКТУРА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ СИСТЕМЫ «ЭКРАН»

В предлагаемой работе делается попытка дать систематическое изложение одного из подходов к решению задачи создания программного обеспечения системы оперативного графического взаимодействия «человек — машина» [1, 2], которое было бы в достаточной степени независимым от конкретной аппаратуры, во-первых, и могло бы быть использовано для решения широкого круга задач, во-вторых.

Программное обеспечение можно разделить на следующие части 1) программы привязки дисплея к оперативной системе ЭВМ (нулевой уровень); 2) программы перевода с языка команд дисплея на язык ЭВМ и обратно (первый уровень); 3) программы представления графических данных (второй уровень). Программы второго уровня должны давать пользователю возможность не только описать графическое изображение произвольной структуры, но и производить манипуляции с изображениями и их частями. Это следующие операции: выделить фрагмент изображения, сместить его, стереть или повторить его в любой точке экрана. Сюда также относятся поворот фрагмента, масштабирование и т. п.

Рассмотрению программ нулевого уровня посвящены, например, [3, 4]. Процедуры второго уровня весьма сжато описываются в [5]. Здесь рассмотрены программы, относящиеся к первому и второму уровням. Обсуждаемое ниже программное обеспечение выполнено на алгоритмическом языке Фортран.

Программы адаптации команд дисплея к ЭВМ и обратно. Цикл процедур, осуществляющих перевод с Фортрана на язык процессора дисплея и обратно, необходим не только в связи с использованием алгоритмического языка достаточно высокого уровня, но и по той причине, что системы команд ЭВМ и процессора дисплея, как правило, имеют мало общего между собой и, кроме того, запоминающее устройство (ЗУ) ЭВМ и буферное ЗУ (БЗУ) дисплея почти всегда отличаются разрядностью слов.

Так, в ЭВМ БЭСМ-6 слово имеет 48 разрядов, в то время как слово БЗУ дисплея «Экран» — только 42 разряда. Рассмотрим в качестве примера команду «Высветить точку или вектор» (см. рис. 1). Здесь код операции (КОП) равен 123_8 и занимает разряды 42—36;

$NTYPE =$ {0 для изображения точки; 1 для изображения вектора;

$LUMIN =$ 0, 1, 2 или 3 — яркость;

$MIG =$ 0 или 1 — признак мерцания;

IX, IY — целочисленные координаты; $0 \leqslant IX$, $IY \leqslant 1023_{10}$. Начало координат находится в левом нижнем углу экрана. Процедура *PØINT (NTYPE, LUMIN, IX, IY, MIG)* осуществляет формирование команды 123 и занесение ее в очередную строку дисплейного файла.

Остальные процедуры, соответствующие командам дисплея, аналогичны процедуре *PØINT*. Структура собственно дисплейного файла предельно проста: это массив из 1025 слов, в котором файлом служат первые 1024 слова (массив максимальной длины, который может быть передан в дисплей и обратно); последнее слово является счетчиком, ука-

зывающим на первую свободную строку в дисплейном файле. После заполнения очередной строки счетчик увеличивается и т. д.

Перечислим необходимый минимум процедур, не останавливаясь на обсуждении их параметров. Кроме *PØINT* сюда относятся *ØPERAT*, формирующая команду «Управление», *CIRCLE* — для кодирования команды «Окружность», две процедуры для работы с пространственными векторами *SPACE*1 и *SPACE*2. Программы, необходимые для упаковки цифровых кодов символов, по-видимому, будут использоваться достаточно редко, так как авторами реализована процедура

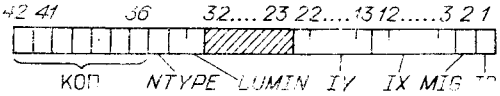

Рис. 1.

TEXT (*N*, *NF*, *IX*, *IY*, *MT*, *MIG*, *MØDE*).

