Алгоритмы и алгоритмические языки

Лекция 16

Язык Паскаль: файловые типы

Язык Паскаль.Типы данных

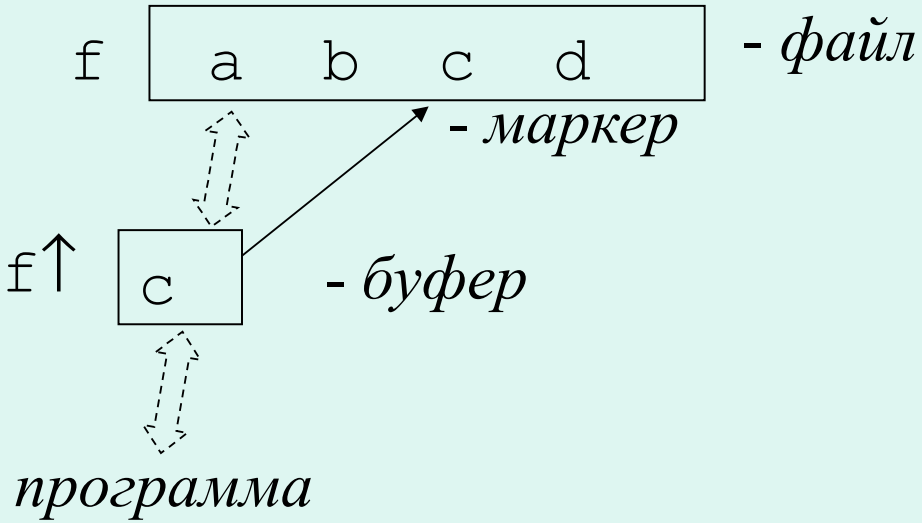
Язык Паскаль. Файловые типы

Объекты файловых типов — файлы.

Файл — последовательность элементов, обладающая следующими свойствами:

- **все элементы одного и того же типа (базового типа файла)**
- **число элементов не фиксировано**

Пустой файл - файл, не содержащий не одного элемента.



Считывать элементы из файла можно только последовательно

Запись элементов происходит в конец файла

Описание файловых типов и переменных

Файловый тип — множество всех файлов, имеющих один и тот же базовый тип

```
<файловый тип> ::= file of <базовый тип>
```

Примеры

```
type T1 = file of char;
          T2 = file of array[1..10] of boolean;
```

Переменные-файлы

```
var t: T1;
    b1, b2: T2;
         z: file of real;
```

Операции над файлами

Присваивание файлов запрещено!
Передача файла в процедуру или функцию - только по ссылке

Режимы работы с файлом

– чтение
– запись

Установив какой-то режим, можно выполнять только чтение (или только запись)
Менять режим работы с файлом можно любое число раз.

Обозначения:
type T = ……..
var f: file of T;
x: T;
{f↑ – буферная переменная}

Работа в режиме чтения

1. Установка режима чтения

```
reset(f)
```

Если файл пуст, после `reset` *значение буферной переменной не определено*

2. Проверка на конец файла

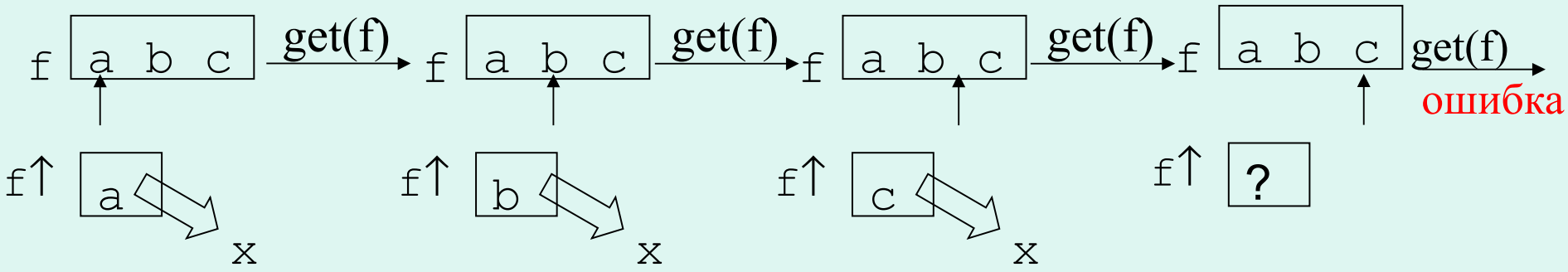
`eof(f)` =
- `true`, если маркер файла указывает на позицию за последним элементом
- `false`, если маркер указывает на какой-то элемент файла

Работа в режиме чтения

3. Считывание элемента из файла

```
get(f)
```

```
x:=f↑ ; get(f); x:=f↑
```



Особые случаи:

1. Если до выполнения get маркер указывал на последний элемент файла, то после get

