**Графическое программирование на Ms Fortran**

Курс лекций по дисциплине по специальности «ПГС» после техникума

Факультет – строительный; кафедры – «Металлические и деревянные конструкции» и «Железобетонные и каменные конструкции»

**Задачи курса**

Усвоение основных принципов разработки инженерных алгоритмов и графических программ, реализуемых в дальнейшем на различных языках программирования. Язык Fortran рассматривается не как самоцель, а как инструмент обучения программированию.

Литература

1) Уорд Т., Бромхед Э. Фортран и искусство программирования персональных ЭВМ. - М.: Радио и связь, 1993.-352с.

2) Программирование на Фортране 77/ Дж. Ашкрофт, Р. Элдридж и др. - М.: Радио и связь, 1990.-272с.

3) Бартеньев О.В. Фортран для студентов. - М.: "Диалог-МИФИ", 1999.-400с.

4) Рыжиков Ю.И. Программирование на Фортране Pоwer Statiоn для инженеров. Практич. рук-во. - СПб.: КОРОНАпринт, 1999.-160с.

5) Н.А Рак, В.И Смех, С.Б. Щербак Программирование инженерных задач на FORTRANе: Уч. методич. пособие по дисц. "Выч. техн. и программир." - Мн.: БГПА, 1998.-45с.

6) Соловьев П.В. FORTRAN для персонального компьютера: Справ. пособие. - М.: Арист, 1991.-224с.

7) Н.А Рак, В.И Смех, С.Б. Щербак Графическое программирование: Уч. методич. пособие по дисц. "Выч. техн. и программир." - Мн.: БГПА, 1997.-40с.

8) Бартеньев О.В. Графика OрenGL: программирование на Фортране. М.: ДИАЛОГ-МИФИ, 2000.-368с.

**Этапы разработки графических программ**

Для разработки графических программ на Fortran необходимо иметь в наличии библиотеки графических процедур, содержащиеся в файле Graphics.lib.

Разработка графических программ состоит из следующих этапов:

Подключение графической библиотеки

Установка видео режима

Установка среды изображения

Создание и управление графическими объектами

Восстановление первоначальной конфигурации перед выходом из программы

Графические процедуры представляют собой функции и подпрограммы. Каждая программа, использующая графическую библиотеку должна ЯВНО объявить процедуры. Отдельную используемую графическую процедуру можно объявить, написав ее интерфейс (смотри справку NORTON GUIDE). Сразу все графические процедуры можно описать, включив в программу файлы FGRAPH.FI и FGRAPH.FD. FGRAPH.FI – содержит объявление процедур в операторах INTERFACE (см. далее).

Оператор INTERFACE объявляет имя процедуры (функции), ее атрибуты и типы формальных параметров. Оператор используется для обеспечения корректности вызовов п/п. Компилятор выполняет проверку числа и типов параметров в вызове п/п с параметрами, указанными в интерфейсе.

Пример:

INTERFACE TO INTEGER FUNCTION Func (p, d, q)

Integer\*2 p

Real d типы параметров

Real\*8q

END

где: INTEGER – тип функции; FUNCTION – функция; Func – имя функции; p, d, q – формальные параметры.

Здесь объявлена целая функция (возвращающая целое значение) Func с тремя входными параметрами p, d, q. Вызов данной функции обычен:

K = Func (p,d,q)

INTERFACE TO \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ SUBROUTINE Mix (a,b,c) – объявляет п/п (находится в библиотечном файле с расширением .LIB), находящейся в другом файле.

Вызов п/п обычен CALL MIX (a,b,c)

Файл FGRAPH.FD содержит объявления структурных и символических констант и объявлений операторов EXTERNAL для графических процедур.

Пример, кода файл FGRAPH.FD:

имя структуры

STRUCTURE /videoconfig/

Integer\*2 numxpixels элементы структуры

а) Integer\*2 numypixels

………………………. операторы объявления типов и/или RECORD.

END STRUCTURE

б) INTEGER\*2 setvideomode [extern] PARAMETER ($YELLOW = #153f3f)

в) EXTERNAL getvideoconfig PARAMETER ($BRIGHTWHITE = #3f3f3f)

г) INTEGER\*4$YELLOW,$BRIGHTWHITE

а) Определение составного типа переменной (структуры).

Обратиться к элементу структуры (для присвоения или получения значения) можно следующим способом:

X = videoconfig.numxpixels

имя структуры элемент структуры

Структуры могут вкладываться друг в друга с использованием оператора RECORD – объявляет структуру для имен, определяемых пользователем.

Пример:

STRUCTURE /full\_adress/ !объявление структуры

RECORD /full\_name/ personsname !вложение структуры full\_name под именем personsname.

INTEGER\*1 age

INTEGER\*4 phone

END STRUCTURE

Структура full\_name может иметь следующий вид:

STRUCTURE /full\_name/

Character\*15 first\_name

Character\*20 last\_name

END STRUCTURE

Обращение к элементам структуры может быть осуществлено следующим образом:

WRITE (5,\*) ‘имя:’, full\_adress.personsname.first\_name

WRITE (5,\*) ‘возраст:’ full\_adress.age

б) Объявление целой функции setvideomode.