Эта процедура перерабатывает алфавитно-цифровую информацию, находящуюся в ЭВМ в коде *ISØ* (ее можно заносить в память по формату *A* фортрановским оператором *READ*), благодаря чему отпадает необходимость в цифровой кодировке в код пишущей машинки «Консул» каждого символа (код, который используется в дисплее) и заносит ее в дисплейный файл. Здесь *N* — число символов в тексте; *NF* — формат текста при выводе на экран (при *NF*=0 текст размещается на экране на поле из 32 строк по 64 символа в строке; при *NF*=1 текст размещается в 32 строках по 32 символа в строке); *MT* — величина символа; *MØDE* — род работы. При *MØDE*=1 все символы имеют одинаковую величину и постоянное значение признака «Мерцание». При *MØDE*=0 размер каждого символа определяется пользователем (через вторую половину массива входных данных). Кроме того, имеются процедуры *PENCIL* (*NADR*, *IX*, *IY*) для дешифровки сообщения от светового карандаша (СК) и по одной процедуре для каждой команды дисплея для дешифровки дисплейного файла, переданного из дисплея в ЭВМ. В процедуры дешифровки введен дополнительный параметр *IP* (высвечивание), определяющий, подлежит ли текущая дисплейная команда высвечиванию. Таким образом, например, процедура, обратная процедуре *PØINT*, имеет вид

*DECØD*3 (*NTYPE*, *LUMIN*, *IX*, *IY*, *MIG*, *IP*).

Признак высвечивания заносится аппаратно в режиме «Автоном» и позволяет программе разобраться, какие линии на экране оператор хочет убрать или ввести заново.

Перечисленные программы совместно с программой привязки позволяют в принципе реализовать все возможности системы «Экран», заложенные в нее разработчиками. Однако труд по написанию и отладке программ без ввода дополнительных процедур более высокого уровня оставался бы чрезвычайно утомительным и малоэффективным даже с учетом того, что программист работал бы на Фортране, а не на машинном языке. В связи с этим обстоятельством предлагается система процедур для описания графических данных и манипуляций с фрагментами изображений.

Программы для организации списков структур. Решение построить программное обеспечение для системы «Экран» на основе использования списковых структур [6] вызвано тем обстоятельством, что графические данные могут быть чрезвычайно разнообразными, и описать их с помощью, например, матричных методов возможно только в виде исключения. Поскольку язык Фортран не имеет операторов для работы со списками, был разработан ряд процедур, написанных на Фортране и частично на автокоде, позволяющих создавать списковые структуры на выделенном массиве.

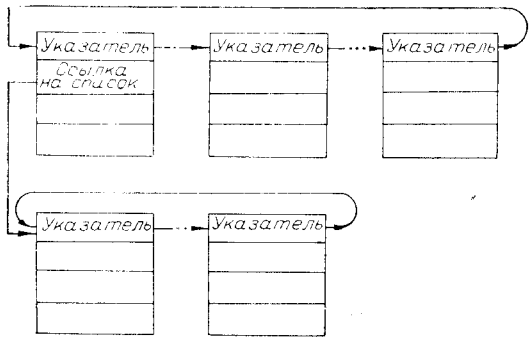

Рис. 2.

Списки организуются способом, принятым, например, в языке Симула [7], т. е. создаются списки, состоящие из «головного элемента» и обычных элементов. Каждый из элементов списка содержит ссылку на последующий. Последний элемент ссылается на головной. Таким образом, мы используем так называемые кольцевые списки (рис. 2).

