

Структурированные операторы Паскаля

Структурированными являются такие операторы, которые состоят из других операторов.



К ним относятся:

-составной оператор;

-условный оператор **IF**;

-условный оператор **CASE**;

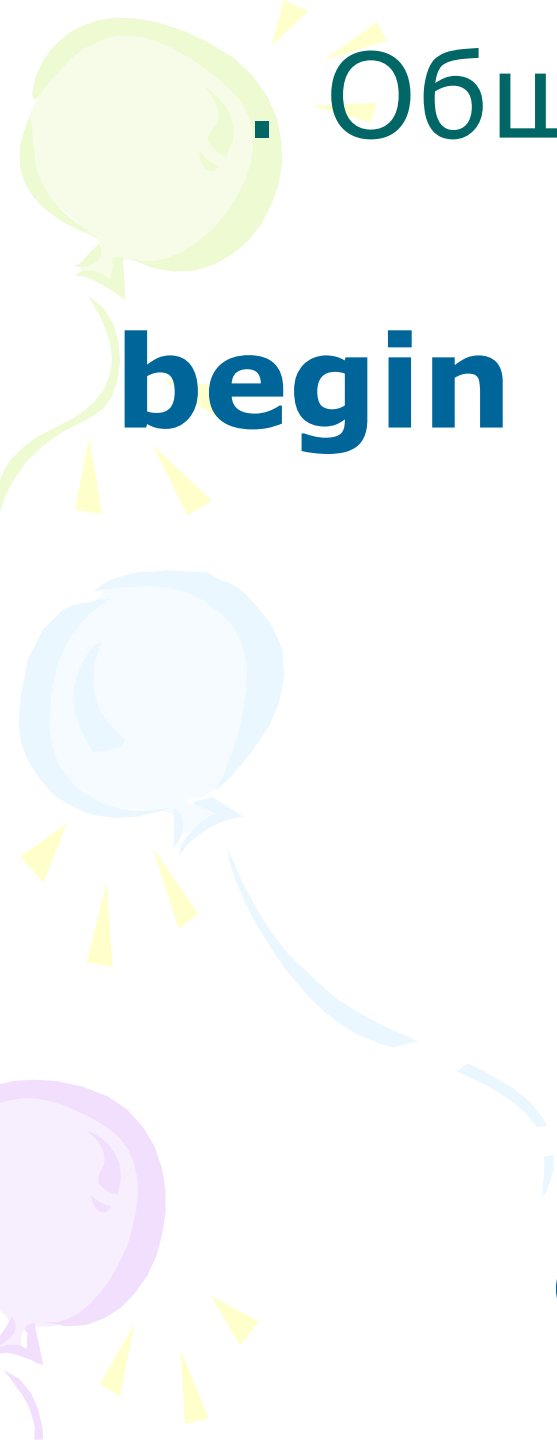
-операторы цикла **WHILE**,

REPEAT, FOR.

Составной оператор

Составной оператор
позволяет объединить
несколько операторов

Паскаля в одну
конструкцию, которая
рассматривается как
составной оператор.



. Общий вид оператора
следующий:

begin

оператор 1;

оператор 2;

.

оператор n

end;

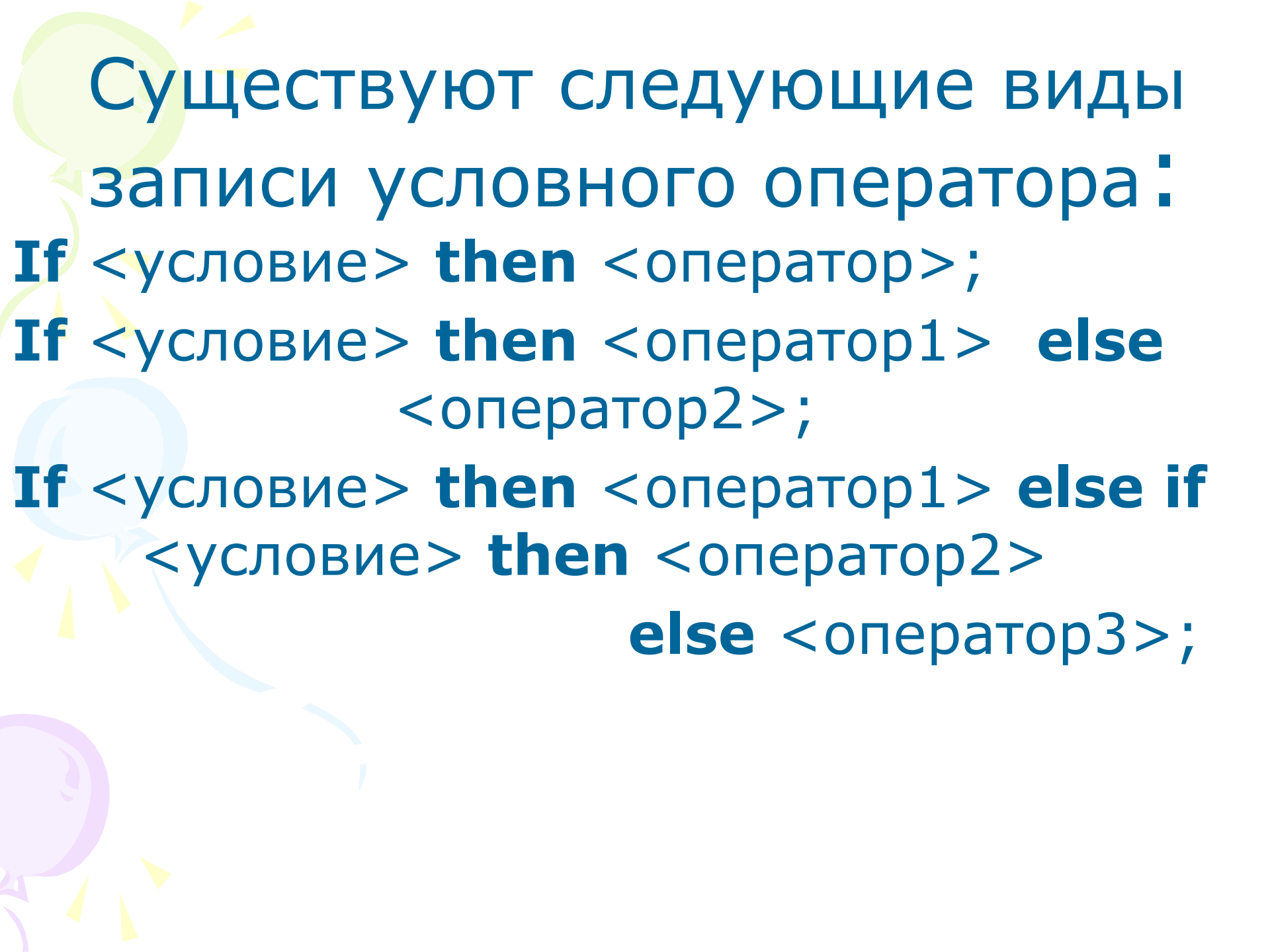
В этой конструкции слова **begin** и **end** выполняют роль операторных скобок.