Атрибут EXTERN указывает, что переменная (в данном случае являющаяся именем функции) размещается в другом исходном файле. Вообще атрибут позволяет определить для переменной, типа переменной, п/п или формальных параметров п/п некоторую дополнительную информацию. Атрибут заключается в квадратные скобки [ ].

Ранее рассматривался пример с атрибутом ALLOCATABLE, указывающим, что массив является динамическим (“размещаемым”).

в) Объявление пользовательского имени как внешней п/п getvideoconfig.

г) Объявление символических констант. В данном случае, задаются значения цветов (в шестнадцатиричном виде) двум целым константам.

Указанные файлы содержат объявления, необходимые для использования графических процедур.

Файл FGRAPH.FI включается в начало каждого исходного файла программы, а FGRAPH.FD в каждую п/п, содержащую графическую процедуру. Включение производится оператором include – вставка содержимого указанного файла в позиции оператора include.

Пример:

include ‘fgraph.fi’ ! вставка текста файла fgraph.fi

………………….

program LINE

………………….

include ‘fgraph.fd’

…………………. символическая константа

I2 = setvideomode ($MAXRESMODE) ! графическая функция установки видеорежима

………………….

end

2. Установка видеорежима

Установка видеорежима необходима для выполнения графических процедур. Под видеорежимом подразумевается:

тип используемого монитора и видеокарты;

разрешение экрана;

использование цветного или монохромного изображения;

количество отображаемых цветов, определяемое количеством памяти выделяемой под цвет. Так “четырех битный цвет” соответствует шестнадцати цветам, восьми битный – 256 (т.е. два в восьмой степени) и т.д.

В настоящее время обычно используется 16 и 24 битные цвета.

Иногда до установки видеорежима для предотвращения возможных проблем несовместимости необходимо получить информацию о текущей видеосистеме.

Получение такой информации:

Record /videoconfig/ Myvideoconf ! включение структуры videoconfig под именем Myvideoconf

……………………..

call getvideoconfig (Myvideoconf) ! вызов п/п определения видеоконфигурации

Элементы структуры videoconfig описаны в файле fgraph.fd. Данной п/п можно узнать разрешение экрана, возможное количество цветов и т.д. Значения возвращаются через элементы структуры.

Установить графический видеорежим можно функцией setvideomode.

Пример:

dum = setvideomode ($MAXRESMODE)

В функцию передается идентификатор (номер) видеорежима. В данном случае передается символическая константа целого типа $MAXRESMODE (= - 3), определенная в файле FGRAPH.FD и определяющая, что выбирается и устанавливается видеорежим с максимальным разрешением.

В случае возникновения ошибки функция возвращает значение 0.

3. Установка среды изображения.

Перед непосредственным выводом изображения необходимо произвести некоторые предварительные настройки.

Установка курсора.

Функция displaycursor позволяет включить или выключить изображение курсора.

Пример:

I2 = displaycursor (1) ! включение курсора, 0 – выключение. Можно использовать символические константы $GCURSOROFF и $GCURSORON.

3.2 Установка палитры.

Каждому цвету на экране ставится в соответствие индекс цвета, по которому он вызывается. Обычно имеется 16 индексов, которым соответствуют 16 стандартных цветов, хотя самих цветов может быть больше.

Для ассоциирования цвета конкретному индексу используются процедуры: remappallpalette, remappalette и selctpalette.

Значение цвета определяется целым числом, состоящим из четырех байт (32 битов). Цвет задается указанием уровня интенсивности для каждого значения красного, зеленого и голубого цветов (RGB модель).

Значение цвета в двоичном виде:

zzzzzzzz zzBBBBBB zzGGGGGG zzRRRRRR, где z это 0, а B, G и R – значения битов для соответствующих цветов различной интенсивности. 1 (двоичные 00000001) соответствует минимальной интенсивности, а 63 (двоичное 00111111) максимальной. Чисто желтому цвету (100% зеленого и 100% красного) соответствуют значение:

(00000000 00000000 00111111 00111111)2 = (1392447)10 = (153f3f)16

белому (всего по 100%): (4144959)10 = (3f3f3f)16

Пример:

i2 = remapalette (1#153f3f) !присвоение индексу 1 желтого цвета

аналогично

i2 = remapalette (1,1392447)

аналогично

i2 = remapalette (1,$YELOW) !где $YELOW – константа, соответствующая желтому цвету (определена в файле Fgraph.fd)

Функция remappallpalette выполняет одновременное переназначение всех индексов цвета. В функцию передается массив значений цветов.

Установка координатных систем

Текстовые координаты.

Текстовые координаты служат для отображения текста шрифтом фиксированного размера. Координаты текста задаются в строках и столбцах. В обычном текстовом режиме: 25 строк и 80 столбцов. Установка текстовых координат осуществляется графической п/п settextposition.

Пример:

call settextposition (10, 50, rc)

10, 50–координаты начала текста; rc – локальное имя структуры rccoord (см. fgraph.fd) с помощью которой возвращается и передается в п/п предыдущая текстовая координата для ее переразмещения в новую. Координата (1,1) находится в левом верхнем углу.

Физические координаты.

Физические координаты являются фиксированными и определяются техническими средствами (видеокартой и монитором).Начало координат находится в левом верхнем углу, если не используется п/п setvieworg. Разрешения 640\*480; 800\*600. Физические координаты являются целочисленными.