Удаление элементов списка осуществляется изменением указателей. Освободившиеся элементы включаются в список пустых элементов, откуда эти слова выбираются по мере надобности. Таким путем осуществляется динамическое распределение памяти. Каждый элемент списка состоит из четырех слогов и обычно занимает одно машинное слово. Первый слог, как видно из рис. 2, всегда используется для сцепления элементов в кольцо. Второй слог при необходимости служит для ссылки на другие списки и позволяет организовать иерархические структуры. Третий слог содержит указатель, определяющий тип текущего элемента или фрагмента, к которому относится элемент, и может использоваться для модификации способа обработки структуры. Четвертый несет собственно информационную нагрузку (например, может содержать имя элемента). Назначение второго — четвертого слогов меняется и зависит от структуры, в которую включается элемент.

Минимальный набор программ, необходимых для работы со списками, следующий: *ELREAD* — чтение слога из элемента; *ELWRIT* — запись слога в элемент. Эти две программы написаны на автокоде Мадлен, так как в них производятся манипуляции с частями машинного слова. Остальные программы написаны на Фортране:

EMPTY — создать пустой список; *FØRM2* — сформировать элементы с заданными слогами;

FIND5 — найти элемент в списке по номеру элемента;

REMØV — удалить элемент из списка;

INCL2 — включить элемент в список;

LAST — найти последний элемент в списке.

Программа *LAST* необходима, так как в каждом элементе существует только указатель на следующий элемент и отсутствует указатель на предыдущий (в отличие от списков в языке Симула). Это вызвано желанием сэкономить память, так как при 48-разрядном машинном слове пятый слог в элемент добавить уже некуда (либо надо на элемент расходовать не одно, а два машинных слова).

Представление графических данных в виде графов. Описание графических данных основано на использовании списковых процедур, описанных в предыдущем параграфе. Очевидно, что в большинстве графических структур имеются повторяющиеся фрагменты. Это обстоятельство очень существенно. Действительно, описывая повторяющийся фрагмент один раз, мы экономим труд, память машины и уменьшаем вероятность ошибок. Учитывая это, мы можем любое изображение рассматривать как граф, в котором узлами являются повторяющиеся фрагменты, а ветви несут информацию о месте расположения фрагмента. Пример изображения приведен на рис. 3, а соответствующий ему граф на рис. 4.

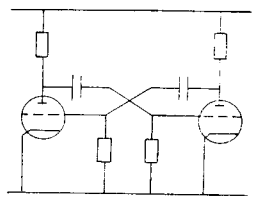

Рис. 3.

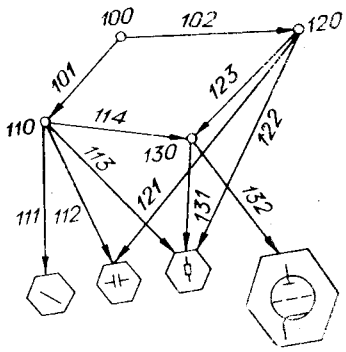

Рис. 4.

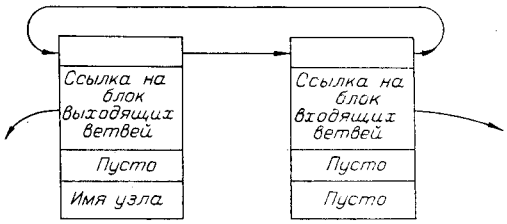

Рис. 5.

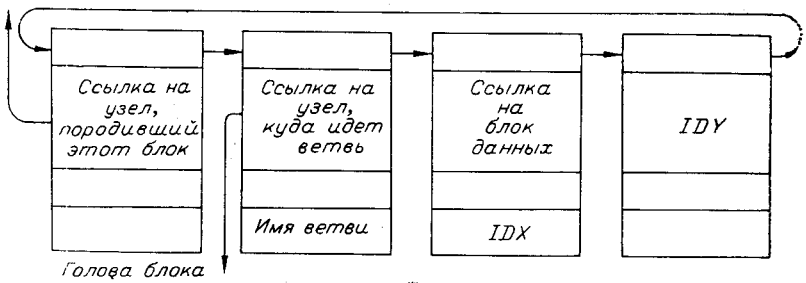

Рис. 6.

