Модуль ABCPascal

Модули языка Pascal

Модуль - это автономно компилируемая программная единица.

Модуль может быть подключен к любой паскаль-программе или к другому модулю (программной единице).

Модуль включает в себя различные разделы описаний (типов, констант, переменных, процедур и функций).

Модуль делится на две основные части: интерфейсную (interface) и исполняемую (implementation). В интерфейсной части даются описания, которые могут использоваться в программной единице, к которой подключен модуль. В исполняемой части приводятся описания, доступные только внутри модуля.

Кроме описаний, модуль может содержать некоторые операторы, подготавливающие условия для использования модуля (например, присваивание переменным начальных значений или установление связи между программными и физическими именами файлов). Такие операторы содержатся в части модуля, которая называется **инициализирующей.** Эта часть не является обязательной.

В Объектном Паскале модуль также может иметь **завершительную часть (finalization)**, состоящую из операторов, обеспечивающих грамотное завершение программной единицы, использующей модуль.

Модули языка Pascal

Структура модулей Паскаля

Unit <имя_модуля>; {начинает заголовок модуля}
interface <интерфейсная часть>;
implementation < исполняемая часть >;
begin
<инициирующая часть>;
end . {признак конца модуля}.

Interface – зарезервированное слово (интерфейс), начинает интерфейсную часть модуля;

implementation – зарезервированное слово (выполнение); начинает исполняемую часть модуля;

begin – зарезервированное слово; начинает инициирующую часть модуля;

конструкция begin <инициирующая часть> необязательна;

Таким образом, модуль Паскаля состоит из заголовка и трех составных частей, любая из которых может быть пустой.

Стандартные модули языка Pascal

Pascal имеет восемь стандартных модулей

SYSTEM входят все процедуры и функции авторской версии языка Pascal, подпрограммы стандартного Паскаля, а также много дополнительных подпрограмм общего характера, ориентированные на конкретную операционную среду.

DOS содержит средства доступа к операционной системе.

CRT обеспечивает практически полный спектр возможностей для доступа к экрану дисплея в текстовом режиме.

PRINTER содержит единственный интерфейсный элемент - переменную Lst. Использование этой переменной в стандартных процедурах Write и WriteLn приводит к выводу информации на печать.

OVERLAY предоставляет средства для организации так называемых оверлейных программ, позволяющих обеспечить достаточно эффективное выполнение больших программных систем, размер которых превышает объем доступной оперативной памяти.

GRAPH объединяет многочисленные программные средства управления графическим режимом работы дисплея. позволяет создавать разнообразные и эффективные графические программы.

TURBO3 И GRAPH3 обеспечивают совместимость с данной версией системы Pascal тех программ, которые были разработаны для ранней версии 3.0.

Модули языка Pascal

Информация, выводимая на экран видеодисплея, может быть двух видов:

- **текстовая,** т.е. состоящая из знаков алфавита, цифр и специальных символов,
- **графическая,** т.е. чертежи, рисунки, графики, различные шрифты.

Поэтому выделяются два режима работы видеотерминала текстовый и графический.

CRT обеспечивает практически полный спектр возможностей для доступа к экрану дисплея в текстовом режиме.

GRAPH объединяет многочисленные программные средства управления графическим режимом работы дисплея. позволяет создавать разнообразные и эффективные графические программы.

Управление экраном в текстовом режиме

Модуль Crt

Модуль **Crt** содержит подпрограммы предназначенные:

– для задержки экрана и его очистки,

– для вывода текста в цвете,

– для вывода его в заданном местоположении,

– для считывания данных с нажатой клавиши,

– для вывода звукового сигнала

– и многое другое, предназначенное для вывода информации в текстовом режиме.

Модуль Crt

В *текстовом режиме* единицей вывода информации служит **символ**.

На экране каждый символ занимает одно ***знакоместо*** — прямоугольный участок размером 8x16 ***пикселей*** (зёрен экрана).

Во весь экран помещается 80x25 *знакомест*.

Курсор (мигающий прямоугольник) помечает то место на экране, куда по умолчанию будет осуществлён вывод очередного символа, — **текущую** *позицию*.