Составной оператор можно включать в любое место программы, где допускается использование только одного оператора.



Условный оператор

Условный оператор
позволяет на
определенном этапе
выбрать одно из двух
действий в результате
анализа некоторых
условий.

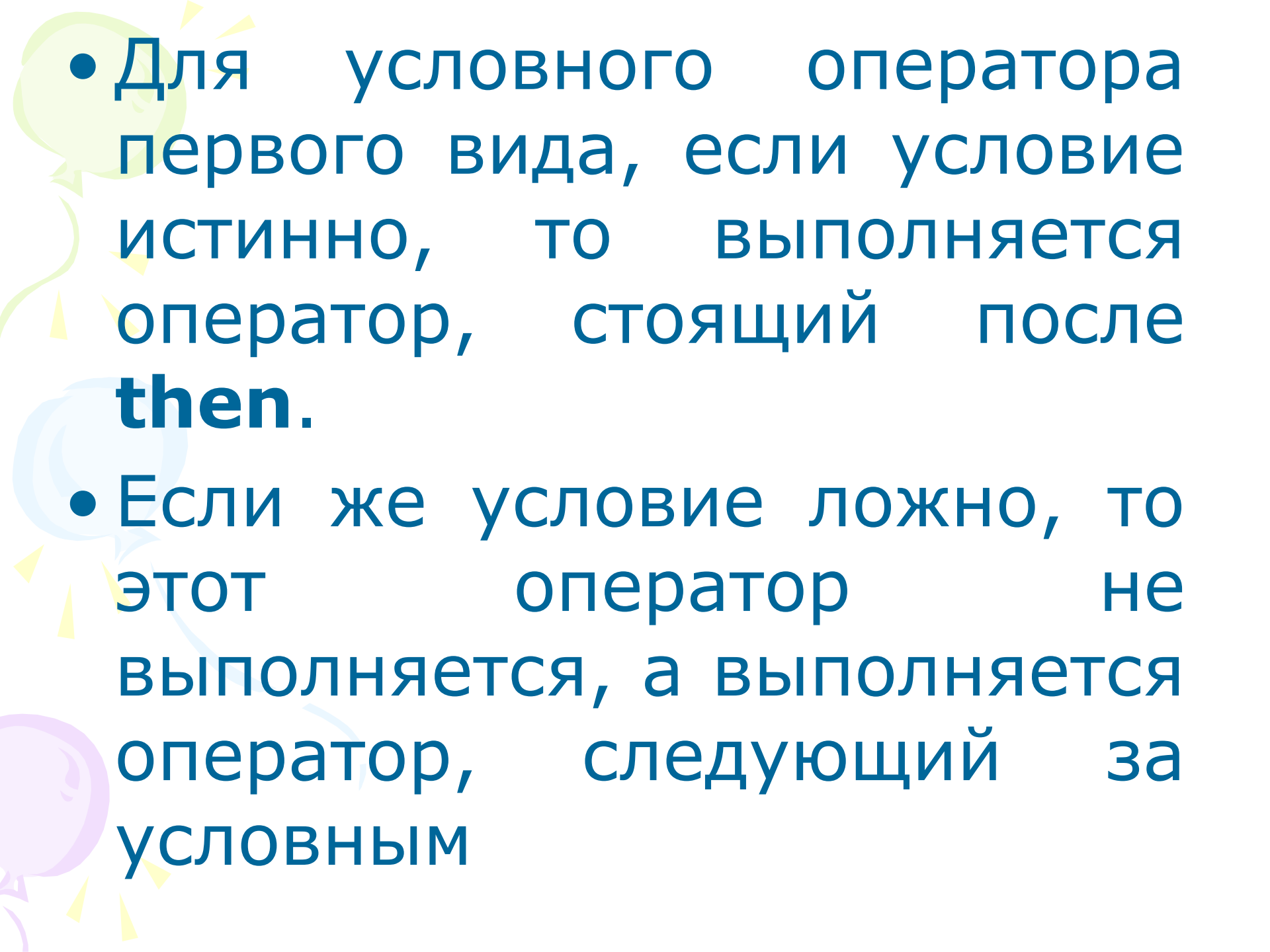


Существуют следующие виды записи условного оператора:

If <условие> **then** <оператор>;

If <условие> **then** <оператор1> **else**
 <оператор2>;

If <условие> **then** <оператор1> **else if**
 <условие> **then** <оператор2>
 else <оператор3>;



- Для условного оператора первого вида, если условие истинно, то выполняется оператор, стоящий после **then**.