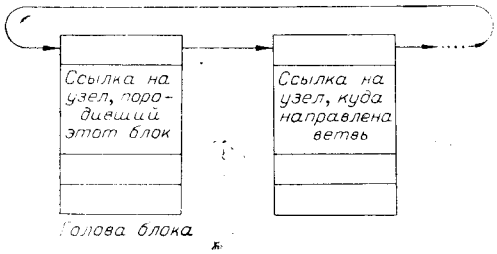

Рис. 7.

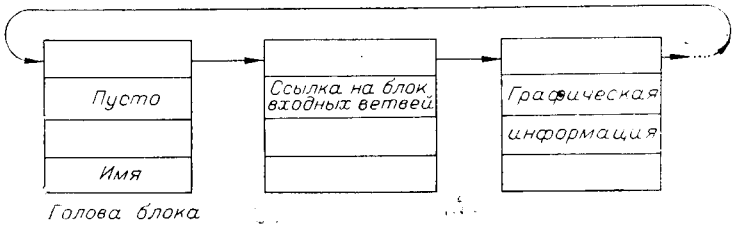

Рис. 8.

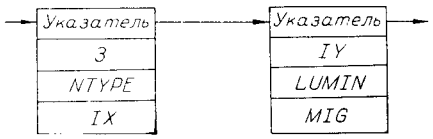

Рис. 9.

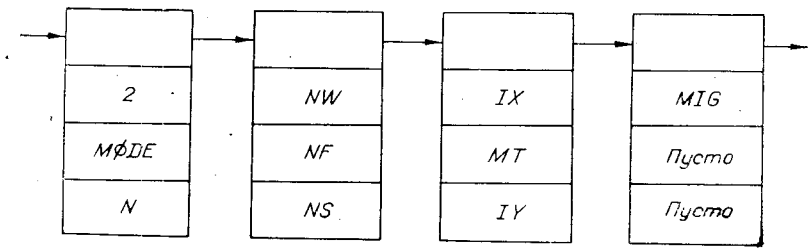

Рис. 10.

Из рис. 3 и 4 ясно, что сопротивление, например, достаточно описать лишь один раз. Описание собственно графических данных сосредоточено в оконечных узлах графа, не имеющих, естественно, выходящих ветвей. Эти узлы будем называть далее страницами (на рис. 4 обозначены шестигранниками). Описание графа осуществляется через описание блоков узлов, блоков ветвей, блоков страниц с графической информацией и блоков данных с числовыми величинами (например, о величине емкостей, сопротивлений и т. п.).

Рассмотрим эти компоненты подробнее.

Блок узла. Как уже говорилось, блок узла фиксирует некоторый объект в кадре. На него возможны повторные ссылки. В блок узла входят: а) указатель на список ветвей, выходящих из узла; б) указатель на список входящих ветвей; в) имя узла. Организация списка для блока узла приведена на рис. 5.

Блок ветвей. Структура блока выходящих ветвей ясна из рис. 6. Здесь *IDX*, *IDY* — величины приращений по осям координат для данной ветви. Незаполненные слоги элементов предназначены для пока незадействованного механизма масштабирования и поворота. Структура блока входных ветвей приведена на рис. 7.

Блок страниц. Блок страниц в отличие от блока узла не имеет ссылки на блок выходящих ветвей, во-первых, и, во-вторых, состоит не из двух элементов, а из большего числа, зависящего от объема графических данных. Структура блока страниц показана на рис. 8. В качестве примера приведем организацию записи в страницу точки — линии (рис. 9) и текста (рис. 10). Как видно из рис. 9, точка занимает два элемента; в первом слове значение второго слога равно трем, что отличает эту дисплейную команду от других. Остальные обозначения те же, что и выше.