Для определения текущей *позиции курсора* предназначена сетка координат, мысленно

Левое верхнее знакоместо имеет координаты (1,1), правое верхнее — (1,80), левое нижнее — (25,1) и правое нижнее — соответственно (25,80).

Процедуры модуля Crt

Активная область ввода / вывода

Процедура **Window**(x1, y1, x2, y2 : Byte) создаст на экране окно с координатами левого верхнего угла в точке (x1, y1) и координатами правого нижнего угла в точке (x2, y2). Теперь активная область экрана будет ограничена этим окном. Текущие координаты курсора будут отсчитываться не от левого верхнего угла экрана, а от левого верхнего угла этого окна.

Очистка

Процедура **ClrScr** очистит весь экран (или активное окно),курсор будет помещён в верхний левый его угол.

Процедура **ClrEol** очистит текущую строку, начиная с текущей позиции курсора и до правого края экрана (окна).

Процедура **DelLine** удалит строку, в которой находится курсор.

Процедура **InsLine** очистит текущую строку целиком. Курсор останется на прежней позиции.

Процедуры модуля Crt

Цвета

Процедура **TextBackground(color : Byte)** установит цвет **фона.**

Процедура **TextColor(color : Byte)** установит цвет выводимого **текста**.

Замечание: Количество доступных цветов (для экрана и символов) всего 16. Они кодируются числами от 0 до 15. Вместо номера цвета возможно использовать соответствующую констант

Константа	Номер	Цвет	Константа	Номер	Цвет
Black	0	Чёрный	DarkGray	8	Тёмно-серый
Blue	1	Синий	LightBlue	9	Ярко-синий
Green	2	Зелёный	LightGreen	10	Ярко-зелёный
Cyan	3	Голубой	LightCyan	11	Ярко-голубой
Red	4	Красный	LightRed	12	Розовый
Magenta	5	Фиолетовый	LightMagenta	13	Ярко-фиолетовый
Brown	6	Коричневый	Yellow	14	Жёлтый
LightGray	7	Светло-серый	White	15	Белый

Процедуры модуля Crt

Звук

Процедура Sound(hz : Word) включит звуковой сигнал с частотой hz Герц.
Процедура NoSound выключит звуковой сигнал.