Пример: перенос начала координат в точку (50, 100)

CALL setvieworg (50, 100, xy)

Теперь координата x изменяется от 50, до .numxpixels-50, а y, от 100 до .numypixels-100, где numxpixels и numypixels – элементы структуры videoconfig, содержащие максимальное число пиксел по осям OX и OY. Переустановка начала координат повлияет на все графические процедуры.

Используя физические координаты можно установить невидимую координатную область ‑ “область клипа”. Рисование в этом случае производится только в этой области, попытка рисования вне области игнорируется.

Пример:

call setcliprgn (10,10,100,100) – установка квадратной “области клипа”. Процедура setcliprgn не изменяет систему координат.

3.3.3 Координаты порта просмотра

“Порт просмотра” (“viewport”) – прямоугольная область экрана в которой осуществляется рисование. “Порт просмотра” устанавливается процедурой setviewport. Действие этой процедуры аналогично процедуре setcliprgn, но с переносом начала координат в левый верхний угол области. При использовании “оконных” координат необходимо создать "порт просмотра”.

Пример:

Call SetViewPort(20,20,200,200)

Аналогично:

Call SetClipRgn(20,20,200,200)

Call SetViewOrg(20,20)

Оконные координаты.

Оконные координаты используются для масштабирования изображения. Они устанавливаются для текущего порта просмотра. Все функции работы с окнами оканчиваются символом “-w”. Значение оконных координат имеют не целочисленный тип Integer (как обычно), а вещественный тип Real\*8.

Пример:

dum = SetWindow (.TRUE.,-200.,-56,3.,500.,400.) – установка оконных координат в текущем “порту просмотра”

.TRUE. – увеличение по оси ординат снизу вверх

.FALSE. – увеличение по оси ординат сверху вниз

|  |  |
| --- | --- |
|  | На рисунке в круглых скобках ( ) показаны физические координаты, в квадратных [ ] оконные координаты. На каждый физический пиксель по ОХ приходится  оконных пикселя и по OY 1.91.  При нулевых размерах окна функция не выполняется. При успешном выполнении функция setwindow возвращает ненулевое значение. |

Пример работы с координатными системами. Приведены стандартная головная программа и п/п.

$DEBUG ! Производит дополнительные проверки при компиляции

include 'fgraph.fi' ! Включение интерфейсов графических

interface to function getch()

integer getch[C,alias:’\_getch’]

end

program graph\_examples

implicit NONE ! Запретить неявное объявление типов

integer \*2 dum ! Фиктивная переменная

integer getch[EXTERN]

include 'fgraph.fd' ! Объявление типов графических функций

dum=setvideomode($MAXRESMODE) ! Установить видеорежим с максимальным разрешением

call graph\_win()

dum=getch() ! Задержка до нажатия любой клавиши

dum=setvideomode($DEFAULTMODE) ! Установить предыдущий видеорежим

end

subroutine graph\_win()

include 'fgraph.fd'

integer\*2 dum

! DOUBLE PRECISION dg,dv,maxdim,xcen,ycen

real\*4 dg,dv,maxdim,xcen,ycen

record /xycoord/xy

record /videoconfig/vc

! Перемещение начала координат в точку (50,100)

call setvieworg(50,100,xy)

! Рисование диагональной линии

call getvideoconfig(vc) ! определение видеоконфигурации

dum=setcolor(2)

call moveto(-50,-100,xy)

dum=lineto(vc.numxpixels-50,vc.numypixels-100)

read\*

call setviewport(100,100,200,200) ! окно просмотра, координаты - физические

dum=floodfill(1,1,3) ! заполнение области (окна просмотра) текущим цветом

! при отсутствии окна просмотра заполняется весь экран

read\*

! Рисование и масштабирование эллипса в квадратном окне просмотра

call clearscreen($gclearscreen)

write(5,\*)'Введите диаметры эллипса по горизонтали и вертикали'

read(5,\*)dg,dv

call setviewport(100,100,300,300) ! окно просмотра, координаты - физические

dum=rectangle($gborder,1,1,199,199) !рамка окна

maxdim=max(dg,dv) ! Максимальный габарит рисунка

dum=setwindow(.false.,0.,0.,maxdim,maxdim)

if(dum.eq.0) write(5,\*)'Ошибка установки окна'

! Рисование, масштабирование эллипса и его центровка

xcen=0.5\*(maxdim-dg)

ycen=0.5\*(maxdim-dv)

dum=ellipse\_w($gfillinterior,xcen,ycen,xcen+dg,ycen+dv)

end



На рисунке в круглых скобках ( ) показаны физические координаты, в квадратных [ ] координаты порта просмотра, в фигурных { } оконные координаты.

3.4. Установка атрибутов фигур.

Под атрибутами рисования понимается:

установка текущего фона рисования

установка текущего цвета рисования

установка текущего способа изображения линии (типа)

установка текущей маски заполнения цветом (штриховки)

Установка цвета фона с индексом 2:

dum = setbcolor (2). По умолчанию индекс 2 соответствует зеленому цвету.

Аналогично dum = SetBkColor ($green). $green – символическая константа, указанная в файле fgraph.fd.

