ZX Like Pascal

v.1.0

Автор: Андрей Шарин ака Andrew771, ‘2015-2020

Общее описание

ZX Like Pascal – кросс-компилятор языка программирования Паскаль для компьютера ZX Spectrum. Имеет усеченную версию по сравнению с классическим Паскалем.

Исходная программа на Паскале создается в файле с расширением PRG в любом текстовом редакторе. Для компиляции необходимо выполнить один из нижеследующих пунктов:

1. Запуск в оконном режиме:

- запустить файл ZXLIKEPASCAL.EXE;
- установить необходимый адрес компиляции программы (по умолчанию – 23900);
- установить флагами параметры компиляции:
  = Attributes (вывод с атрибутами/без атрибутов текста, окон, спрайтов, карт);
  = Edge checking for sprites (проверка выхода спрайтов за границы экрана и вывод на экран только помещающейся части спрайта);
  = Virtual screen (вывод на виртуальный/обычный экран текста, спрайтов, карт);
- выбрать необходимый файл PRG, нажав кнопку “Select and compile file”.

2. Запуск из командной строки или из файла BAT:

- запустить файл ZXLIKEPASCAL.EXE с необходимыми выставленными параметрами (флагами):
  имя файла с расширением PRG в кавычках;
  адрес компиляции в кавычках;
  -a – Attributes (вывод с атрибутами текста, окон, спрайтов, карт);
  -b – Edge checking for sprites (проверка выхода границ спрайтов за пределы экрана);
  -c – Virtual screen (вывод на виртуальный экран текста, спрайтов, карт).

Например, zxlikepascal.exe "demo_Z.prg" "23900" -b -c

Программа на Паскале скомпилируется в программу на Ассемблере и запишется в текстовый файл с тем же именем и расширением ASM. Ассемблерный файл можно далее скомпилировать в любом кросс-ассемблере для ZX Spectrum.

НУЖНО ОЧЕНЬ ВНИМАТЕЛЬНО СЛЕДИТЬ ЗА ПАРАМЕТРАМИ КОМПИЛЯЦИИ, ЧТОБЫ ОНИ СТРОГО СООТВЕТСТВОВАЛИ КОДУ В ИСХОДНОЙ ПРОГРАММЕ (УЧЕТ АТРИБУТОВ, ВЫВОД НА ВИРТУАЛЬНЫЙ ЭКРАН).

В случае обнаружения синтаксической ошибки в исходной программе выводится сообщение на экран с описанием ошибки. Конкретное местонахождение ошибки можно посмотреть в файле ERROR.PRG.

Поддерживаются только целочисленные и строковые типы, одномерные и двумерные массивы.

Символьный экран - 64×24. Шрифт используется 4×8, в строке 64 символа. Поддерживается латиница и кириллица, буквы только прописные.

Имеются встроенные операторы вывода окон, познакоместных спрайтов и двумерных карт из элементов-спрайтов.

Спрайты могут выводиться по методам clear, and, or, xor:

- Clear: спрайт выводится на экран, полностью затирая под собой имеющееся на экране изображение;
- And: спрайт выводится на экран, смешивая каждый бит спрайта с каждым битом имеющегося на экране изображения по принципу and. Этот метод может использоваться для вывода спрайта-маски, чтобы затереть изображение под будущий спрайт четко по границе спрайта. В спрайте-маске изображение внутри границы спрайта кодируется нулями, за границами спрайта единицами;
- Or: спрайт выводится на экран, смешивая каждый бит спрайта с каждым битом имеющегося на экране изображения по принципу or. Этот метод может использоваться для вывода спрайта после предварительной очистки области спрайтом-маской внутри границ спрайта;
- Xor: спрайт выводится на экран, смешивая каждый бит спрайта с каждым битом имеющегося на экране изображения по принципу xor.

Карты могут представлять собой ландшафты, лабиринты, игровые поля, составленные из отдельных элементов-спрайтов. Идентификаторы элементов (спрайтов) задаются в двумерном массиве, байт на клетку. Имеется также встроенный оператор поиска элементов на карте по различным критериям.