Программы для работы с графами изображений. Блок узла создается программой *CREAK*. Эта же программа используется и для создания блока страниц (пустого). Для соединения двух узлов или узла и страницы используется программа *INCLUD*, которая создает при необходимости блоки входящих и выходящих ветвей и заполняет их в соответствии с рис. 6 и 7. Для включения в блок страницы графических данных служат программы: *CODTXT* для команды «Текст» (в соответствии с рис. 10); *CODPNT* для команды «Точка — линия» и т. д. После того как граф сформирован, с помощью программы *INSPEC* осуществляется просмотр графа; при этом, начиная с вершины графа, проводятся все возможные траектории к страницам, одновременно вычисляются координаты очередной страницы. При выходе на страницу программа *INSPEC* вызывает подпрограмму *DECODP*, которая производит дешифровку страницы и заполнение дисплейного файла.

Для примера просмотр графа, изображенного на рис. 4, будет осуществляться по следующим траекториям:

100 — 101 — 110 — 111 — 200;
100 — 101 — 110 — 112 — 201;
100 — 101 — 110 — 113 — 202;
100 — 101 — 110 — 114 — 130 — 131 — 202;
100 — 101 — 110 — 114 — 130 — 132 — 203;
100 — 102 — 120 — 121 — 201;
100 — 102 — 120 — 123 — 130 — 131 — 202;
100 — 102 — 120 — 122 — 202.

Описанных процедур достаточно для пассивного вывода на экран графической информации любого вида.

Программы для манипуляций с фрагментами изображений в режиме диалога. Для работы в режиме диалога помимо упомянутых выше процедур дешифровки дисплейного файла нужны процедуры более высокого уровня. К ним относятся прежде всего процедуры поиска объекта в структуре. Таких процедур две: *INSP*1 и *INSP*2.

Подпрограмма *INSP*1 осуществляет поиск страницы (и траектории, ведущей к ней), где находится, например, отрезок (или символ), на который указано световым карандашом. Эта программа необходима для диалога простейшего вида, в котором используется указание на объект. Более сложной является ситуация, когда нам необходимо ввести, например, отрезок световым карандашом или заменить ранее имевшийся отрезок вновь введенным, т. е. возникает задача определить, к какому из множества графических элементов, находящихся на экране, ближе всего вновь введенная точка или нарисованный отрезок. Эта проблема возникает при ручной правке эскизов, схем и т. п. Таким образом, требуется провести поиск объекта уже не по заданному адресу, а по переданным от СК координатам и некоторому критерию (например, минимум расстояний, на которые могут отстоять старая и новая точки). Поиск осуществляется программой *INSP*2.

Кроме этих программ, для работы в режиме манипулирования фрагментами изображения нужны специальные программы:

COPY — снять копию объекта, уже имеющегося в структуре, и поместить его в некоторую точку экрана;

EXCLUD — исключить объект из структуры;

SWITCH — переключить, изменить положение объекта в структуре;

DPLACE — сместить объект на экране, не меняя его положения в структуре.

Очевидно, что, имея описание изображения в виде сложной иерархической структуры, целесообразно названные процедуры использовать, главным образом, для манипуляций над сложными фрагментами изображения, а не только над его элементарными «кирпичиками» (напомним, что в роли последних здесь выступают страницы данных). Это легко реализуется, так как программы *INSP*1 и *INSP*2 находят траекторию, ведущую от головы графа к нужной странице. Двигаясь от страницы обратно (к голове графа) по траектории, можно активировать фрагменты все более высокого уровня. Индицировать текущий фрагмент можно, например, мерцанием на экране. Как только определен нужный уровень фрагмента, прекращается движение по траектории и выполняется манипуляция (снять копию, сместить и т. п.).

Заключение. Предлагаемое программное обеспечение, за исключением программ первого уровня, совершенно не зависит ни от конструкции дисплея, ни от класса ЭВМ. Поскольку большая часть программ второго уровня написана на языке Фортран, переход от одной машины к другой не требует дополнительных усилий.