значение `f↑` *не определено,*

eof(f) = true

2. Если перед выполнением get было eof(f)=true, то get будет ошибкой

Работа в режиме чтения

Пример

Суммирование элементов файла

```
var G: file of real;  S: real;


S:=0;
reset(G);  {на начало G + режим чтения}
while not eof(G) do   {пока не конец G}
  begin
    S:=S+G↑;  {добавить в S очередное число}
    get(G)    {к следующему элементу файла}
  end
```

Работа в режиме чтения

3. Запись элемента файла в переменную

```
read(f,x)
```

```
x:=f↑ ;
get(f);
```

Пример: сумма элементов файла

```
S:=0;
reset(G);
while not eof(G) do
begin
  read(G,x);
  S:=S+x
end;
```

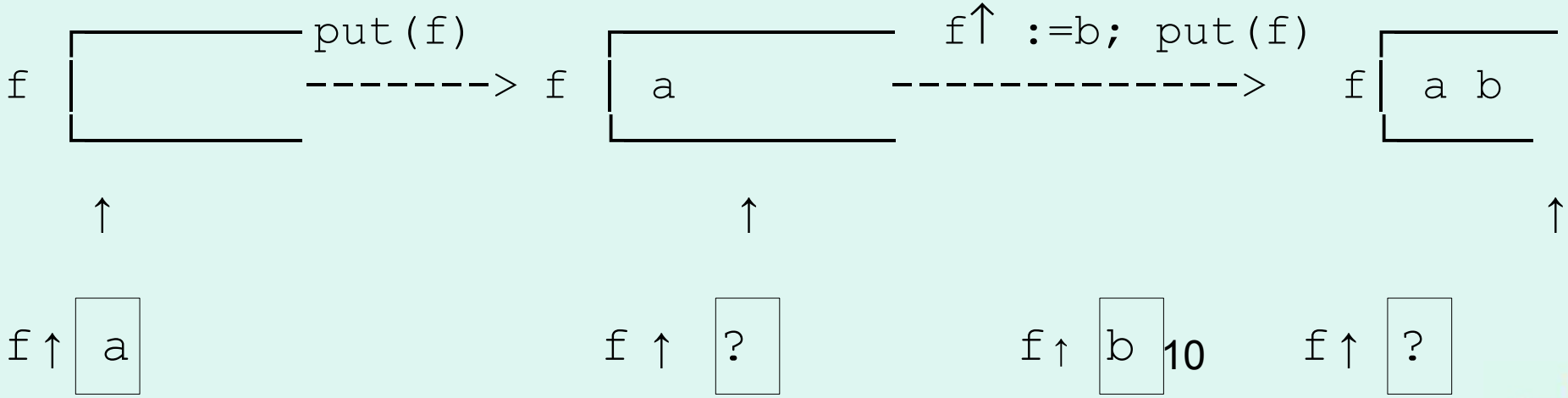
Работа в режиме записи

1. Установка режима записи

```
rewrite(f)
```

В Паскале запись в файл возможна
только с самого начала файла и
только после уничтожения всего предыдущего содержимого файла.

2. Запись элемента в файл `put(f)`



Работа в режиме записи

3. Запись элемента в файл — write

```
write(f,x)
```

```
f↑:=x;
put(f);
```

Пример:

Запись в файл всех цифр

```
var t: file of char;
    c: char;

rewrite(t);      {очистка t + режим записи}
for c:='0' to '9' do
 begin t↑:=c; put(t) end;
       { write(t,c) }
```

Реализация дополнительных операций для работы с файлами

Копирование содержимого одного файла в другой

```
type F = file of T;
           {в качестве T  - любой допустимый тип}

procedure copy(var source, dest:F); {source → dest}
 var x: T;
begin
   reset(source); rewrite(dest);
   while not eof(f) do
   begin
     read(f,x);
     write(g,x);
   end
end;
```