Окна представляют собой прямоугольные области на экране. Для окон имеются операторы очистки окна на экране, сохранения окна из экрана в память, вывода окна из памяти на экран, попиксельного скроллинга окна влево, вправо, вверх, вниз. Операторы сохранения и вывода окон из памяти на экран можно, например, применять для движения спрайта, запоминая в окне фон перед выводом спрайта. После вывода и перемещения спрайта выводится окно из памяти, восстанавливая фон. Область окна располагается в памяти, начиная с адреса 57344, поэтому при его использовании Ваши

программы должны занимать область до этого адреса. Если окно в памяти не используется, то область памяти до стека свободна для Вашей программы.

Вывод текста, спрайтов и карт можно осуществлять как с атрибутами, так и без них (задается флагом компиляции “Attributes”). В случае неиспользования атрибутов в кодах спрайтов их хранить не нужно.

Вывод текста, спрайтов и карт можно осуществлять как непосредственно на экран, так и на виртуальный экран в памяти для последующего быстрого вывода на реальный экран (задается флагом компиляции “Virtual screen”). Виртуальный экран используется для построения в памяти изображения из многих элементов и его последующего быстрого вывода на экран целиком, для исключения мерцания экрана. Метод рекомендуется применять при наличии перестраиваемого изображения из большого количества элементов для исключения мерцания экрана между кадрами. Виртуальный экран располагается в памяти, начиная с адреса 57344, поэтому при его использовании Ваши программы должны занимать область до этого адреса. Если виртуальный экран не используется, то область памяти до стека свободна для Вашей программы.

Выход за пределы допустимых значений переменных и массивов не отслеживается, за ними должен следить программист (это сделано для увеличения быстродействия программ). Выход спрайтов за пределы экрана отслеживается только при включенном флаге компиляции “Edge checking for sprites”, выводится только помещающаяся на экран часть спрайта.

После последнего END программы на Паскале в файле PRG могут записываться процедуры на Ассемблере, коды спрайтов, карт и другая информация. Вся информация без изменения копируется в файл ASM.

Структура программы

```
program <program name>;

const
      <constant name 1>=<numeric constant 1>;
      <constant name 2>=<numeric constant 2>;
      …
var
      <variable name 1.1>,<variable name 1.2>,…: <type 1> <=volume>;
      <variable name 2.1>,<variable name 2.2>,…: <type 2> <=volume>;
      …
procedure <procedure name 1>;
<statement block>;

procedure <procedure name 2>;
<statement block>;
…

{main program}
begin
<statement block>;
end.

<assembler procedures>
<sprite codes>
```

Комментарии

Комментарии записываются в фигурных скобках { } и могут помещаться в любом месте программы.

Имена

Имена констант, переменных, массивов, процедур должны состоять из любого количества латинских прописных или строчных букв или цифр. Должны начинаться с буквы.
Имена спрайтов и ассемблерных процедур должны быть только **прописными (заглавными)** буквами или цифрами. Должны начинаться с буквы.

Константы

Константы допустимы только числовые со значениями от 0 до 65535.

Переменные и массивы

Допустимые типы переменных и массивов:

`Byte`	Целые числа от 0 до 255.
`Word`	Целые числа от 0 до 65535.
`String[n]`	Символьные строки. n – максимальная длина строки от 1 до 255.
`Array[1..n1] of Byte` `Array[1..n1,1..n2] of Byte`	Одномерные и двумерные массивы целых чисел от 0 до 255. n1,n2 – максимальные индексы от 1 до 65535.
`Array[1..n1] of Word` `Array[1..n1,1..n2] of Word`	Одномерные и двумерные массивы целых чисел от 0 до 65535. n1,n2 – максимальные индексы от 1 до 65535.
`Array[1..n1] of String[n2]`	Одномерные массивы символьных строк. n1 – максимальный индекс от 1 до 65535, n2 – максимальная длина строки от 1 до 255.

Нижняя граница индексов массивов всегда равна 1.