Структура программного обеспечения не ориентирована на какой-либо узкий класс задач. Особенно эффективным оно окажется при решении задач, связанных с анализом различного рода сетей, схем и т. п.

При решении конкретных задач не требуется слишком высокой квалификации для разработки пакетов программ, в основу которых положены описанные базовые процедуры.

Дальнейшие разработки, по нашему мнению, должны дать пользователю следующие дополнительные возможности: ввод и вывод числовой информации; присвоение имен фрагментам изображений; манипуляции над фрагментами изображений через посредство имен (что особенно удобно при работе с движущимися объектами); выделение на экране части очень сложной структуры («вырезание окна ножницами»); вывод на экран фрагмента в увеличенном виде и т. д.

ЛИТЕРАТУРА

1. Б. С. Долговесов и др. Система «Экран» для графического взаимодействия человека с ЭВМ.— Автометрия, 1971, № 4.
2. Б. С. Долговесов и др. Отображение графической и буквенно-цифровой информации в системах графического взаимодействия человека с ЭВМ.
3. Е. Г. Бабат и др. Адаптация системы «Экран» в ЭВМ БЭСМ-6.— Автометрия, 1971, № 4.
4. Е. Г. Бабат и др. Система «Экран» для графического взаимодействия с ЭВМ БЭСМ-7. Препринт ИАЭ СО АН СССР. Новосибирск, 1972.
5. Ю. И. Родионов. О математическом обеспечении системы «Экран». Представление графических данных в ЭВМ.— Тезисы конференции «Автоматизация научных исследований на основе применения ЭВМ». Новосибирск, 1972.
6. B. Zacharov. Computer Graphics.— Proc. of the 1970 CERN Computing and Data Processing Schol, CERN 71-6, 1971, 223—291.
7. О. И. Дал, К. Нигард. Симула — язык для программирования систем с дискретными событиями.— В сб. «Алгоритмы и алгоритмические языки», вып. 2. М., 1967.

Поступила в редакцию 25 октября 1972 г.

УДК 681.3.06

А. Н. ГИНЗБУРГ, А. В. ЛОГИНОВ, В. М. ПЛЯСОВ

(Новосибирск)

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ В СИСТЕМЕ ГРАФИЧЕСКОГО ВЫВОДА

В связи с постоянным развитием вычислительной техники в области автоматизации научных исследований и машинного проектирования возникла настоятельная потребность в выводе информации из ЭВМ в виде графиков, чертежей и рисунков со всевозможными подписями и пометками. В данной работе описан набор программ графического вывода в системе «ЭВМ» класса "Минск" (в виду совместности моделей "Минск-22", "Минск-22М", "Минск-32") — графопостроитель "Вектор-1301"». В частности, комплекс программ написан для ЭВМ «Минск-22» на языках Фортран ИФВЭ-67 и Полукод ИФВЭ-67.

Планшетный графопостроитель (ГП) «Вектор-1301» представляет собой самописец, выполненный в виде стола, параллельно граням которого перемещается пишущий узел с породержателями. В настоящее время работает система «Графопостроитель "Вектор-1301"—"Минск-22"», которая эксплуатируется в Институте автоматики и электрометрии СО АН СССР. Связь ГП «Вектор-1301» с ЭВМ «Минск-22» осуществляется с помощью специальной команды «—65». В младших восьми разрядах слова, определяемого вторым адресом команды «—65», содержится команда управления графопостроителем. Для работы с прерыванием отведена 25-я ячейка.

Состав и структура комплекса. На рисунке приведена архитектура комплекса программ (язык Фортран), дающего пользователю следующие возможности: а) устанавливать формат поля рисунка и режимы работы (отладка, защита рисунков и т. д.); б) неоднократно изменять систему координат; в) перемещаться в любую точку поля рисунка как с опущенным, так и с поднятым пером; выводить г) единичный символ в любом формате; д) числа и сегменты текстов в любом формате; е) оси