Toже цвета рисования.

dum = setcolor (2)

В случае неуспешного выполнения возвращается значение -1.

Установка шаблона для рисования линии

Call SetLineStyle (center), где center – имя 16-ти битной переменной. Бит 1 указывает на рисование соответствующего пиксела текущим цветом, 0 на отсутствие изменений цвета соответствующего пиксела. Шаблон повторяется по всей длине линий.

Пример задания переменной center.

$NODEBUG

integer\*2 center /2#1111111100011000/ ! штрихпунктирная линия – ‘ – ‘ – ‘ –

$DEBUG

Установка шаблона штриховки.

Call SetFillMask (shad), где shad – 64-х битный массив 8\*8 бит, где каждый бит представляет собой пиксел.

Пример задания shad:

$NODEBUG

integer\*1 shading (8)/  
> 2#00000001,  
> 2#00000010,  
> 2#00000100,  
> 2#00001000,  
> 2#00010000,  
> 2#00100000,  
> 2#01000000,  
> 2#10000000/

$DEBUG

Создание и управление графическими объектами

4.1. Рисование графических примитивов

Рисование эллиптических дуг.

|  |  |
| --- | --- |
|  | Функция аrc использует систему координат “порта просмотра”, функция arc\_w использует систему ”оконных” координат. Центр дуги есть центр ограниченного прямоугольника.  Dum = arc (x1, y1, x2, y2, x3, y3, x4, y4). |

Очистка экрана.

П/п ClearScreen очищает область, заполняя ее текущим цветом фона.

Пример:

Call ClearScreen ($GCLEARSCREEN) – очистка всего экрана (заполнение его текущим цветом фона)

При символической константе $GVIEWPORT – очистка текущей области просмотра; $GWINDOW – очистка текущего окна.

Рисование эллипса

|  |  |
| --- | --- |
|  | dum = ellipse ($GFILLINTERIOR, x1, y1, x2, y2) – рисование эллипса, закрашенного текущей маской заполнения.  При использовании символической константы $GBORDER – рисуется контур эллипса, ellipse\_w – тоже в “оконных” координатах. |

Закрашивание замкнутой области.

dum = FloodFill (x, y, bnd) – закрашивание замкнутой области с точки (х,у) (не лежащей на контуре) текущим цветом. Bnd – индекс цвета ограничивающего контура. При достижении цвета bnd закрашивание в данном направлении заканчивается. При неуспешном выполнении возвращается нулевое значение.

Рисование линии.

Функция LineTo рисует линию с текущей позиции до заданной точки (х, y) в координатах порта просмотра. После успешного выполнения (возвращено ненулевое значение) текущая позиция устанавливается в точку (х, у). dum = LineTo (x, y)

Перемещение текущей позиции графического вывода в заданную точку (х1, у1)

Calll MoveTo (x1, y1, xy) – xy – локальное имя структуры xycoord, возвращающей координаты предыдущей позиции.

Рисование прямоугольника.

dum = Rectangle ($GFILLINGTERIOR, x1, y1, x2, y2)

При использовании $GBORDER рисуется только контур прямоугольника. Возвращается 0 только при возникновении ошибки.

Рисование точки.

dum = SetPixel (x, y), в случае ошибки возвращается – 1.

Пример см. на рисунке стр. 5

4.2. Отображение текста

Шрифты делятся на:

битовые – отображаются поточечно ‑ символ состоит из набора точек. Битовые шрифты имеют фиксированный размер и не масштабируются.

векторные – символ состоит из последовательных линий и дуг, его образующих. Векторные шрифты масштабируемы.

Данная ситуация аналогична форматам изображения.

точечному (растровому), состоящему из разноцветных точек (bmp).

векторному, состоящему из набора графических примитивов (wmf, dwg). Конвертация из растрового формата в векторный более сложна, чем наоборот, но возможна (например, с помощью программы RasterVect (htttp://www.rastervect.com))

Шрифты в MS Fortran:

|  |  |  |
| --- | --- | --- |
| Тип шрифта | Отображение | Размеры |
| Courier  Helv  Tms Rmn  Modern  Roman | Битовый  Битовый  Битовый  Векторный  Векторный | 10\*8; 12\*9; 15\*12  10\*5; 12\*7; 15\*8; 18\*9; 22\*12; 28\*16  10\*5; 12\*6; 15\*8; 16\*9; 29\*12; 26\*16  масштабируемый  масштабируемый |

Для использования шрифтов необходимы следующие действия:

выбор (регистрация) шрифта

установка текущего шрифта из регистра

отображение текста текущим шрифтом

выгрузка шрифтов из памяти

4.2.1. Выбор шрифта

Процесс регистрации шрифта представляет собой загрузку в ОЗУ соответствующего .fon файла

Синтаксис: dum = RegisterFonts (filename), где: filename – имя fon файла с необязательным указанием пути доступа. Возможно использование подстановочных символов \* и ?.

Пример:

Dum = RegisterFonts (‘C:\MSLIB\FONT\\*.fon‘)

Вывести текст шрифтом может только функция outgtext. В случае ошибки возвращается следующее значения:

- 1 нет заданного файла или каталога

- 2 один или более FON файлов является двоичными

- 3 один или более FON файлов испорчены.