При объявлении переменных и массивов можно (но не обязательно) инициализировать их значения, например:

```
var
      a: byte = 36;
      b: string[10] = 'привет';
      m: array[2,3] of word = [1,4,6, 3,8,5];
```

Арифметические выражения

`<arithmetic expression>` - арифметическое выражение состоит из любого количества аргументов и операций над ними. Аргументами могут быть числа, константы, числовые переменные и ячейки числовых массивов. Результатом арифметического выражения является вычисленное числовое значение.

Допустимые операции:

`+`	Сложение
`-`	Вычитание
`*`	Умножение
`/`	Целочисленное деление
`%`	Остаток от деления
`( )`	Приоритет вычислений
`Random(x)`	Случайное число от 0 до x включительно. x может быть также арифметическим выражением

Строковые выражения

`<string expression>` - строковое выражение состоит из любого количества аргументов и операций над ними. Аргументами могут быть строковые переменные, ячейки строковых массивов, а также непосредственно строки символов, заключенные в апострофы (например, 'abc123'). Результатом строкового выражения является вычисленная строка.

Допустимые операции:

`+`	Конкатенация (сложение)

Логические выражения

<logical expression> - логическое выражение состоит из любого количества арифметических выражений и операций над ними. Не допускается использование строковых выражений. Результатом логического выражения является флаг «верно»/«не верно».
Скобки для изменения приоритета операций не поддерживаются.

Допустимые операции:

=	Равно	Приоритет 1
>	Больше	Приоритет 1
>=	Больше или равно	Приоритет 1
<	Меньше	Приоритет 1
<=	Меньше или равно	Приоритет 1
<>	Не равно	Приоритет 1
And	Логическое И	Приоритет 2
Or	Логическое ИЛИ	Приоритет 2

Блоки операторов

<statement block> - операторы, выполняющиеся подряд, разделяются точкой с запятой и объединяются в блоки, начинающиеся ключевым словом BEGIN и заканчивающиеся END. Если блок состоит из одного оператора, то ключевые слова BEGIN и END не обязательны.

Процедуры

Процедуры допустимы только без параметров и локальных переменных.

Операторы (по алфавиту)

<numeric variable>:=<arithmetic expression>; <string variable>:=<string expression>;	Присваивание значения переменной.
ArrayClear(<array name>);	Очистка массива, заполнение нулями числового массива или пустыми строками строкового массива. <array name> - имя массива без параметров.
Asm(<procedure name>);	Вызов процедуры на ассемблере. Тело ассемблерной процедуры размещается после последнего END программы на Паскале. При выходе из процедуры по ассемблерной команде RET управление передается следующему оператору Паскаля. <procedure name> - имя процедуры обязательно заглавными буквами. В ассемблерной процедуре доступны переменные Паскаля. Для вызова переменной необходимо указать знак нижнего подчеркивания и имя переменной обязательно заглавными буквами. Например: переменная из Паскаля abc01 в ассемблере может быть вызвана по имени _ABC01.
Border(<x>);	Установка цвета бордюра. <x> - номер цвета бордюра, может быть <arithmetic expression>.