Позиционирование

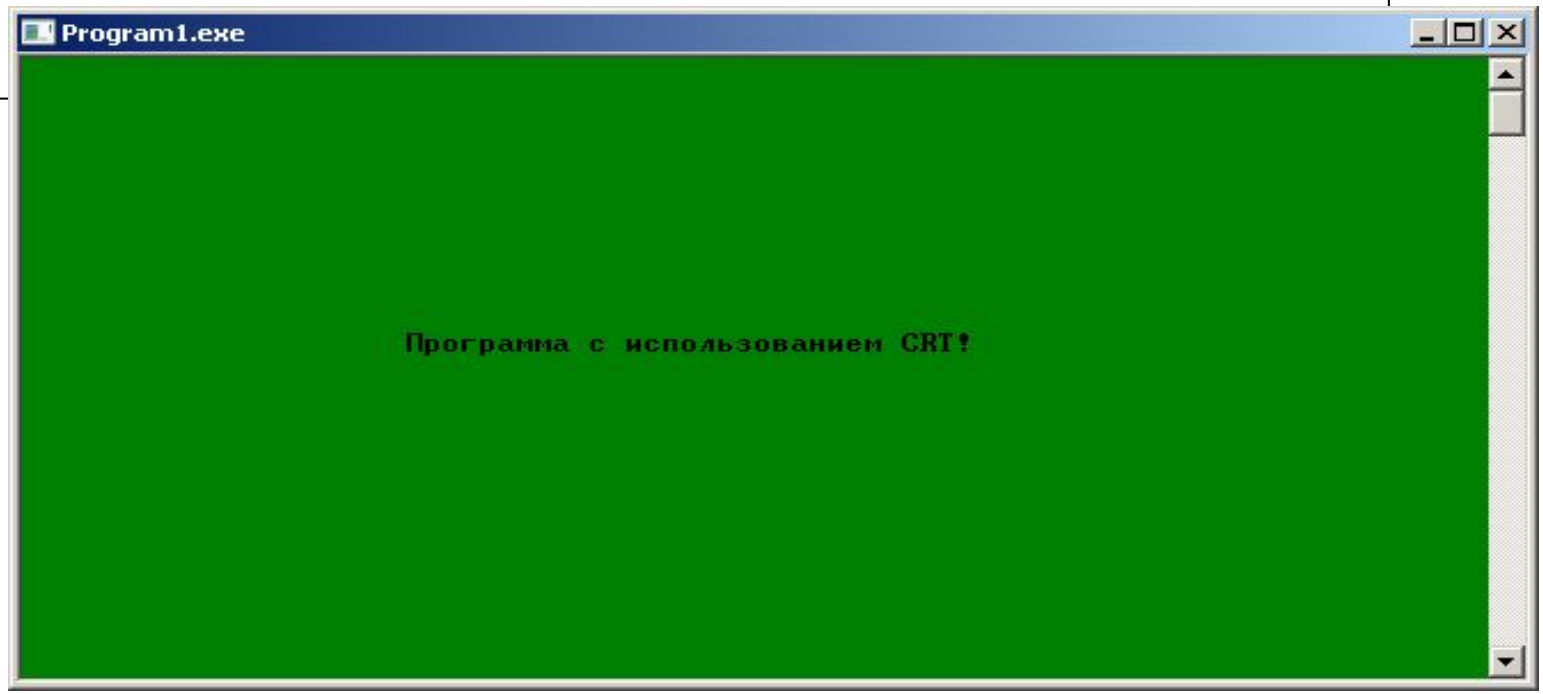
Процедура GotoXY(x, y : Byte) переместит курсор в заданную позицию в пределах текущего окна (экрана).
Функция WhereX : Byte вычислит положение курсора в текущем окне (или на экране): его горизонтальную составляющую. Напомним, что координата X отсчитывается от левого края экрана (окна).
Функция WhereY : Byte вычислит положение курсора в текущем окне (или на экране): его вертикальную составляющую. Напомним, что координата Y отсчитывается от верхнего края экрана (окна).

Ожидание

Процедура Delay(ms : Word) приостановит исполнение программы на ms миллисекунд.
Функция KeyPressed : Boolean отслеживает нажатия клавиш (на клавиатуре).
Функция ReadKey : Char возвращает код символа, чья клавиша (или комбинация клавиш) была нажата.