4.2.2. Установка шрифта

Установка шрифта производится функцией SetFont

Пример:

Dum = SetFont (“t‘Roman’h18”)

После выполнения функции setfont установленный шрифт становится текущим, т.е. он действует до тех пор, пока не будет установлен другой шрифт (либо не отменен видеорежим, либо не окончена программа). Текущим может быть только один шрифт. Параметры функции setfont задаются строкой символов, которые определяют желаемые характеристики выбранного шрифта. Порядок параметров не имеет значения.

|  |  |
| --- | --- |
| Параметр | Значение |
| t’fontname’  hy  wx  f  p  v  r  b | Имя шрифта. В апострофах указывается одно из следующих: courier, helv, tms rmn, modern, script, roman, helv Cyrillic  Высота символов пикселях (у)  Ширина символов пикселях (х)  Выбор фиксированного шрифта  Выбор пропорционального шрифта  Выбор векторного шрифта  Выбор битового шрифта  Выбор наиболее подходящего шрифта |

Функция setfont при успешном выполнении возвращает 0, иначе – 1.

Приоритет параметров при выборе шрифта

высота

начертание (f/p)

ширина

разрядка (v/r).

Отображение текста.

Для написания текста текущим шрифтом необходимо:

установить координату вывода текста п/п move to – см. выше

передать текст на экран с помощью процедуры OutGText.

Пример:

Call MoveTo (50,60,xy)

Call OutGText (‘Minsk, 2001’)

После выполнения п/п outgtext текущая графическая позиция изменяется. Отметим, что данная п/п не обеспечивает возможность форматирования текста, то есть для вывода целых или вещественных значений сначала необходимо их преобразование в строку (символьную константу или т. н. внутренний файл) с помощью оператора write.

Character \*12 val

……………………

call moveto (xdin + 5, posy, xy)

call outgtext (‘E=’)

write (val,’(f12.2\)’)E

call outgtext (val) ! вывод значения

call outgtext (' MPa')

Определение параметров установленного шрифта можно осуществить функцией GetFontInfo, которая возвращает характеристики шрифта в виде структуры fontinfo, определенной в файле fgraph.fd.

Пример:

RECORD /fontinfo/fi

………………………

dum = GetFontInfo (fi)

write (5,\*)’ Путь к файлу шрифта –‘,fi.filename

Элементы структуры fontinfo:

type – тип шрифта: 1-векторный, 0-битовый

ascent – размещение от верхней базовой линии пиксела

Integer\*2 pixwidth –ширина символов в пикселях (0-пропорциональный шрифт)

pixheight - высота символов в пикселях (0-пропорциональный шрифт)

avgvidth – средняя ширина символов в пикселях

filename – имя файла с путем доступа

Character facename – имя шрифта.

Выгрузка шрифта.

Выгрузка зарегистрированных шрифтов из памяти осуществляется п/п UnRegisterFonts ()

Пример программы деления экрана на 4 области.

! File: geomshar.for

$DEBUG ! Производит дополнительные проверки при компиляции

$STORAGE:2 ! Переменные и константы целого типа:2-х байтовые (integer\*2)

$DECLARE ! Генерация предупреждающих сообщений для необъявленных переменных

include 'fgraph.fi' ! Включение интерфейсов графических функций в начале каждого файла

interface to function getch() ! Объявление ф-ции getch, ее

integer getch [C, alias:'\_getch'] ! атрибутов, типов формальных

end ! параметров

program Demonstr\_devide\_screen ! Объявление головной программы

implicit NONE ! Запретить неявное объявление типов

integer \*2 i2 ! Фиктивная переменная

integer getch [EXTERN]

include 'fgraph.fd' ! Объявление типов графических функций, объявление и задание параметров, определение структур (STRUCTURE) в каждой графической п/п

i2=setvideomode($MAXRESMODE) ! Установить видеорежим с max разрешением

call devide\_graph\_area()

i2=getch() ! Задержка до нажатия клавиши

i2=setvideomode($DEFAULTMODE) ! Установить предыдущий видеорежим

end

! п/п деления экрана на 4 области

subroutine devide\_graph\_area()

include 'fgraph.fd'

integer\*2 i2

integer\*2 nx,ny ! Max кол-во пиксел по X и Y

integer\*2 xdin,ydin ! Координаты общей точки областей

! integer\*2 nr,nc ! Max кол-во строк и столбцов

real kx,ky

record /videoconfig/ vc ! Подключ. структуры videoconfig

! record /rcCOORD/ rc ! и rccoord, определенных в файле fgraph.fd

record /xyCOORD/ xy ! xy - образец типа xycoord

call clearscreen($GCLEARSCREEN)! Очистка экрана

call getvideoconfig(vc) ! Определение видеоконфигурации

kx=75.0 ! Коэф-ты в %, определяющие положе-

ky=80.0 ! ние общей точки областей

nx=vc.numxpixels ! см. файл fgraph.fd

ny=vc.numypixels ! см. файл fgraph.fd

xdin=int2(kx\*nx/100)

ydin=int2(ky\*ny/100)