- Если же условие ложно, то этот оператор не выполняется, а выполняется оператор, следующий за условным

Например:

if $x < 0$ then $y = x + x$.

Второй вид записи оператора позволяет производить выполнение оператора 1, если условие истинно.

Если условие ложно, то выполняется оператор 2.

Например:

if $x > 0$ then $y := \text{sqrt}(x)$ else $y := x$.

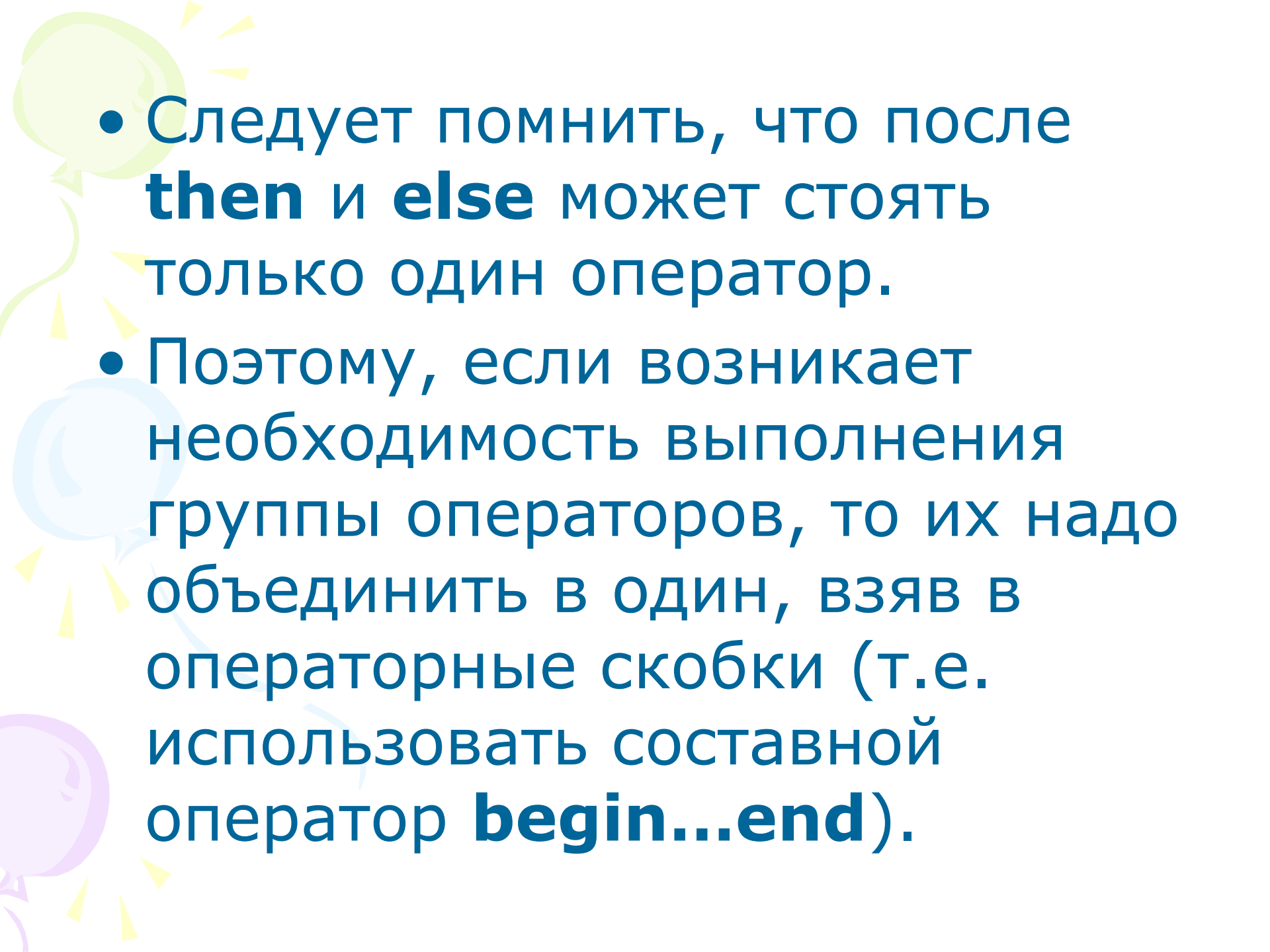
В третьей форме записи условный оператор расширен за счет вложенности новых условий.

Например:

if $x < a$ **then** $p := \ln(x)$

else if $x > b$ **then** $p := \sin$

else $p := \cos(x)$.

- 
- Следует помнить, что после **then** и **else** может стоять только один оператор.
 - Поэтому, если возникает необходимость выполнения группы операторов, то их надо объединить в один, взяв в операторные скобки (т.е. использовать составной оператор **begin...end**).