Case <numeric variable> of <value 1.1>,<value 1.2>,…: <statement block 1>; <value 2.1>,<value 2.2>,…: <statement block 2>; … else <statement block N> end;	Множественная альтернатива (множественный выбор). Если <numeric variable> равно <value 1.1>, или <value 1.2>, или <value 1.3>, или … (количество значений и условий может быть любым), то выполняется <statement block 1>. Если <numeric variable> равно <value 2.1>, или <value 2.2>, или <value 2.3>, или …, то выполняется <statement block 2>. Если ни одно из условий не выполнено, то выполняется <statement block N> после конструкции ELSE. Конструкция ELSE не обязательна.
ClrScr;	Очистка экрана и установка текущих атрибутов на нем.
Color(<x>);	Установка текущих цветов изображения и фона (текущих атрибутов). <x> - байт атрибутов, может быть <arithmetic expression>. Байт атрибутов: 7-й бит – FLASH; 6-й бит – BRIGHT; 5,4,3-й биты – PAPER; 2,1,0-й биты – INK. Для задания необходимых битов можно воспользоваться формулой: x=128*flash+64*bright+8*paper+ink. Пример: flash=1, bright=0, paper=2, ink=6, тогда Color(128*1+64*0+8*2+6)
Delay(<x>);	Пауза длительностью x/50 секунды. <x> - длительность паузы, кратная x/50 секунды, может быть <arithmetic expression>.
For <numeric variable>:=<arithmetic expression 1> to <arithmetic expression 2> do <statement block>; For <numeric variable>:=<arithmetic expression 1> downto <arithmetic expression 2> do <statement block>;	Цикл, выполняемый определенное количество раз. <numeric variable> - переменная, текущее значение счетчика цикла, доступно другим операторам внутри цикла. <arithmetic expression 1> - начальное значение счетчика. <arithmetic expression 2> - конечное значение счетчика. Если используется конструкция DO, то счетчик возрастает на 1. Если DOWNTO, то убывает на 1.
GotoXY(<x>,<y>);	Установка курсора в символьную позицию <x>,<y> экрана. <x> - столбец экрана, от 0 до 63, может быть <arithmetic expression>. <y> - строка экрана, от 0 до 23, может быть <arithmetic expression>.
If <logical expression> then <statement block 1> else <statement block 2>;	Оператор условия. Если <logical expression> верно, то выполняется <statement block 1>. Иначе выполняется <statement block 2> после конструкции ELSE. Конструкция ELSE не обязательна.
MapPut1x1(<x>,<y>);	Вывод карты на экран с клетки <x>,<y> карты. Параметры карты должны быть предварительно заданы в операторе MapSet. Каждый спрайт элемента карты занимает 1×1 знакоместо. <x> - горизонтальная координата клетки карты, может быть <arithmetic expression>. <y> - вертикальная координата клетки карты, может быть <arithmetic expression>.

`MapPut2x2(<x>,<y>);`	Вывод карты на экран с клетки <x>,<y> карты. Параметры карты должны быть предварительно заданы в операторе MapSet. Каждый спрайт элемента карты занимает 2×2 знакоместа. <x> - горизонтальная координата клетки карты, может быть <arithmetic expression>. <y> - вертикальная координата клетки карты, может быть <arithmetic expression>.
`MapSearch(<x>,<y>,<id element>,<id condition>, <distance>,<value variable>,<x variable>, <y variable>);`	Поиск элемента (значения клетки) на карте вокруг исходной клетки <x>,<y> карты. Параметры карты должны быть предварительно заданы в операторе MapSet. <x> - горизонтальная координата исходной клетки карты, может быть <arithmetic expression>. <y> - вертикальная координата исходной клетки карты, может быть <arithmetic expression>. <id element> - значение элемента (клетки карты), которое необходимо найти, может быть <arithmetic expression>. <id condition> - условие поиска элемента (0 – равно значению элемента, 1 – не равно значению элемента, 2 – меньше значения элемента, 3 – больше или равно значению элемента), может быть <arithmetic expression>. <distance> - расстояние в клетках от исходной клетки карты, может быть <arithmetic expression>. <value variable> - имя переменной типа Byte, в которую запишется количество найденных элементов (если не найдены, то 0). <x variable> - имя переменной типа Byte, в которую запишется горизонтальная координата ближайшего найденного элемента (если не найден, то 0). <y variable> - имя переменной типа Byte, в которую запишется вертикальная координата ближайшего найденного элемента (если не найден, то 0).