i2 = setbkcolor($GRAY) ! Установка цвета фона

i2=setcolor(6) ! Цвет красный

i2=rectangle($GBORDER,1,1,nx-1,ny-1) ! Рамка всей области экрана

i2=rectangle($GBORDER,2,2,xdin,ydin) ! Рамка области чертежа

i2=rectangle($GBORDER,2,ydin+1,xdin,ny-2)! Рамка области ввода

i2=rectangle($GBORDER,xdin+1,2,nx-2,ydin)!Рамка информац. области

i2=rectangle($GBORDER,xdin+1,ydin+1,nx-2,ny-2)! Область автора

i2=setcolor(12) ! Цвет светлокрасный

! Оформление области автора

if (registerfonts('D:\MSLIB\\*.fon').LT.0) then ! инициализация

i2=registerfonts('C:\MSLIB\\*.fon') ! шрифтов

end if ! библиотеки

i2=setfont("t'Helv Cyrillic'h18w8b"C) !Установка шрифта

! i2=settextcolor(2) ! Установка цвета текста

! nr=vc.numtextrows

! nc=vc.numtextcols

call moveto(xdin+40,ydin+10, xy)

call outgtext (' ¦ ¦ + +') ! Вывод текста в текущ. позицию

call moveto(xdin+30,ydin+30, xy)

call outgtext ('+ГЕбМГ +.+.')

call moveto(xdin+50,ydin+50, xy)

call outgtext (' 1998')

end

Пример использования функции Getch, возвращающей код нажатой клавиши.

! Программа печати кода нажатой клавиши

$debug

interface to function getch()

integer\*2 getch [C, alias:'\_getch'] ! Интерфейс Си функции

end

integer \*2 getch [EXTERN]

integer \*2 scod

integer i

do i=1,26 ! Отступ в 26 строк

print\*

enddo

1 FORMAT(32x,'КОД КЛАВИШИ - ',I3)

print\*,' НАЖМИТЕ ЛЮБУЮ КЛАВИШУ'

10 print\*

SCOD=getch() ! Отслеживание кода нажатой клавиши

print1,scod ! Печать кода

print\*

print\*,' ( Для выхода в VC - нажмите ^BREAK )' ! Прерывание выполнения программы

print\*,' ( Для продолжения - любую клавишу ) '

go to 10 ! Бесконечный цикл

end

Дополнительные возможности вывода текста.

Текст стандартным шрифтом в любую позицию экрана можно вывести процедурой OutText. Процедура outtext не производит форматирование текста и работает во всех видео режимах. Текст выводится стандартным шрифтом.

Пример:

RECORD /record/ rc

……………………..

dum = SetTextColor (9) ! восстановление цвета текста

call SetTextPosition (20,20,rc)

text = ‘программа демонстрации’

call outtext (text)

Вывод текста (сверху вниз) можно осуществить только в пределах заданной области экрана (текстового окна), которое устанавливается процедурой SetTextWindow. При заполнении окна происходит прокрутка.

Синтаксис call SetTextWindow (r1,c1,r2,c2)

Где: r1,c1- текстовые координаты (строка и столбец) левого верхнего угла прямоугольного окна, r2,c2 – то же правого.

Пример работы с текстовым окном:

! File: textwin.for

! Программа установки текстового окна и ввода элементов массива

! Программа устойчива при некорректном вводе. Выход из программы осуществляется нажатием Esc

$DEBUG ! Производит дополнительные проверки при компиляции

$STORAGE:2 ! Переменные и константы целого типа:2-х байтовые (integer\*2)

$DECLARE ! Генерация предупреждающих сообщений для необъявленных переменных

include 'fgraph.fi' ! Включение интерфейсов графических функций в начале каждого файла

interface to function getch() ! Объявление ф-ции getch, ее

integer getch [C, alias:'\_getch'] ! атрибутов, типов формальных параметров

end

program Demonstr\_Text\_Window ! Объявление головной программы

implicit NONE ! Запретить неявное объявление типов

integer \*2 i2 ! Фиктивная переменная

include 'fgraph.fd' ! Объявление типов графических функций, объявление и задание параметров, определение структур (STRUCTURE) в каждой графической п/п

i2=SetVideoMode($MAXRESMODE) ! Установить видеорежим с max разрешением

call SetTextWin() ! Вызов п/п

i2=SetVideoMode($DEFAULTMODE) ! Установить предыдущий видеорежим

end

subroutine SetTextWin()

include 'fgraph.fd'

integer getch [EXTERN] ! Функция возврата кода нажатой клавиши

integer\*2 i2

integer\*2 i, j, k, m

integer\*2 nr, nc ! Max кол-во строк и столбцов

integer\*2 nx, ny ! Max кол-во пиксел по X и Y

real A(3,3) ! Вспомогательный массив

real kr,kc

character\*12 ValElem

character\*6 vl

character\*1 one

record /videoconfig/ vc ! Подключ. структуры videoconfig

record /rcCOORD/ rc ! и rccoord, определенных в файле fgraph.fd

call ClearScreen($GCLEARSCREEN)! Очистка экрана

call GetVideoConfig(vc) ! Определение видеоконфигурации

nx=vc.NumXPixels ! см. файл fgraph.fd

ny=vc.NumYPixels ! см. файл fgraph.fd

nr=vc.NumTextRows ! см. файл fgraph.fd

nc=vc.NumTextCols ! см. файл fgraph.fd

i2=SetBkColor($GRAY) ! Установка цвета фона

i2=SetColor(2) ! Цвет рисования зеленый

i2=SetTextColor(6) ! Цвет текста красный

kr=0.8

kc=0.75

call SetTextWindow(Int(kr\*nr)+2,2,nr-1,Int(kc\*nc)) ! Текстовое окно

i2=rectangle($GBORDER,1,Int(kr\*ny),Int(kc\*nx),ny-1)! Рамка области ввода

do i=1,3

do j=1,3

call SetTextPosition(1,2,rc) ! Установка начальной позиции

write(vl,11) i, j ! Помещение индексов массива в буфер (внутренний файл) (по формату 11)

