

ОСНОВЫ ЯЗЫКА PASCAL

Загрузка изображений

Файлы

Файл – это область на диске, имеющая имя.

Файл

ы

Текстовы

е

только текст без оформления,
не содержат управляющих
символов (с кодами < 32)

ASCII (1 байт на символ)

UNICODE (2 байта на символ)

*.txt, *.log,

*.htm, *.html

Двоичные

могут содержать любые
символы кодовой таблицы

*.doc, *.exe,

*.bmp, *.jpg,

*.wav, *.mp3,

*.avi, *.mpg

Папки (каталоги)

Работа с файлами

Особенности:

- имя файла упоминается только в команде `assign`, обращение к файлу идет через файловую переменную
- файл, который открывается на чтение, должен **существовать** (`FileExists` в библиотеке `SysUtils`)
- если файл, который открывается на запись, существует, старое содержимое **уничтожается**
- данные записываются в файл в текстовом виде

Загрузка изображения из файла

Любые дисковые файлы становятся доступными программе после связывания их с файловой переменной, объявленной в программе. Все операции в программе производятся только с помощью связанной с ним файловой переменной.

```
Assign(f, 'FileName');
```

Процедура `assign` обеспечивает связь файловой переменной программы с реальным файлом на диске.

```
Assign(F,'D:\BP\USER\10A\familia\chisla.pas');
```

Процедуры

Reset (f) – открывает для чтения файл, с которым связана файловая переменная f.

Rewrite (f) – открывает для записи файл, с которым связана файловая переменная f.

Если указанный файл уже существовал, то все данные из него уничтожаются.

Close (f) – закрывает открытый до этого файл с файловой переменной f.

Вызов процедуры **Close** необходим при завершении работы с файлом.

Указатели

- указатель – это переменная, в которой можно хранить адрес другой переменной;
- запись p^{\wedge} обозначает *значение* ячейки, на которую указывает указатель p ;
- при объявлении указателя надо указать тип переменных, на которых он будет указывать, а перед типом поставить знак \wedge ;
- можно объявлять указатель и не связывать его при этом с каким-либо конкретным типом данных (*нетипизированный указатель*). Для этого служит стандартный тип **POINTER**. С их помощью удобно динамически размещать данные, структура и тип которых меняются в ходе работы программы.
- **nil** – это *нулевой указатель*, он никуда не указывает

```
p: pointer;
```



Нельзя использовать указатель, который указывает неизвестно куда (будет сбой или зависание)!

Вывод изображения на экран

`PutImage(x, y, BitMap, BitType)` – помещает битовое изображение на экран.

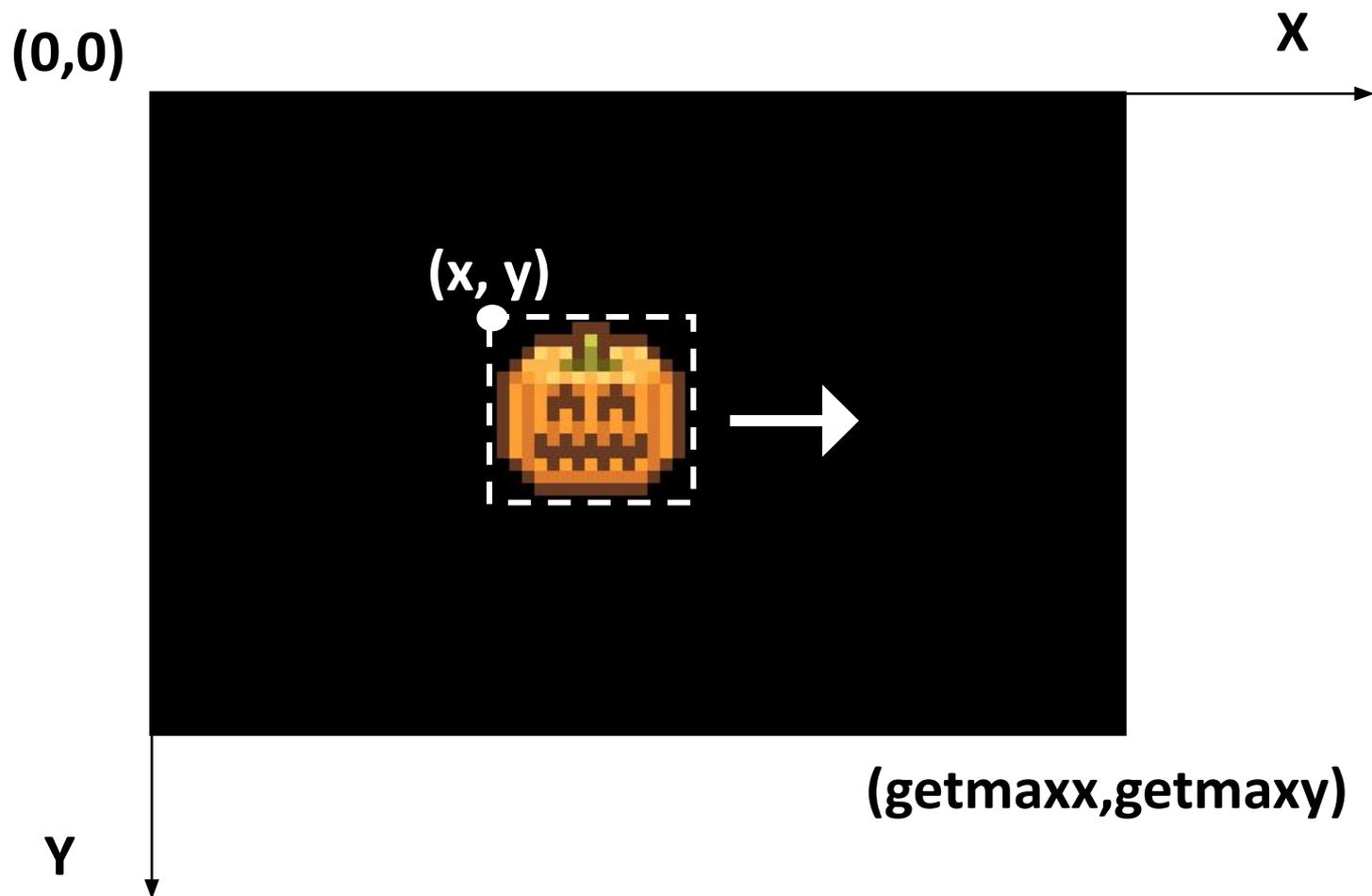
Точка с координатами `(x, y)` — верхний левый угол прямоугольной области на экране.