- Кроме того, при необходимости учета нескольких условий используются логические операции: and (и), or (или), not (не) .
- **Например**, алгоритм: если $A < D$ и $A > C$ то $Y1 := A^2$ и $Y2 := A * C$; следует записать:
- **If (A < D) and (A > C) then begin
Y1 := sqr(A); Y2 := A * C end; .**

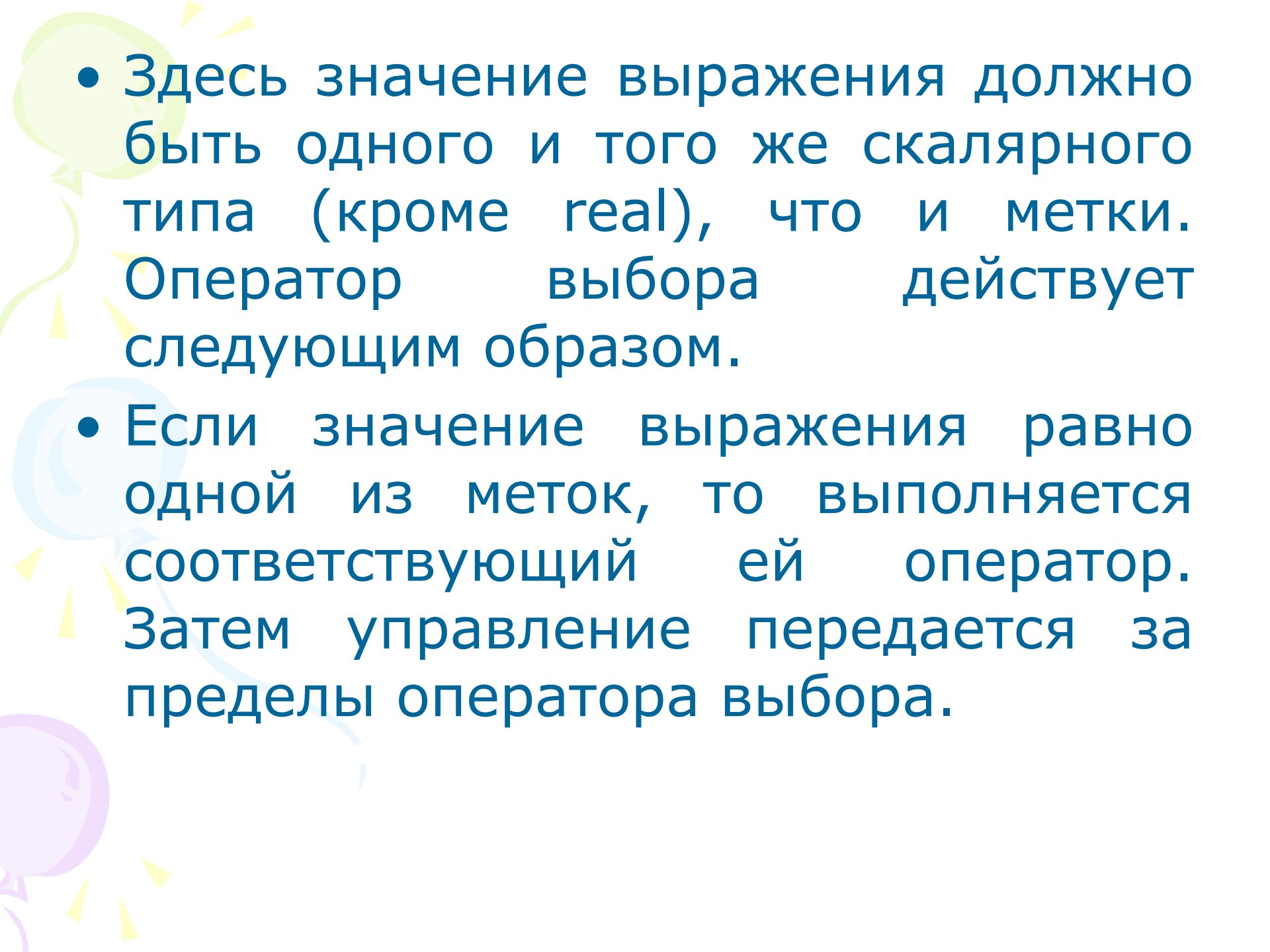
Оператор выбора CASE

- Оператор CASE предназначен для программирования алгоритмов с большим числом разветвлений.
- Этот оператор обеспечивает выполнение одного оператора (простого или составного) из нескольких возможных.



Общий вид оператора CASE:

```
case <выражение-селектор> of  
  <список меток 1>: оператор 1;  
  <список меток 2>: оператор 2;  
  . . . . .  
  <список меток n>: оператор n;  
else <оператор>  
end;
```

- 
- Здесь значение выражения должно быть одного и того же скалярного типа (кроме `real`), что и метки. Оператор выбора действует следующим образом.
 - Если значение выражения равно одной из меток, то выполняется соответствующий ей оператор. Затем управление передается за пределы оператора выбора.

Замечание

- Метки оператора CASE не описываются в разделе **label**, и на них нельзя переходить оператором GOTO.
- Метки внутри одного оператора выбора должны быть различными.

Операторы цикла

Для организации циклов (повторов) при записи алгоритмов на языке Паскаль используются три вида операторов цикла:

- **WHILE** – оператор цикла с предварительным условием;
- **REPEAT** – оператор цикла с последующим условием;
- **FOR** – оператор цикла с управляющим параметром.

Оператор цикла WHILE

Общий вид оператора следующий:

```
while <условие> do <оператор>;
```

где

<условие> – логическое выражение;
<оператор> – тело цикла (простой или составной оператор).

Оператор действует следующим образом.

- Проверяется условие, если оно истинно, выполняются операторы циклической части.
- Как только оно становится ложным, происходит выход из цикла.
- На литературном языке это будет выглядеть примерно так:
«Пока указанное условие верно, выполнять следующее».