call OutText(' Ведите элемент массива А'//vl)

ValElem=' ' ! Символьное обнуление

m=0 ! Числовое обнуление

do k=1,24 ! Количество вводимых цифр

i2=getch() ! Возврат кода нажатой клавиши

if (i2.eq.13) exit ! Нажат ввод

if (i2.eq.27) return ! Нажата Esc (выход из программы)

call GetTextPosition(rc) ! Возврат текущей позиции

call SetTextPosition(rc.row,rc.col,rc) ! и ее установка

write(one,'(a)') char(i2) ! Печать в буфер введенного символа

if (i2.eq.8) then ! Нажата клавиша Backspace

call SetTextPosition(rc.row,rc.col-1,rc)

write(one,'(a)') ' ' ! При нажатии BackSpace удаляется

end if ! последний введенный символ (запись в буфер пробела)

call OutText(one) ! Печать введенного символа

if (i2.eq.8) call SetTextPosition(rc.row,rc.col-1,rc) ! При печати пробела смещаемся влево, поэтому возврат

if (i2.ge.48.and.i2.le.57.or.i2.eq.43.or.i2.eq.45.or.i2.eq.46.or.i2.eq.69.or.i2.eq.101) then

! Перечислены коды следующих символов:0,1,2,3,4,5,6,7,8,9,+,-,.,E,e

m=m+1 ! Отсчет количества введенных символов

if (m.eq.12) exit ! Максимум 12 символов

write(ValElem,'(a,a)') ValElem(1:m),char(i2) ! В буфер помещается первые m символов и введенный

else

if (i2.ne.8) call SetTextPosition(2,2,rc)

if (i2.ne.8) call OutText(' Ошибка ввода! ')

if (i2.eq.8) write(ValElem,'(a,a)') ValElem(1:m),' ' ! При нажатии Backspace

if (i2.eq.8) m=m-1 ! Удаление последнего символа

end if

end do

call SetTextPosition(3,2,rc)

read(ValElem,'(f12.0)') A(i,j) ! Преобразование символа в число (через буфер)

write(5,\*)'Элемент массива ',A(i,j)

i2=getch() ! Задержка до нажатия клавиши

call ClearScreen($GWINDOW) ! Очистка текстового окна

i2=SetColor(8) ! Цвет зеленый

i2=rectangle($GFILLINTERIOR,0,0,nx,ny)

i2=SetColor(2) ! Цвет зеленый

i2=rectangle($GBORDER,1,Int(kr\*ny),Int(kc\*nx),ny-1)

end do

end do

11 format('(',i1,',',i1,') ')

end

Для переноса текста в пределах окна используется функция WrapOn.

Пример:

dum = WrapOn($gwrapon), где:

$gwrapon – символическая константа, указывающая на перенос строки по границе окна

$gwrapoff – обрывание строки по границе окна.

Функция не оказывает влияние на вывод текста шрифтом (с помощью outgtext).

Вставка рисунка из файла

Вставка рисунка формата pcx в любую позицию экрана возможно с использованием файлов библиотек gx\_cl.lib и pcx\_cl.lib, gxint.for, pcxint.for. Вставляемый рисунок должен иметь формат pcx.

! File: ins\_pic.for

! Программа демонстрации вывода рисунка из файла на экран

! ?? Закомментировать строки 156,157 файла gxlib.for

! Компиляция производится с подключением библиотек gx\_cl.lib, pcx\_cl.lib

! (fl ins\_pic.for gx\_cl.lib pcx\_cl.lib)

$DEBUG ! Производит дополнительные проверки при компиляции

$STORAGE:2 ! Переменные и константы целого типа:2-х байтовые

$INCLUDE: 'gxint.for' ! Подключение библиотечных файлов

$INCLUDE: 'pcxint.for'

$include: 'fgraph.fi' ! Включение интерфейсов графических ф-ций

program InsertPcxImage! Объявление головной программы

implicit NONE ! Запретить неявное объявление типов

include 'fgraph.fd' ! Объявление типов графических функций

include 'gxlib.for'

include 'pcxlib.for'

integer\*2 i2 ! Фиктивная переменная

integer\*2 pcxType

character\*20 pcxImage

parameter (pcxType=gxVGA\_12) ! см. стр.178 файл gxlib.for

! parameter (pcxType=gxVGA\_11) ! см. стр.178 файл gxlib.for

parameter (pcxImage='bpaexp.pcx'C) ! Имя файла рисунка

record /videoconfig/ vc

i2 = gxSetDisplay(pcxType) ! Определение используемого типа монитора и режима

i2 = gxSetMode(gxGRAPHICS)

i2=setvideomode($MAXRESMODE) ! Установить видеорежим с max разрешением

call getvideoconfig(vc)

i2 = pcxFileDisplay(pcxImage,100,100,0) ! Вставка рисунка

read\*

i2=setvideomode($DEFAULTMODE) ! Установить предыдущий видеорежим

end

Формат файла PCX

Первые 128 байт каждого файла PCX или PCC содержат заголовок файла изображения. Заголовок определяет ширину и глубину изображения, количество плоскостей, число бит на точку, а также другую информацию, необходимую для многократного создания изображения.