Текстовые файлы

`text` — стандартный тип текстовых файлов

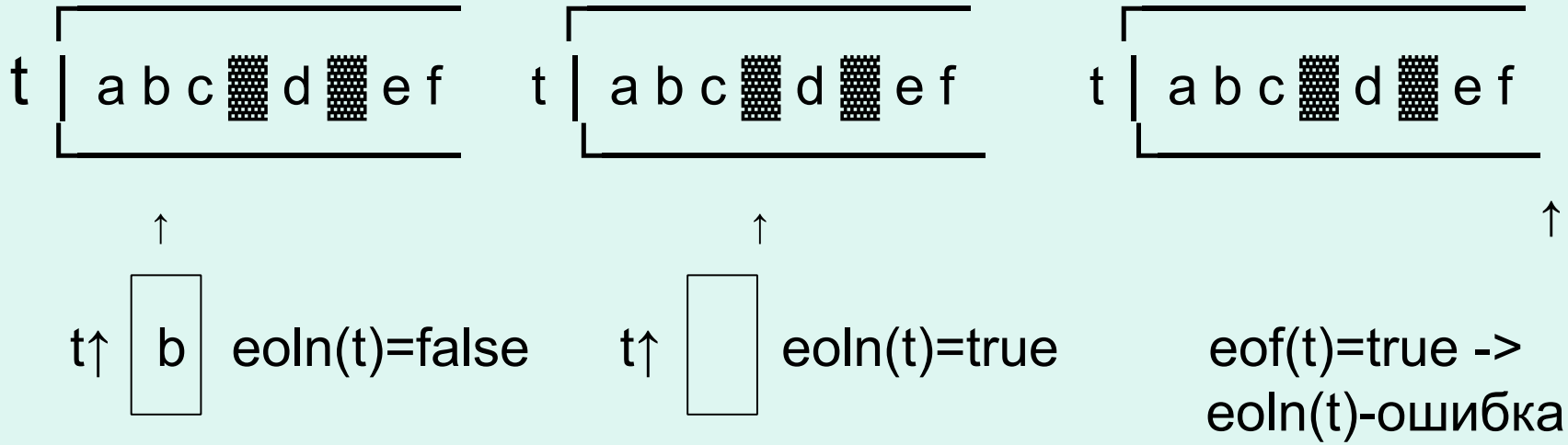
```
var t : text;
```

`text` и `file of char` — разные типы

Операции над строками текстового файла

1. Логическая функция eoln(t)

eoln(t) = { `true`, если маркер файла t на конце строки
          { `false`, если маркер файла t на обычном символе



t↑ | b | eoln(t)=false

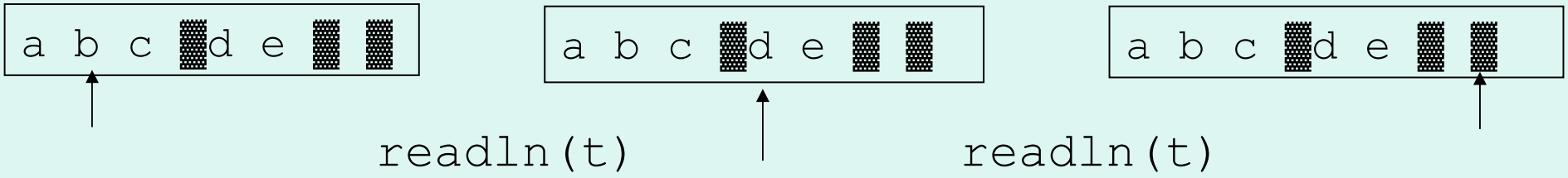
t↑ | | eoln(t)=true

eof(t)=true ->
eoln(t)-ошибка

При eoln(t) = true `t↑=' '`

Операции над строками текстового файла

2. Переход на начало следующей строки - процедура readln(t)




```
while not eoln(t) do
    get(t);
  get(t);
```

3. Запись конца строки в файл - процедура writeln(t)

Работа с текстовым файлом. Пример

Подсчет количества строк, начинающихся с #, в текстовом файле t

```
k:=0;
reset(t);
while not eof(t) do   {цикл по строкам}
begin
  if not eoln(t) then   {непустая строка}
    begin
      read(c);
      if c= '#' then k:=k+1
    end;
  readln(t)  {пропускаем оставшиеся символы строки}
end
```

Файлы input и output

input — стандартный файл ввода
output — стандартный файл вывода

```
{
 var input, output : text;

   reset(input);
   rewrite(output);
}
```

Сокращенная запись:

```
  read(x) = read(input, x)
  readln  = readln(input)
  write(x)= write(output,x)
  writeln = writeln(output)
```

Расширенные возможности чтения и записи для текстовых файлов

```
 var t : text;
```

Ввод

```
   read(t,x)    x- char, integer или real

  read(t,x1,...,xn)    = read(t,x1);...;read(t,xn)
  readln(t,x1,...,xn) = read(t,x1,...,xn);readln(t)
```

Вывод

```
   write(t,p)
p -  это e, e:m или e:m:n
e – выражение типа char, boolean, integer или real,
    либо строка
m – ширина поля вывода
n – количество цифр в дробной части результата e
write(t,p1,...,pn)    = write(t,p1);...;write(t,pn)
writeln(t,p1,...,pn) = write(t,p1,...,pn);writeln(t)
```

Внутренние и внешние файлы

Внутренние файлы существуют только во время выполнения программы: появляются в начале ее работы и уничтожаются при выходе их нее

Внешние файлы существуют до/после работы программы.
Пример: `input, output`

Для использования внешних файлов необходимо указать их в заголовке программы:

```
program p(input, output, A,B)
```

Файлы, описанные в программе, но не указанные в заголовке, являются внутренними

Пример работы с текстовыми файлами

A,B — внешние текстовые файлы

Переписать содержимое файла A в файл B с сохранением деления на строки

```
program copy(A,B);
var A, B: text;
      c: char;
begin
  reset(A); rewrite(B);
  while not eof(A) do
   if eoln(A) then begin readln(A); writeln(B) end
              else begin read(A,c); write(B,c) end
end.
```