`MapSet(<array name>,<sprite bitmap>,<screen x>,<screen y>,<width>,<height>);`	Установка параметров карты. <array name> - имя массива без параметров, содержащего коды клеток карты. Массив должен быть обязательно двумерный типа Byte. Первый индекс – горизонтальная координата клетки, второй индекс – вертикальная координата клетки. Коды элементов карты должны быть последовательными от 0 до максимума, не более 255. <sprite bitmap> - метка начала спрайтов элементов карты, имя метки обязательно заглавными буквами. Коды спрайтов для элементов карты размещаются после последнего END программы на Паскале. <screen x> - горизонтальная координата знакоместа на экране, с которой выводится карта, от 0 до 31, может быть <arithmetic expression>. <screen y> - вертикальная координата знакоместа на экране, с которой выводится карта, от 0 до 22 обязательно кратная 2, может быть <arithmetic expression>. <width> - количество клеток по ширине для выводимого фрагмента карты, может быть <arithmetic expression>. <height> - количество клеток по высоте для выводимого фрагмента карты, может быть <arithmetic expression>.
`Randomize;`	Инициализация генератора случайных чисел.
`Read(<any variable>,<any variable>…);`	Чтение значений переменных, вводимых пользователем с клавиатуры. Количество и тип переменных могут быть любыми. Если пользователь вводит некорректное значение (например, буквы вместо цифр для числовых переменных, слишком длинную строку), то программа стирает и запрашивает значение заново, при этом работа программы не приостанавливается.
`ReadKey(<byte variable 1>,<byte variable 2>);`	Чтение одновременно двух нажатых клавиш. Коды клавиш помещаются в указанные переменные обязательно типа Byte. Если клавиша не нажата, то помещается 0. Работа программы не приостанавливается, происходит сразу переход на следующий оператор. <byte variable 1> - имя переменной типа Byte, в которую читается код 1-й нажатой клавиши. <byte variable 2> - имя переменной типа Byte, в которую читается код 2-й нажатой клавиши.
`Readln(<any variable>,<any variable>…);`	То же, что и Read, только в конце производится перевод курсора на следующую строку.
`Repeat <statement block> until <logical expression>;`	Цикл с постусловием. Блок операторов <statement block> выполняется до тех пор, пока не будет верно условие <logical expression>
`ScreenThaw;`	Растворение (постепенное стирание) изображения на экране. Атрибуты на экране не изменяются.

SoundEffect(<n>,<t>,<f>);	Звуковой эффект на бипере. <n> - количество волн, может быть <arithmetic expression>. <t> - длительность звучания одной волны, может быть <arithmetic expression>. <f> - начальная частота волны, может быть <arithmetic expression>. Примеры: SoundEffect(10,35,0) – звук пулемета; SoundEffect(1,200,255) – звук бомбы.
SpritePutAnd(<sprite name>,<color>,<x>,<y>);	Вывод спрайта на экран по принципу AND с имеющимся под ним изображением на экране. <sprite name> - метка спрайта, имя метки обязательно заглавными буквами. Коды спрайтов размещаются после последнего END программы на Паскале. <color> - цвет спрайта, байт атрибутов, может быть <arithmetic expression>. Байт атрибутов: 7-й бит – FLASH; 6-й бит – BRIGHT; 5,4,3-й биты – PAPER; 2,1,0-й биты – INK. <x> - горизонтальная координата знакоместа на экране, с которой выводится спрайт, от 0 до 31, может быть <arithmetic expression>. <y> - вертикальная координата знакоместа на экране, с которой выводится спрайт, от 0 до 23, может быть <arithmetic expression>.
SpritePutClear(<sprite name>,<color>,<x>,<y>);	Вывод спрайта на экран, зачищая имеющееся под ним изображение на экране. <sprite name> - метка спрайта, имя метки обязательно заглавными буквами. Коды спрайтов размещаются после последнего END программы на Паскале. <color> - цвет спрайта, байт атрибутов, может быть <arithmetic expression>. Байт атрибутов: 7-й бит – FLASH; 6-й бит – BRIGHT; 5,4,3-й биты – PAPER; 2,1,0-й биты – INK. <x> - горизонтальная координата знакоместа на экране, с которой выводится спрайт, от 0 до 31, может быть <arithmetic expression>. <y> - вертикальная координата знакоместа на экране, с которой выводится спрайт, от 0 до 23, может быть <arithmetic expression>.