Пример: Зависимость удельной теплоемкости химического соединения от температуры выражается формулой

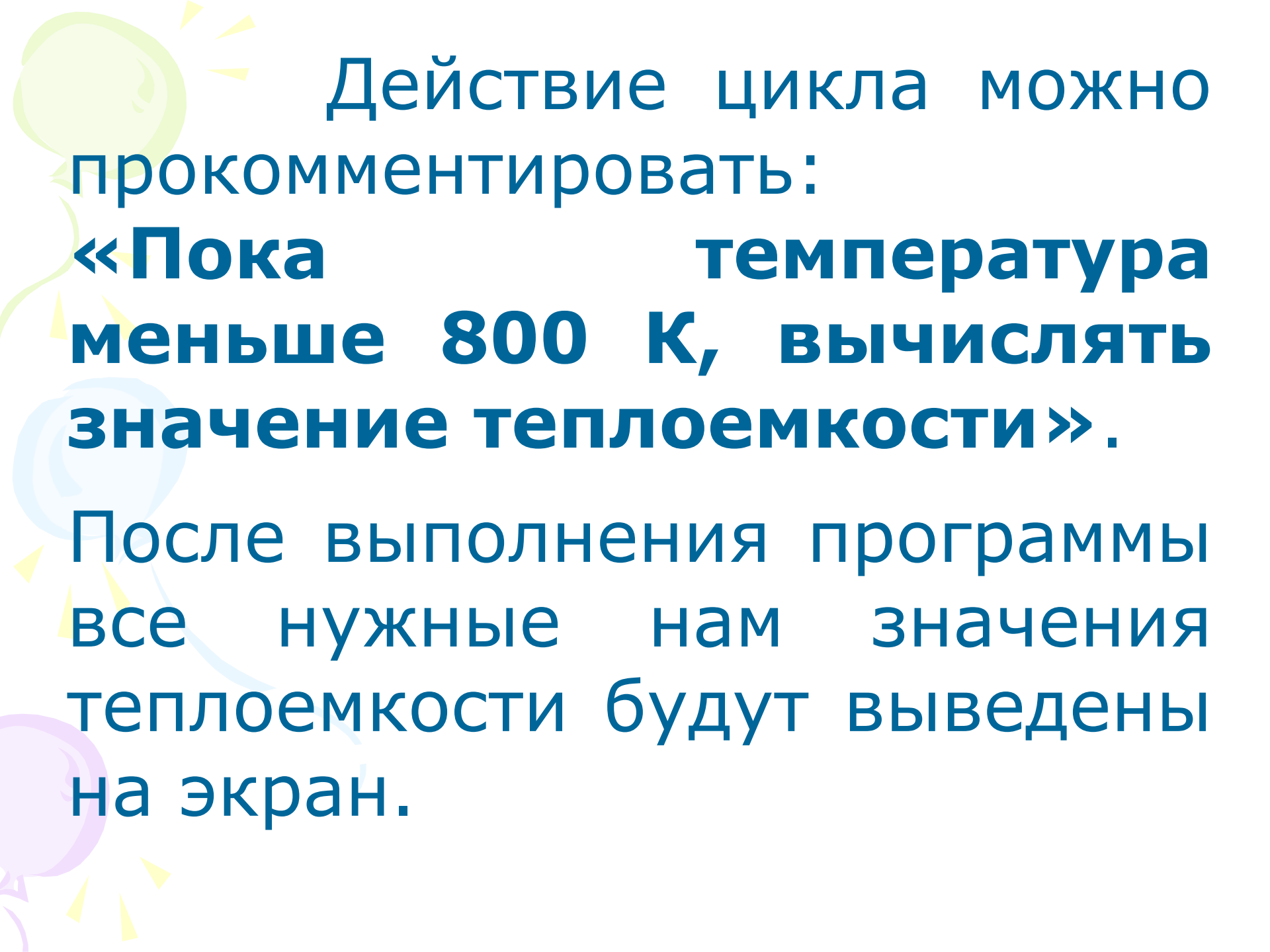
$$C_p = a + bT + cT^2 ,$$

где ***a***, ***b***, ***c*** – постоянные коэффициенты;
T – температура.

Вычислить удельную теплоемкость в интервале температур от 200 до 800 К с шагом 50 К.

Программа

```
Program Tep1;  
var a,b,c,Cp:real;  
    T,h:integer;  
Begin  
  writeln('Введите коэффициенты');  
  readln(a,b,c);  
  T:=200; h:=50;  
  while T<=800 do  
    begin  
      Cp:=a+b*T+c*T*T;  
      T:=T+h;  
      writeln('T=',T:7:3,' Cp=',Cp:10);  
    end;  
End.
```



Действие цикла можно
прокомментировать:

**«Пока температура
меньше 800 К, вычислять
значение теплоемкости».**

После выполнения программы
все нужные нам значения
теплоемкости будут выведены
на экран.



Оператор цикла REPEAT

Общий вид оператора
следующий:

repeat

<оператор 1>;

. {операторы циклической части}

<оператор n>

until <условие>;



Оператор действует следующим образом.

- Выполняются операторы циклической части, проверяется условие.
- Если оно ложно, то вновь выполняется тело цикла, если оно истинно, то происходит выход из цикла.

Это может быть выражено так:

«Повторять действие до тех пор, пока не выполнится условие».

Примечание.

Так как границы цикла обозначены словами **REPEAT** и **UNTIL**, нет необходимости заключать операторы циклической части в операторные скобки **begin – end**, хотя их использование не является ошибкой.

Пример. Вычислить значение теплоемкости C_p с использованием оператора **REPEAT**.

Program Topl;

var a,b,c,Cp:real;

T,h:integer;

Begin

writeln('Введите a,b,c=');

readln(a,b,c);

T:=200; h:=50;

repeat

Cp:=a+b*T+c*T*T;

writeln('T=',T:3,' Cp=',Cp:7:2);

T:=T+h;

until T>800;

End.