`BitMap` — нетипизированный параметр, в котором содержится высота, ширина и двоичный образ изображения, которое будет помещено на экран.

`BitType` определяет, какая двоичная операция будет использована при выводе изображения на экран: `NormalPut(0)`, `XORPut(1)` – это часто используемая методика в анимации для перемещения изображения по экрану.

```
Putimage(x, y, p^, 1);
```

Неуправляемое движение



Функция загрузки рисунков

```
function loader(filename: string): pointer;  
var f: file; size: longint; p: pointer;  
begin  
  assign(f, filename);  
  if FileExists(filename) then  
    begin  
      reset(f, 1);  
      size := FileSize(f);  
      GetMem(p, size);  
      BlockRead(f, p^, size);  
      Close(f);  
      loader:=p;  
    end;  
  end;  
end;
```

Выделяем память
размера size

Переписываем информаци
размера size из файла в место,
куда указывает p

Инициализация изображений

Все подгружаемые рисунки должны находиться в папке с FreePascal, либо путь прописывается в настройках:

```
procedure initpict;  
begin  
    p1 := loader( 'apple.bmp' );  
    p2 := loader( 'orange.bmp' );  
end;
```

Неуправляемое движение

```
procedure neupr(var x, y, hx, hy: integer; p:
pointer);
begin
  putimage(x, y, p^, 1);
  if (x < 0) or (x > getmaxx - sh ) then
    hx:=-hx;
  if (y < 0) or (y > getmaxy - vs) then
    hy:=-hy;
  x := x + hx;
  y := y + hy;
  putimage(x, y, p^, 1);
  delay(20);
end;
```

Управляемое движение

```
procedure upr(var x, y: integer; h: integer;
p: pointer);
begin
  putimage(x, y, p^, 1);
  ch := readkey;
  if ch = #0 then
  begin
    ch := readkey;
    case ch of
      left: if x > h then x := x - h;
      right: if x < getmaxx-h-sh then x := x + h;
      up: if y > h then y := y - h;
      down: if y < getmaxy-h-vs then y := y + h;
    end;
  end;
  putimage(x, y, p^, 1);
end;
```

Проверка отбивания

x_1, y_1 – координаты неуправляемого объекта,
 x_2, y_2 – координаты управляемого объекта.

```
function proverka (var x1, y1, x2, y2: integer) :  
boolean;  
Begin  
If ((x1 > x2 - sh1 + 1) and (x1 < x2 + sh2 - 1)) and  
((abs(y2 - y1 - vs1) <= 6) or (abs(y1 - y2 - vs2) <= 6)) then  
    proverka := true  
else  
    proverka := false;  
end;
```

Полная программа

```
Uses wingraph, wincrt;  
Const down = #80, ...  
var x1, y1, x2, y2, gd, gm, hx, hy, h: integer;  
function loader(filename: string): pointer; ...  
procedure InitPict;...  
procedure Neupr(var x, y, hx, hy: integer; p: pointer); ...  
Procedure Upr(var x, y: integer; h: integer; p: pointer); ...  
Procedure InitData; ...
```

Begin

```
  Initgraph(gd, gm, '');  
  InitData;  
  InitPict;  
  putimage(x1, y1, p1^, 1);  
  putimage(x2, y2, p2^, 1);  
  repeat  
    Neupr(x1, y1, hx, hy, p1);  
    if Keypressed then Upr(x2, y2, h, p2);  
  until ch=esc;  
  Closegraph;  
end.
```

начальные
условия

процедуры

первая отрисовка
картинок

Задание

1. Сделать «Отбивалку» с картинками.
2. Сделать игру «Защита города». При попадании в город он разрушается.
3. Сделать игру «Собери фрукты» на основе игры «Собери шарики».
4. Сделать игру «Защита города». Добавить падающие метеориты.