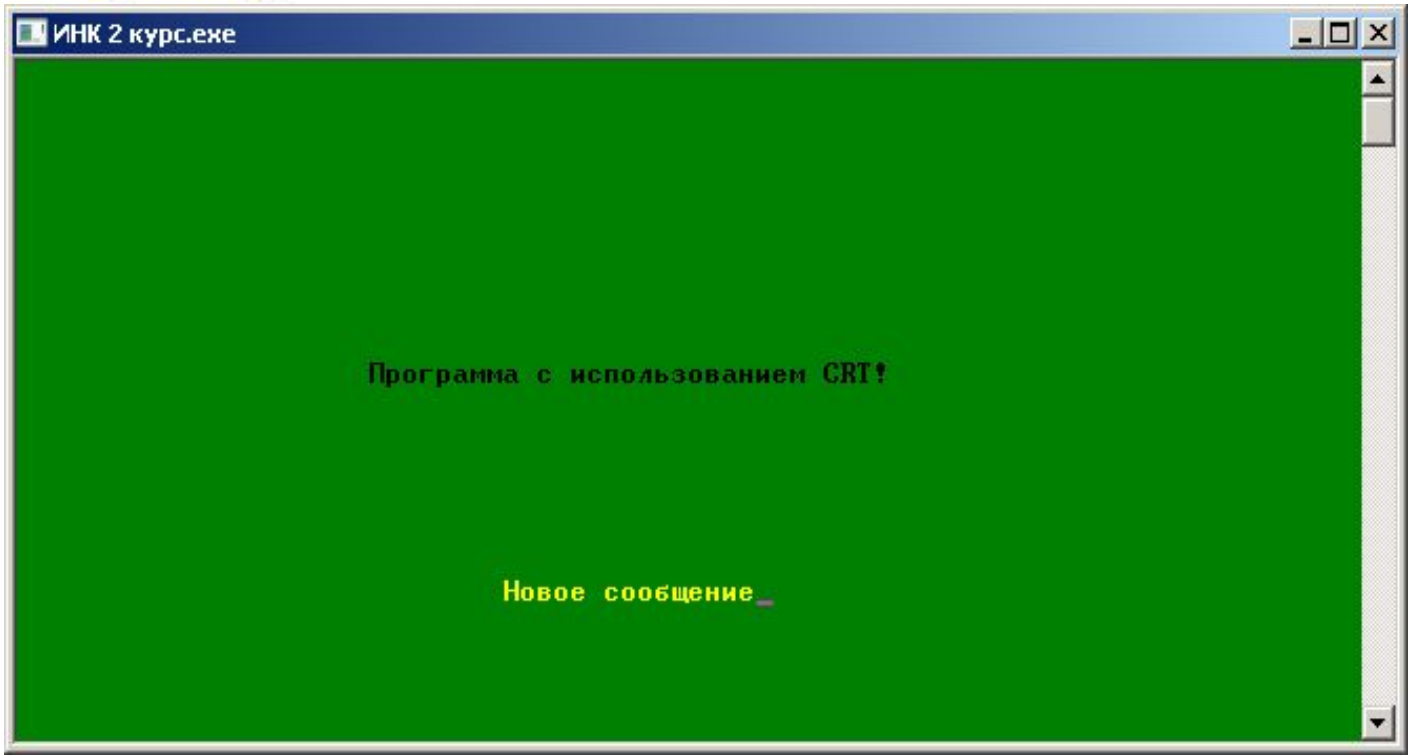
Пример №1 (Вывод сообщения)

```
uses crt; //Подключение модуля CRT
begin
  textbackground(2); // установка цвета фона(Зеленого)
  clrscr; // очистка экрана и установка курсора на позицию (1;1)
  textcolor(0); // установка цвета текста (Черный)
  gotoxy(22, 12); // установка курсора на позицию 22,11
  write('Программа с использованием CRT!'); // вывод текста на экран
readln
end.
```