Примечание.

Действие оператора **REPEAT**, противоположно действию оператора **WHILE**, т.к. в первом условии выхода из цикла должно быть истинным, а во втором – ложным.

Значения переменных, входящих в условие операторов **WHILE** и **REPEAT** должны обязательно изменяться в теле цикла, иначе цикл не будет завершен. (В приведенном нами примере - это значение переменной **T**).

Оператор цикла FOR.

Оператор цикла FOR используется для организации цикла, когда известно число повторений. **Существует два варианта оператора:**

– при увеличении значения параметра (цикл с положительным шагом: +1)

for i:=n1 **to** n2 **do** <оператор>;

– при уменьшении значения параметра (цикл с отрицательным шагом: -1)

for i:=n1 **downto** n2 **do** <оператор> ,

где **i** – параметр цикла; **n1** и **n2** – начальные и конечные значения параметра цикла; **<оператор>** – тело цикла (простой или составной операторы).

Параметры i , $n1$, $n2$ должны иметь один и тот же тип, кроме `real`, шаг параметра цикла всегда 1.

Цикл действует таким образом.

Параметру i присваивается начальное значение $n1$ и сравнивается со значением $n2$.

До тех пор, пока параметр i меньше или равен конечному значению $n2$ (в первом варианте) или больше, или равен $n2$ (во втором варианте), выполняются операторы циклической части; в противном случае происходит выход из цикла.

Примечание:

1. Внутри цикла нельзя изменять начальное (**n1**) и конечное (**n2**) значения параметра цикла, а также само значение *i*.
2. После завершения цикла значение параметра *i* становится неопределенным (т.е. ничему неравным), за исключением выхода из цикла при помощи **GOTO**.
3. Во всех трех операторах цикла внутри цикла можно использовать операторы **IF**, **GOTO**. Разрешается в любой момент выходить из цикла, не дожидаясь его завершения. Но запрещено при помощи этих операторов передавать управление извне цикла внутрь цикла.

Пример: Рассмотрим расчет теплоемкости с использованием оператора FOR.

```
Program Tep1;  
var a,b,c,Cp:real;  
    T,h,i:integer;
```

```
Begin
```

```
writeln('Введи a,b,c=');
```

```
readln(a,b,c);
```

```
T:=200; h:=50;
```

```
for i:=1 to 13 do
```

```
  begin
```

```
    Cp:=a+b*T+c*T*T;
```

```
    writeln('T=',T:3,' Cp=',Cp:7:2);
```

```
    T:=T+h;
```

```
  end;
```

```
End.
```

Параметр цикла **i** изменяется от 1 до 13, т.к. на заданном интервале температуры от 200 до 800 К с шагом 50 К должно быть вычислено тринадцать значений теплоемкости.

Следует помнить. Операторы цикла **WHILE** и **FOR** могут содержать в теле цикла только один оператор. Поэтому при необходимости вычисления нескольких операторов необходимо заключать их в операторные скобки (т.е. использовать составной оператор **begin ...end**).



Структурированные типы данных

Массивы

Массив – это упорядоченная последовательность элементов одного типа, обозначенных одним именем.

Отдельная величина последовательности называется элементом массива (переменная с индексом). Индекс указывает положение (адрес) элемента в массиве.

Любой массив имеет имя,
размерность и длину (размер).
Количество индексов у
переменной с индексом
определяет размерность
массива. Длина массива – это
общее число его элементов.

Примерами массивов могут быть:

1) вектор $\mathbf{x} = \{x_1, x_2, \dots, x_{10}\}$ – это одномерный массив состоящий из десяти элементов x_i , где $i=1, \dots, 10$

2) матрица

$$\begin{matrix} a_{11} & a_{12} & a_{13} \\ & a_{21} & a_{22} & a_{23} \end{matrix}$$

это двумерный массив из шести элементов a_{ij} , где $i=1,2; j=1,2,3$.