`SpritePutMirrorAnd(<sprite name>,<color>,<x>,<y>);`	Вывод спрайта на экран, зеркального относительно вертикальной оси, по принципу AND с имеющимся под ним изображением на экране. <sprite name> - метка спрайта, имя метки обязательно заглавными буквами. Коды спрайтов размещаются после последнего END программы на Паскале. <color> - цвет спрайта, байт атрибутов, может быть <arithmetic expression>. Байт атрибутов: 7-й бит – FLASH; 6-й бит – BRIGHT; 5,4,3-й биты – PAPER; 2,1,0-й биты – INK. <x> - горизонтальная координата знакоместа на экране, с которой выводится спрайт, от 0 до 31, может быть <arithmetic expression>. <y> - вертикальная координата знакоместа на экране, с которой выводится спрайт, от 0 до 23, может быть <arithmetic expression>.
`SpritePutMirrorClear(<sprite name>,<color>,<x>,<y>);`	Вывод спрайта на экран, зеркального относительно вертикальной оси, зачищая имеющееся под ним изображение на экране. <sprite name> - метка спрайта, имя метки обязательно заглавными буквами. Коды спрайтов размещаются после последнего END программы на Паскале. <color> - цвет спрайта, байт атрибутов, может быть <arithmetic expression>. Байт атрибутов: 7-й бит – FLASH; 6-й бит – BRIGHT; 5,4,3-й биты – PAPER; 2,1,0-й биты – INK. <x> - горизонтальная координата знакоместа на экране, с которой выводится спрайт, от 0 до 31, может быть <arithmetic expression>. <y> - вертикальная координата знакоместа на экране, с которой выводится спрайт, от 0 до 23, может быть <arithmetic expression>.
`SpritePutMirrorOr(<sprite name>,<color>,<x>,<y>);`	Вывод спрайта на экран, зеркального относительно вертикальной оси, по принципу OR с имеющимся под ним изображением на экране. <sprite name> - метка спрайта, имя метки обязательно заглавными буквами. Коды спрайтов размещаются после последнего END программы на Паскале. <color> - цвет спрайта, байт атрибутов, может быть <arithmetic expression>. Байт атрибутов: 7-й бит – FLASH; 6-й бит – BRIGHT; 5,4,3-й биты – PAPER; 2,1,0-й биты – INK. <x> - горизонтальная координата знакоместа на экране, с которой выводится спрайт, от 0 до 31, может быть <arithmetic expression>. <y> - вертикальная координата знакоместа на экране, с которой выводится спрайт, от 0 до 23, может быть <arithmetic expression>.

SpritePutMirrorXor(<sprite name>,<color>,<x>,<y>);	Вывод спрайта на экран, зеркального относительно вертикальной оси, по принципу XOR с имеющимся под ним изображением на экране. <sprite name> - метка спрайта, имя метки обязательно заглавными буквами. Коды спрайтов размещаются после последнего END программы на Паскале. <color> - цвет спрайта, байт атрибутов, может быть <arithmetic expression>. Байт атрибутов: 7-й бит – FLASH; 6-й бит – BRIGHT; 5,4,3-й биты – PAPER; 2,1,0-й биты – INK. <x> - горизонтальная координата знакоместа на экране, с которой выводится спрайт, от 0 до 31, может быть <arithmetic expression>. <y> - вертикальная координата знакоместа на экране, с которой выводится спрайт, от 0 до 23, может быть <arithmetic expression>.
SpritePutOr(<sprite name>,<color>,<x>,<y>);	Вывод спрайта на экран по принципу OR с имеющимся под ним изображением на экране. <sprite name> - метка спрайта, имя метки обязательно заглавными буквами. Коды спрайтов размещаются после последнего END программы на Паскале. <color> - цвет спрайта, байт атрибутов, может быть <arithmetic expression>. Байт атрибутов: 7-й бит – FLASH; 6-й бит – BRIGHT; 5,4,3-й биты – PAPER; 2,1,0-й биты – INK. <x> - горизонтальная координата знакоместа на экране, с которой выводится спрайт, от 0 до 31, может быть <arithmetic expression>. <y> - вертикальная координата знакоместа на экране, с которой выводится спрайт, от 0 до 23, может быть <arithmetic expression>.
SpritePutXor(<sprite name>,<color>,<x>,<y>);	Вывод спрайта на экран по принципу XOR с имеющимся под ним изображением на экране. <sprite name> - метка спрайта, имя метки обязательно заглавными буквами. Коды спрайтов размещаются после последнего END программы на Паскале. <color> - цвет спрайта, байт атрибутов, может быть <arithmetic expression>. Байт атрибутов: 7-й бит – FLASH; 6-й бит – BRIGHT; 5,4,3-й биты – PAPER; 2,1,0-й биты – INK. <x> - горизонтальная координата знакоместа на экране, с которой выводится спрайт, от 0 до 31, может быть <arithmetic expression>. <y> - вертикальная координата знакоместа на экране, с которой выводится спрайт, от 0 до 23, может быть <arithmetic expression>.
TextBackGround(<x>);	Установка текущего цвета фона. <x> - цвет от 0 до 7, может быть <arithmetic expression>.
TextColor(<x>);	Установка текущего цвета символов. <x> - цвет от 0 до 7, может быть <arithmetic expression>.
VirtScreenClear;	Очистка виртуального экрана и установка текущих атрибутов на нем.
VirtScreenPut;	Вывод виртуального экрана на экран.