Пример №2 Вывод сообщения по таймеру

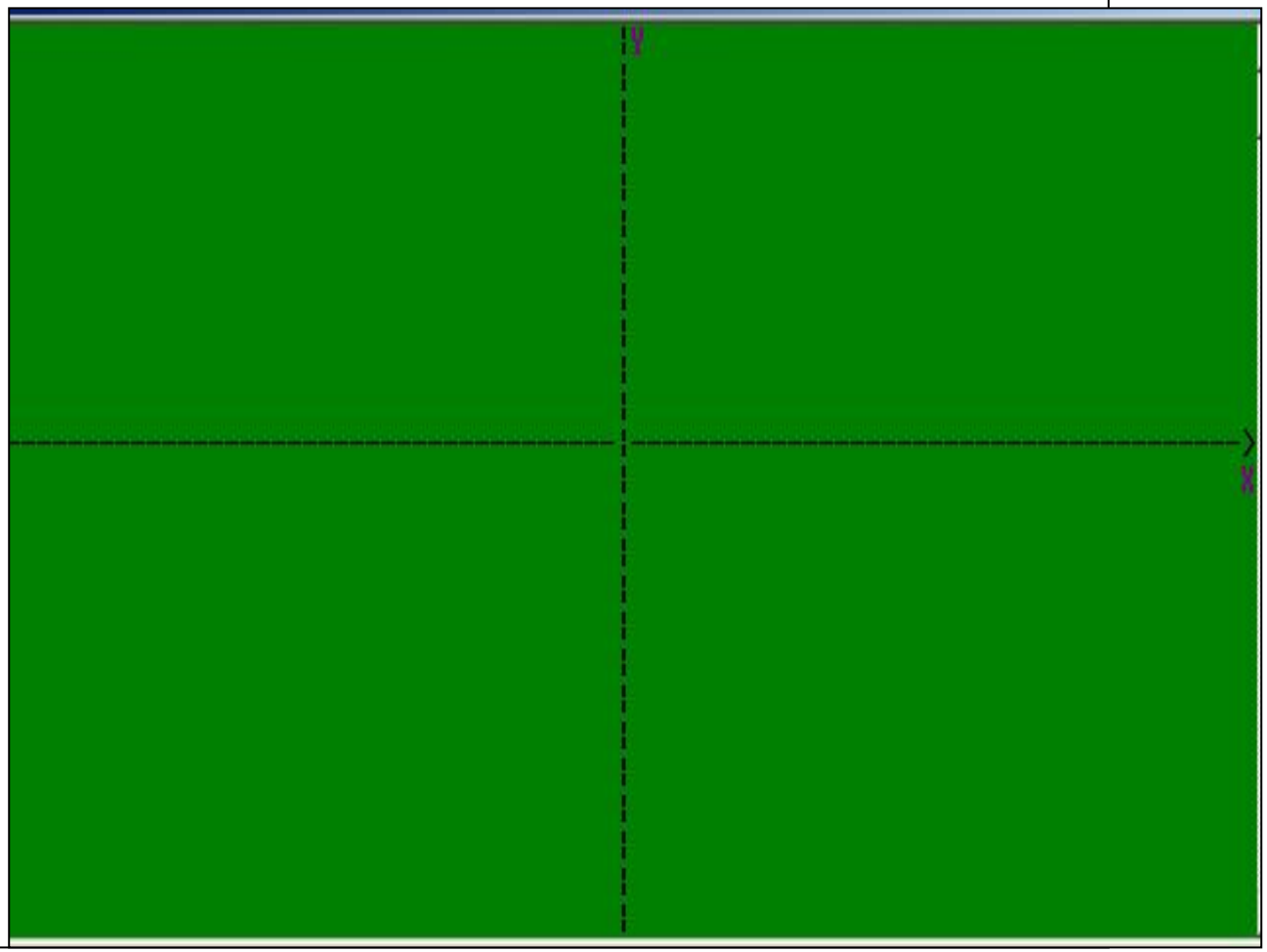
```
uses crt; //Подключение модуля CRT
begin
  textbackground(2); // установка цвета фона(Зеленого)
  clrscr; // очистка экрана и установка курсора на позицию (1;1)
  textcolor(0); // установка цвета текста (Черный)
  gotoxy(22, 12); // установка курсора на позицию 22,11
  write('Программа с использованием CRT!'); // вывод текста на экран
  delay(1500); //задержка на 1500 мсек
  gotoxy(30, 20);
  textcolor(14);
  write('Новое сообщение');
  readln;
  clrscr;
  readln
end.
```



Пример №3 Построение графика (y = -x)