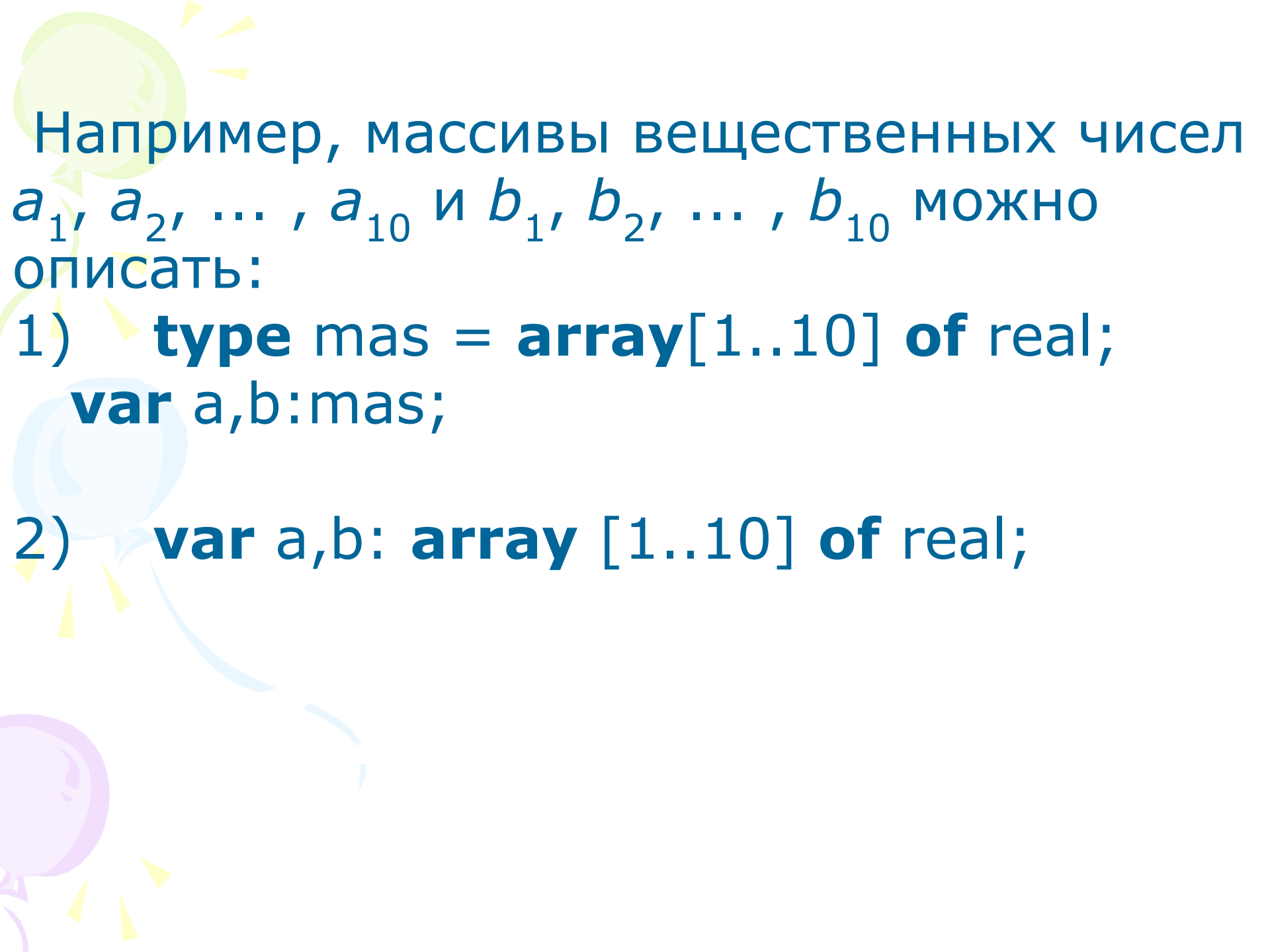
Описание массивов.

Возможны два способа описания массивов:

1) **type** <имя типа> = **array**[<тип индексов>] **of** <тип компонент>;
var <имя массива> : <имя типа> .

Вначале определяется некоторый тип со структурой массива, а затем описывается переменная, как имеющая данный тип.

2) **var** <имя массива> : **array** [**<тип индексов>**] **of** <тип компонент>.



Например, массивы вещественных чисел a_1, a_2, \dots, a_{10} и b_1, b_2, \dots, b_{10} можно описать:

1) **type** mas = **array**[1..10] **of** real;
var a,b:mas;

2) **var** a,b: **array** [1..10] **of** real;

Доступ к каждому элементу массива можно выполнить путем указания имени массива, за которым в квадратных скобках следует индекс элемента.

Индекс элемента может быть задан переменной – **$a[i]$** ; числом – **$a[5]$** ; выражением – **$a[2*i-1]$** .

Примеры описания массивов:

var

{одномерный массив целых чисел}

x:array[1..10]of integer;

{одномерный массив вещественных чисел}

y:array[1..5]of real;

{двумерный массив вещественных чисел}

a:array[1..3,1..5]of real;

{трехмерный массив символьных данных}

b:array[1..2,1..5,1..3]of char;.

Ввод–вывод массивов.

Для ввода–вывода массивов используются циклы. Рассмотрим ввод и вывод массивов на примерах.

Пример ввода–вывода **одномерного** массива.

Пусть для решения задачи необходимо ввести численные значения молекулярных масс десяти химических веществ: M_1, M_2, \dots, M_{10} .



```
var M:array[1..10]of real;  
    i:integer;
```

```
Begin
```

```
{ввод значений массива M в столбце}
```

```
writeln('Введите M');
```

```
for i:=1 to 10 do
```

```
    begin
```

```
        writeln('M', i);
```

```
        readln(M[i]);
```

```
    end;
```

```
{вывод элементов массива M в строку}
```

```
for i:=1 to 10 do
```

```
    write(M[i]);
```

```
End.
```

Ввод значений с экрана монитора будет происходить следующим образом. После появления записи: «введи М», следует записать численное значение М [1] и нажать на «**Enter**» и так до десятого элемента включительно.

Введи М

М1 <значение М[1]> ;

М2 <значение М[2]> ;

.

М10 <значение М[10]> .

Пример ввода-вывода **двумерного**

массива.

Требуется ввести значения

теплоемкостей пяти органических

соединений, представляющих три

гомологических ряда: алканы, алкены,

спирты.

$Cp_{11'}$, $Cp_{12'}$, ... , Cp_{15}

$Cp_{21'}$, $Cp_{22'}$, ... , Cp_{25}

$Cp_{31'}$, $Cp_{32'}$, ... , Cp_{35}

```
var Cp:array[1..3,1..5] of real;
```

```
  i,j:integer;
```

```
Begin
```

```
{ВВОД значений массива}
```

```
  for i:=1 to 3 do
```

```
    for j:=1 to 5 do
```

```
      begin
```

```
        writeln('Введите Cp',i,j);
```

```
        readln(Cp[i,j]);
```

```
      end;
```

```
{ВЫВОД элементов массива по строкам}
```

```
  for i:=1 to 3 do
```

```
    begin
```

```
      for j:=1 to 5 do
```

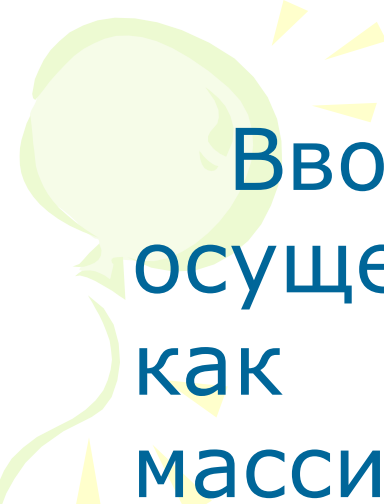
```
        write(Cp[i,j]:7:3,' ');
```

```
      writeln;
```


```
    end;
```

```
  . . . . .
```

```
End.
```



Ввод массива будет осуществляться таким же образом, как и в случае одномерного массива.