Заголовок несколько изменен в новой версии пакета. В него добавлены три новых поля: PALINFO, SHRES и SVRES. Поле palinfo определяет, является ли изображение цветным (palinfo=1) или в оттенках серого (palinfo=2). Когда программа PC Paintbrush загружает изображение, и она видит, что это серое изображение, программа устанавливает палитру как свою собственную внутреннюю серую палитру. Пакет не обрабатывает палитры подобным способом (текущая палитра ВСЕГДА хранится ВМЕСТЕ с изображением), поэтому он игнорирует это поле. Однако, в целях совместимости, пакет устанавливает флаг palinfo для обозначения цвета.

Два других поля, shres и svres, предназначены для хранения разрешения развертки (сканнера). Они были созданы для того, чтобы не путать замену разрешения экрана на разрешение развертки (сканнера) в полях hres и vres. Прежние пользователи заметят, что это было бы показано как 'Разрешение дисплея: 75 х 75' при использовании утилиты pcxHdr. Теперь поля hres и vres будут содержать экранное разрешение создающего устройства, во всех будущих версиях Paintbrush (1.62 и выше).

Доступ к заголовку может быть произведен при помощи любой функции заголовка пакета. Структура заголовка уже была определена для каждого языка.

Некоторые относящиеся к графике термины

Bitmap‑Способ кодирования изображения пиксел за пикселом.

Buffer‑буфер. Область временного хранения данных, часто используется для компенсации разницы в скорости работы различных компонентов системы. Часто, в качестве буфера используется дополнительная память, зарезервированная для временного хранения данных, которые передаются между центральным процессором системы и периферией (такой, как винчестер, принтер или видеоадаптером). Особенно полезен буфер для компенсации разницы в уровнях интенсивности потоков данных, для обеспечения места размещения данных, когда процессы асинхронны (например, данные переданные в контроллер видеоплаты должны дождаться, когда графический процессор закончит выполнение текущей операции, и считает новую порцию информации), и для сохранения данных в неизменном виде (как буфер для видеокадра). Некоторые буферы являются частью адресуемой памяти центрального процессора системы, другие буферы памяти являются частью периферийных устройств.

Gamma Correction‑перед выводом на дисплей линейные данные RGB должны быть обработаны (скорректированы) для компенсации гаммы (нелинейной составляющей) дисплея.

Pixel‑пиксель. Комбинированный термин, обозначающий элемент изображения, являющийся наименьшим элементом экрана монитора. Другое название - pel. Изображение на экране состоят из сотен тысяч пикселей, объединенных для формирования изображения. Пиксель является минимальным сегментом растровой строки, которая дискретно управляется системой, образующей изображение. С другой стороны, это координата, используемая для определения горизонтальной пространственной позиции пикселя в пределах изображения. Пиксели на мониторе - это светящиеся точки яркого фосфора, являющиеся минимальным элементом цифрового изображения. Размер пикселя не может быть меньше точки, которую монитор может образовать. На цветном мониторе точки состоят из групп триад. Триады формируются тремя различными фосфорами: красным, зеленым и синим. Фосфоры располагаются вдоль сторон друг друга. Пиксели могут отличаться размерами и формой, в зависимости от монитора и графического режима. Количество точек на экране определяются физическим соотношением ширины к высоте трубки.

RGB‑система цветообразования, в которой конечный цвет получается за счет смешения, с различной интенсивностью, трех основных цветов: красного (Red), зеленого (Green) и синего (Blue). Самое известное устройство, которое использует систему RGB, это цветной монитор.

Resolution‑разрешение. Количество пикселей представленное битами в видеопамяти, или адресуемое разрешение. Видеопамять может организовываться соотношением пикселов (битов) по оси x (пикселы на строке) к числу пикселов по оси y (столбцы) и к размеру отводимой памяти на представление глубины цвета. Стандартная видеопамять VGA 640 пикселов на 480 пикселов и, обычно, с глубиной представления цвета 8 бит. Чем выше разрешение, тем более детально изображение, тем больше нужно хранить о нем информации. Но не вся хранимая информация может быть отображена на дисплее.

Texture‑двумерное изображение хранящееся в памяти компьютера или графического акселератора в одном из пиксельных форматов. В случае хранения в сжатом виде на дисках компьютера, текстура может представлят собой обычный бит-мап который мы привыкли видеть в форматах bmp, jpg, gif и т.д. Перед использованием, текстура разворачивается в памяти и может занимать объем в десятки раз больше первоначального размера. Существует порядка двух десятков более или менее стандартизированных пиксельных форматов текстур.