- Построение координатной оси

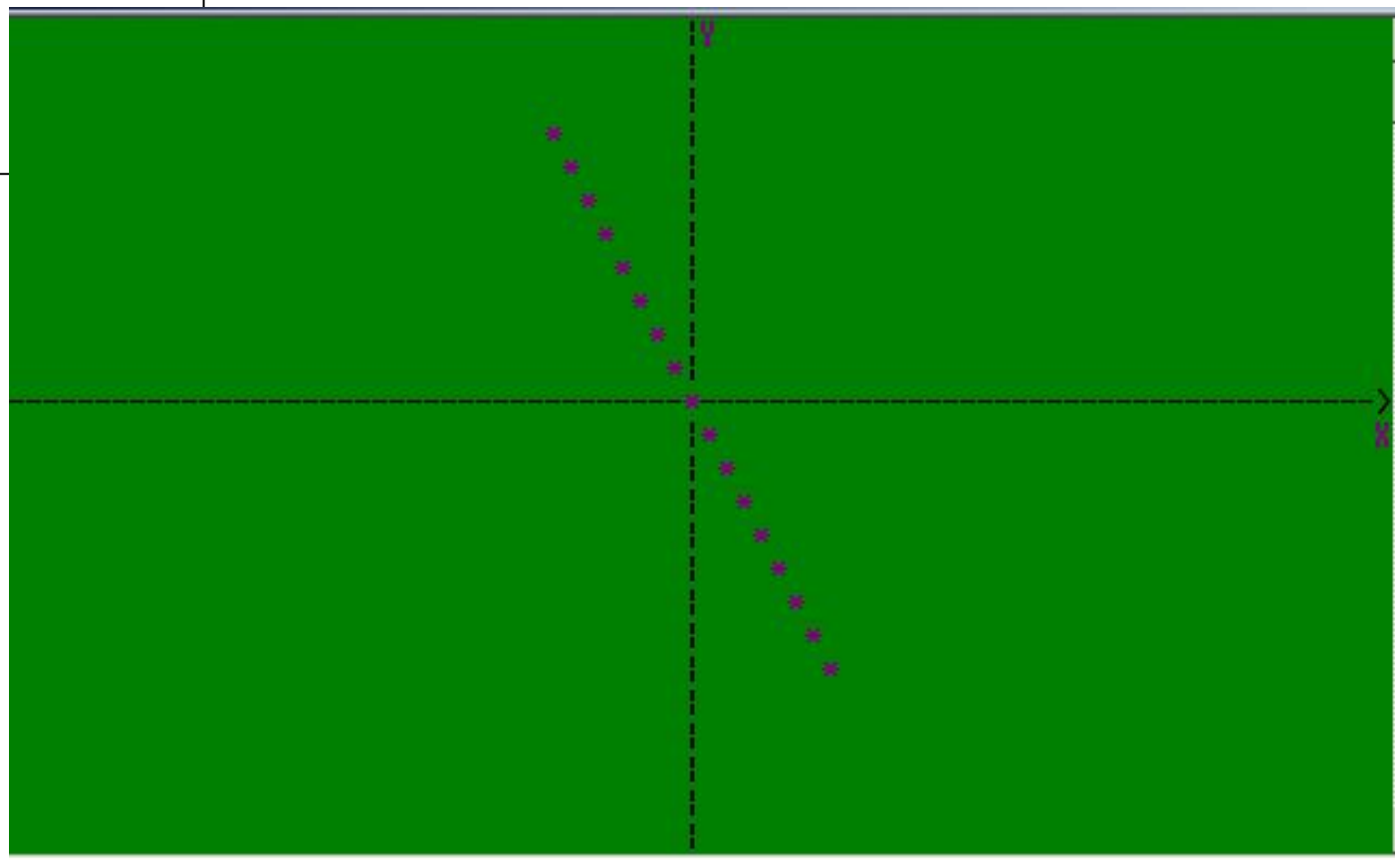
```
for i:=1 to 79 do begin
textcolor(0);
gotoxy(i, 12);
write('-');
delay(15);
end;
textcolor(0);
gotoxy(80, 12);
write('>');
for i:=1 to 25 do begin
textcolor(0);
gotoxy(40, i);
write('|');
delay(15);
end;
textcolor(5);
gotoxy(80, 13);
write('X');
textcolor(5);
gotoxy(41,1);
write('Y');
```



Пример №3 Построение графика (y = -x)