`While <logical expression> do <statement block>;`	Цикл с предусловием. Блок операторов <statement block> выполняется, пока верно условие <logical expression>
`WindowClear;`	Очистка окна и установка текущих атрибутов в нем. Параметры окна должны быть предварительно заданы в операторе WindowSet.
`WindowGet;`	Копирование в память изображения окна с экрана. Параметры окна должны быть предварительно заданы в операторе WindowSet.
`WindowPut;`	Вывод на экран изображения окна из памяти, запомненного ранее оператором WindowGet. Параметры окна могут быть предварительно заданы в операторе WindowSet, если они отличаются от параметров для WindowGet.
`WindowScrollDown;`	Скроллинг окна вниз на 1 пиксель. Параметры окна должны быть предварительно заданы в операторе WindowSet.
`WindowScrollLeft;`	Скроллинг окна влево на 1 пиксель. Параметры окна должны быть предварительно заданы в операторе WindowSet.
`WindowScrollRight;`	Скроллинг окна вправо на 1 пиксель. Параметры окна должны быть предварительно заданы в операторе WindowSet.
`WindowScrollUp;`	Скроллинг окна вверх на 1 пиксель. Параметры окна должны быть предварительно заданы в операторе WindowSet.
`WindowSet(<x>,<y>,<width>,<height>);`	Установка текущих параметров окна. <x> - горизонтальная координата знакоместа на экране, с которого выводится окно, от 0 до 31, может быть <arithmetic expression>. <y> - вертикальная координата знакоместа на экране, с которого выводится окно, от 0 до 23, может быть <arithmetic expression>. <width> - ширина окна в знакоместах, может быть <arithmetic expression>. <height> - высота окна в знакоместах, может быть <arithmetic expression>.
`Write(<any expression>,<any expression>…);`	Вывод на экран значений переменных и выражений. Количество, тип переменных и выражений могут быть любыми.
`Writeln(<any expression>,<any expression>…);`	То же, что и Write, только в конце производится перевод курсора на следующую строку.

Спрайты

Спрайты могут иметь произвольную форму размером не более 255 знакомест, не обязательно прямоугольную.
Коды спрайтов размещаются после последнего END программы на Паскале.
Формат спрайтов:

```
NAME        DEFB 6
            DEFB 0,1,  63,96,204,152,145,163,130,139
            ...
```

NAME – имя спрайта заглавными буквами.
DEFB 6 – количество знакомест в спрайте.
DEFB 0,1, 63,96,204,152,145,163,130,139 – описание знакоместа, где:
0 – вертикальная координата знакоместа (0..23);
1 – горизонтальная координата знакоместа (0..31);
63,96,204,152,145,163,130,139 – восемь байт изображения знакоместа сверху вниз.

Количество описаний знакомест в спрайте должно соответствовать количеству знакомест. В данном примере должно быть 6 описаний знакомест.
Цвет спрайта, если включен флаг компиляции “Attributes”, задается в операторах вывода спрайта SpritePut...

Спрайты карты

Коды спрайтов карты размещаются после последнего END программы на Паскале. Спрайты идут последовательно в памяти, согласно кодам элементов карты. Коды элементов карты должны быть последовательными от 0 до максимума, не более 255. Количество спрайтов должно строго соответствовать количеству кодов карты.

Для карты из спрайтов 1×1 знакоместо один спрайт занимает 9 байт с атрибутами или 8 байт без атрибутов.
Формат одного спрайта карты 1×1 знакоместо:

```
NAME
            DEFB 3
            DEFB 7,31,63,127,120,247,252,249
```

NAME – метка начала спрайтов элементов карты, имя метки обязательно заглавными буквами. Метка ставится вначале всех спрайтов карты, одна для всех спрайтов карты.
DEFB 3 – атрибут для знакоместа спрайта; отсутствует в случае отключенного флага компиляции “Attributes”.
DEFB 7,31,63,127,120,247,252,249 – восемь байт изображения знакоместа сверху вниз.

Для карты из спрайтов 2×2 знакоместа один спрайт занимает 36 байт с атрибутами или 32 байта без атрибутов.
Формат одного спрайта карты 2×2 знакоместа:

```
NAME
            DEFB 3,7,23,16
            DEFB 7,31,63,127,120,247,252,249
            DEFB 192,240,248,252,124,62,62,62
            DEFB 243,243,120,127,63,31,7,0
            DEFB 62,62,60,252,248,240,192,0
```

NAME – метка начала спрайтов элементов карты, имя метки обязательно заглавными буквами. Метка ставится вначале всех спрайтов карты, всегда одна для всех спрайтов карты.
DEFB 3,7,23,16 – атрибуты для каждого из 4-х знакомест спрайта, могут быть различными; отсутствуют в случае отключенного флага компиляции “Attributes”.
Далее идет по 8 байт изображения для каждого из 4-х знакомест спрайта.

Таблица кодов символов и клавиш

Код (дес.)	Символ	Клавиша
13		Enter
16		Symbol Shift
17		Caps Shift
32	Space	Space
33	!	
34	"	
35	#	
36	$	
37	%	
38	&	
39	'	
40	(	
41	)	
42	*	
43	+	
44	,	
45	-	
46	.	
47	/	
48	0	0
49	1	1
50	2	2
51	3	3
52	4	4
53	5	5
54	6	6
55	7	7
56	8	8
57	9	9
58	:	
59	;	
60	<	
61	=	
62	>	
63	?	
64	@	
65	A	A
66	B	B
67	C	C
68	D	D
69	E	E
70	F	F
71	G	G
72	H	H
73	I	I
74	J	J
75	K	K
76	L	L
77	M	M
78	N	N
79	O	O
80	P	P
81	Q	Q
82	R	R
83	S	S
84	T	T

Код (дес.)	Символ	Клавиша
85	U	U
86	V	V
87	W	W
88	X	X
89	Y	Y
90	Z	Z
91	[	
92	\	
93	]	
94	^	
95	_	
96	~	
97	†	
98	‡	
99	€	
100	©	
101	¬	
102	°	
103	±	
104	┃	
105	─	
106	┼	
107	┌	
108	┐	
109	└	
110	┘	
111	<>	
112	£	
113	{	
114	}	
115	А	
116	Б	
117	В	
118	Г	
119	Д	
120	Е	
121	Ж	
122	З	
123	И	
124	Й	
125	К	
126	Л	
127	М	
128	Н	
129	О	
130	П	
131	Р	
132	С	
133	Т	
134	У	
135	Ф	
136	Х	
137	Ц	
138	Ч	
139	Ш	
140	Щ	
141	Ъ	
142	Ы	
143	Ь	

Код (дес.)	Символ	Клавиша
144	Э	
145	Ю	
146	Я	