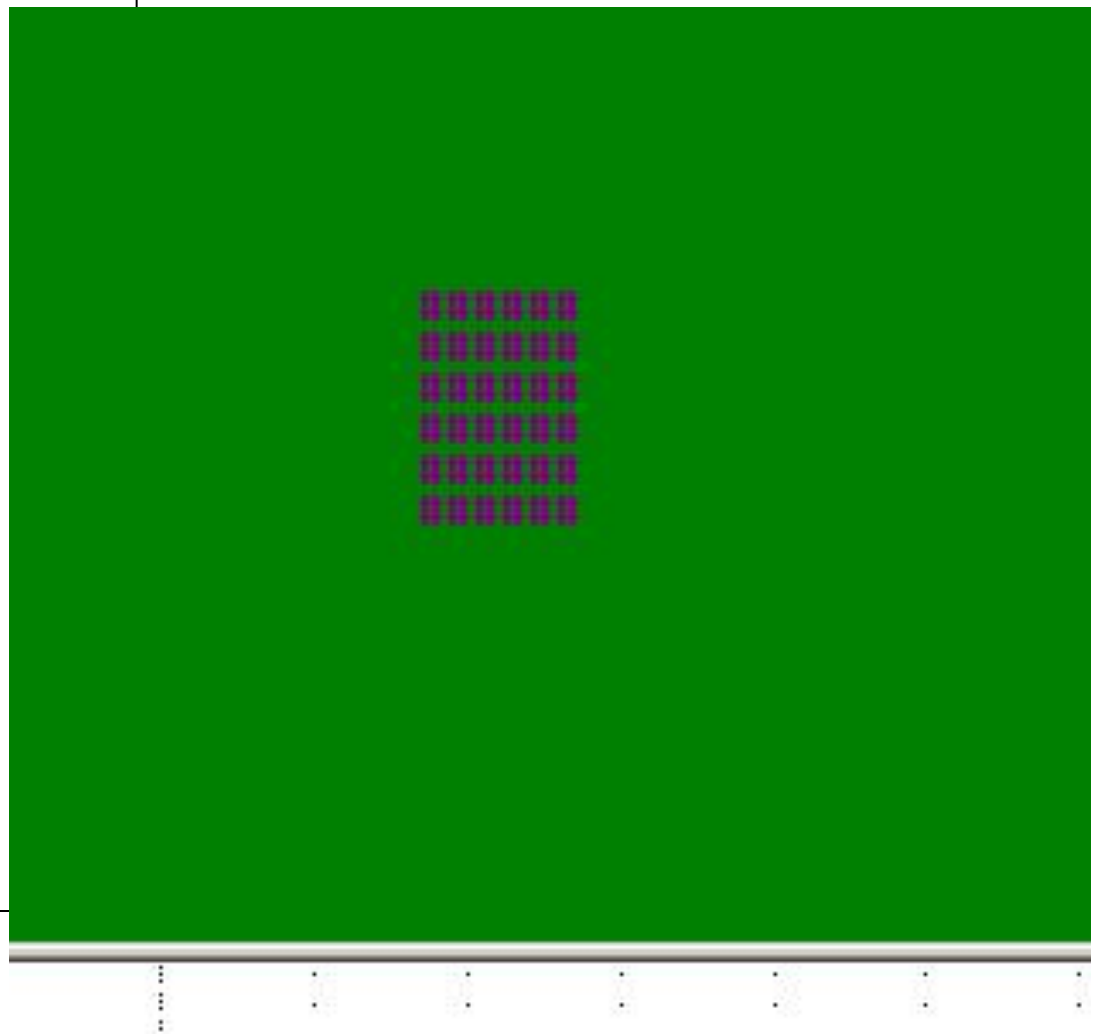
- Построение графика

```
for i:=-8 to 8 do begin
x := i + 40;
y := i + 12  ;
textcolor(5);
gotoxy(x, y);
write('*');
end;
```



Пример №4 Нарисовать квадрат из “#” в цикле

```
uses crt; //Подключение модуля CRT
var j,i,x,y: integer;
begin
  textbackground(2);
  clrscr;
  for i:=10 to 15 do begin
  for j:=38 to 43 do begin
  textcolor(5);
  y:=10;
  gotoxy(j, i);
  write('#');
   delay(100);
  end;
 end;
  readln
end.
```



Прочитать строку символов и опустить символы строки по очереди на строку с указанным номером

Обозначим:

строку символов переменной **S,**

номер строки для размещения результата программы **- L**,

длину строки - **N,**

номер текущего символа в строке **- X,**

номер текущей вертикальной координаты - **Y.**

Алгоритм:

читать номер L;

читать строку символов S;

определить длину строки N;

для x:=1 до N повторять

{y:=1;

повторять

стереть символ в позиции (x,y);

увеличить y координату;

выдать символ в позиции (x,y);

выдержать паузу

до y=L }

Будем использовать стандартные цвета: цвет символов – белый, цвет фона – чёрный.

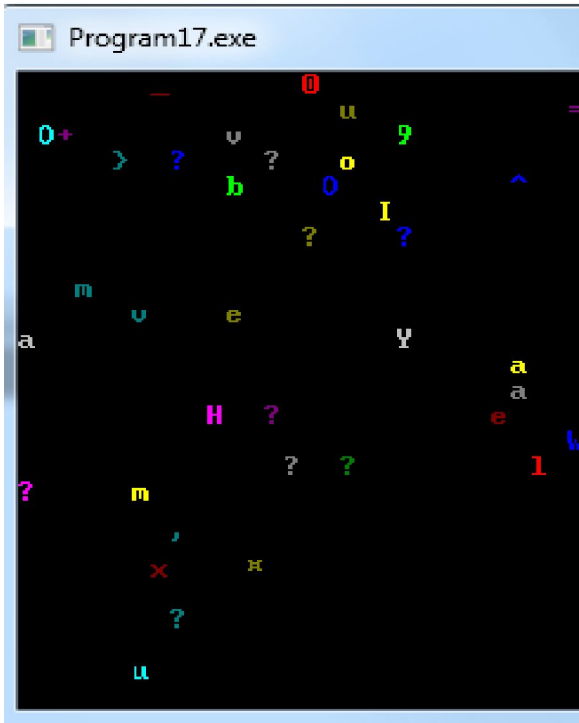
```
Uses CRT;
var  S: string;
  X, Y, N, L: integer;
begin
  Textcolor(7);
  Window(1, 1, 80, 25);
  Clrscr;
  Writeln('задайте номер строки');
  Readln(L);
  Writeln('задайте строку символов');
  Readln(S);
  N := Length(S);
  for X := 1 to n do
  begin
    Y := 1;
    repeat
      Gotoxy(X, Y);
      Textcolor(0);{установить цвет символа,
совпадающий с цветом фона}
      Write(S[x]);{стереть символ}
      Y := Y + 1;
      Gotoxy(X, Y);{спуститься вниз}
      Textcolor(7);{установить стандартный – белый
цвет символов}
      Write(S[X]);{выдать символ}
      Delay(1000);{пауза}
    until Y = L
  end;
  Readln
end.
```

```
program scrsav;
uses Crt;
var n, i, x, y, c, t, z : Word;
    err : Integer;

begin
  n := 10;
 if ParamCount > 0 then
   if paramstr(1) = '?' then
   begin
     WriteLn('scrsav [density : Byte] (=10 by default)');
     Halt(1);
   end
   else
   begin
     Val(ParamStr(1), n, err);
     if (err <> 0) or (n <= 0) then
        n := 10;
   end;
 Randomize;  {активизация генератора случайных чисел}
 while not KeyPressed do
 begin
    y := Random(24) + 1;    {генерация случайного числа от 1 до 25}
    x := Random(79) + 1;    {генерация случайного числа от 1 до 80}
    z := Random(220) + 33; {генерация случайного символа}

    c := Random(14) + 1;    {генерация случайного цвета от 1 до 15}
    GotoXY(x, y);
    TextColor(c);
    Delay(n);
   Write(Chr(z));
   for i := 1 to 10 do
   begin
      y := Random(24) + 1; {генерация случайного числа от 1 до 25}
      x := Random(79) + 1; {генерация случайного числа от 1 до 80}
      GotoXY(x, y);
      TextColor(Black);
      Delay(n);
     Write(' ');
   end;
 end;
end.
```

Написать простейший скрин–сейвер (screen saver) — программу, предохраняющую монитор от пережигания.

Его основными чертами должны стать:

- преобладание чёрного фона;
- регулярная смена позиций цветовых пятен;
- прекращение работы при нажатии произвольной клавиши на клавиатуре.



```
Randomize;  {активизация генератора случайных чисел}
while not KeyPressed do
begin
   y := Random(24) + 1;  {генерация случайного числа от 1 до 25 (координаты дисплея)}
   x := Random(79) + 1;  {генерация случайного числа от 1 до 80 (координаты дисплея)}
   z := Random(220) + 33; {генерация случайного символа (ASCII код символа) }
   c := Random(14) + 1;   {генерация случайного цвета от 1 до 15 (таблица цветов) }
```

GotoXY(x, y); // переместит курсор в заданную позицию в пределах текущего окна //(экрана), где x, y сгенерированы случайным образом

TextColor(c); // установит цвет выводимого текста.

Delay(n); // приостановит исполнение программы на ms миллисекунд

Write(Chr(z)); //выводит символ на экран

TextColor(Black); // установит цвет выводимого текста - черный