При выводе массива на экран появятся три строки по пять численных значений теплоемкостей:

$C_p[1,1]$ $C_p[1,2]$... $C_p[1,5]$

.....

$C_p[3,1]$ $C_p[3,2]$... $C_p[3,5]$



Численные значения элементов массива могут быть также заданы:

- **в разделе *const***

```
Type mas = array[1..3] of real;
```

```
mas1 = array[1..2,1..3] of integer;
```

```
Const M:mas = (12.3,14.6,18.4);
```

{одномерный массив}

```
Sp:mas1 = ((10,16,8), (6,20,12));
```

{двумерный массив}

или

```
Const M: array[1..3] of
```

```
real=(12.3,14.6,18.4); {одномерный массив}
```

```
Sp: array[1..2,1..3] of integer =
```

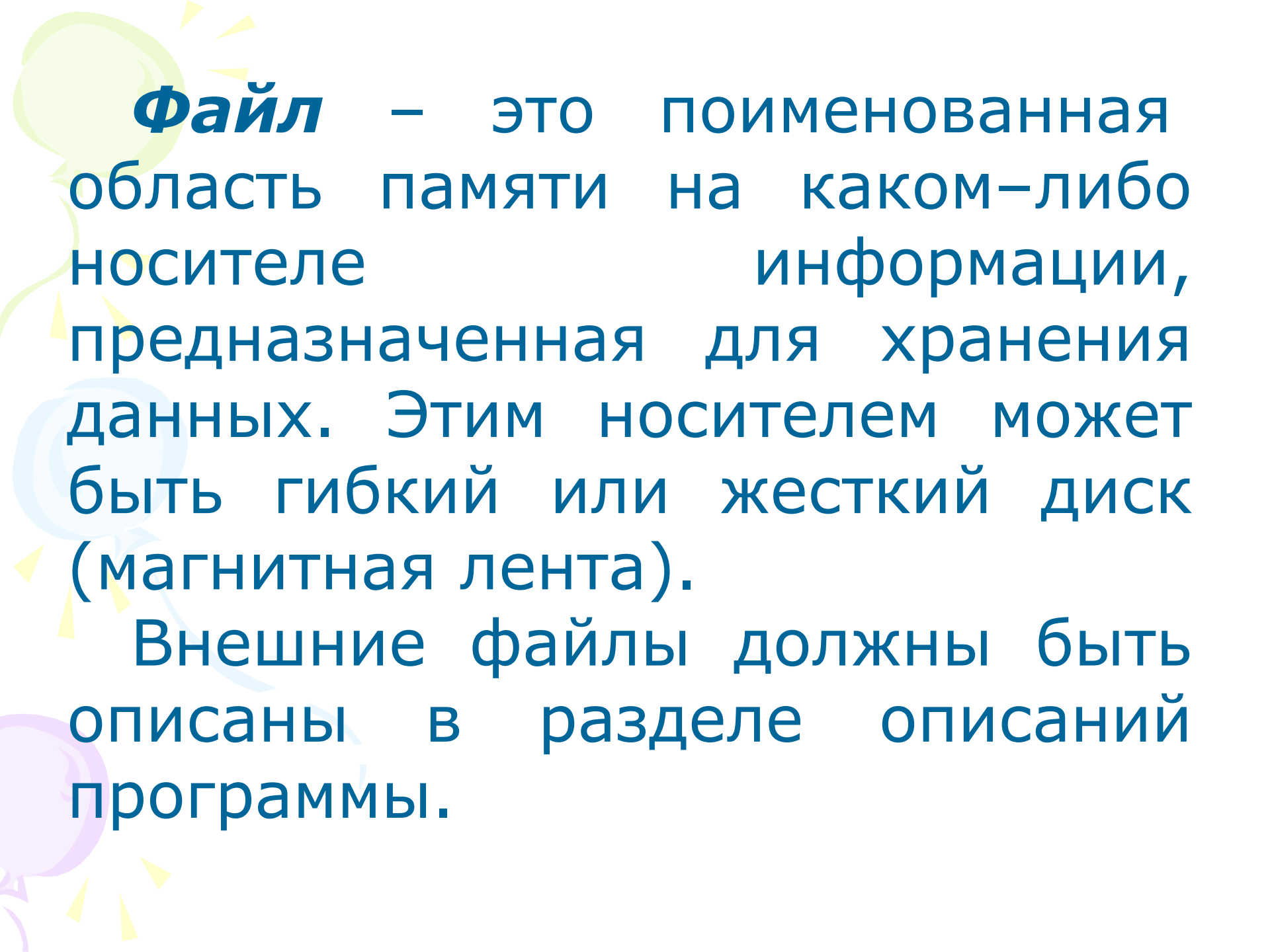
```
((10,16,8), (6,20,12)); {двумерный массив}
```

Файлы

Удобным способом хранения информации служит запись этой информации на магнитный носитель (жесткие, гибкие диски, магнитные ленты).

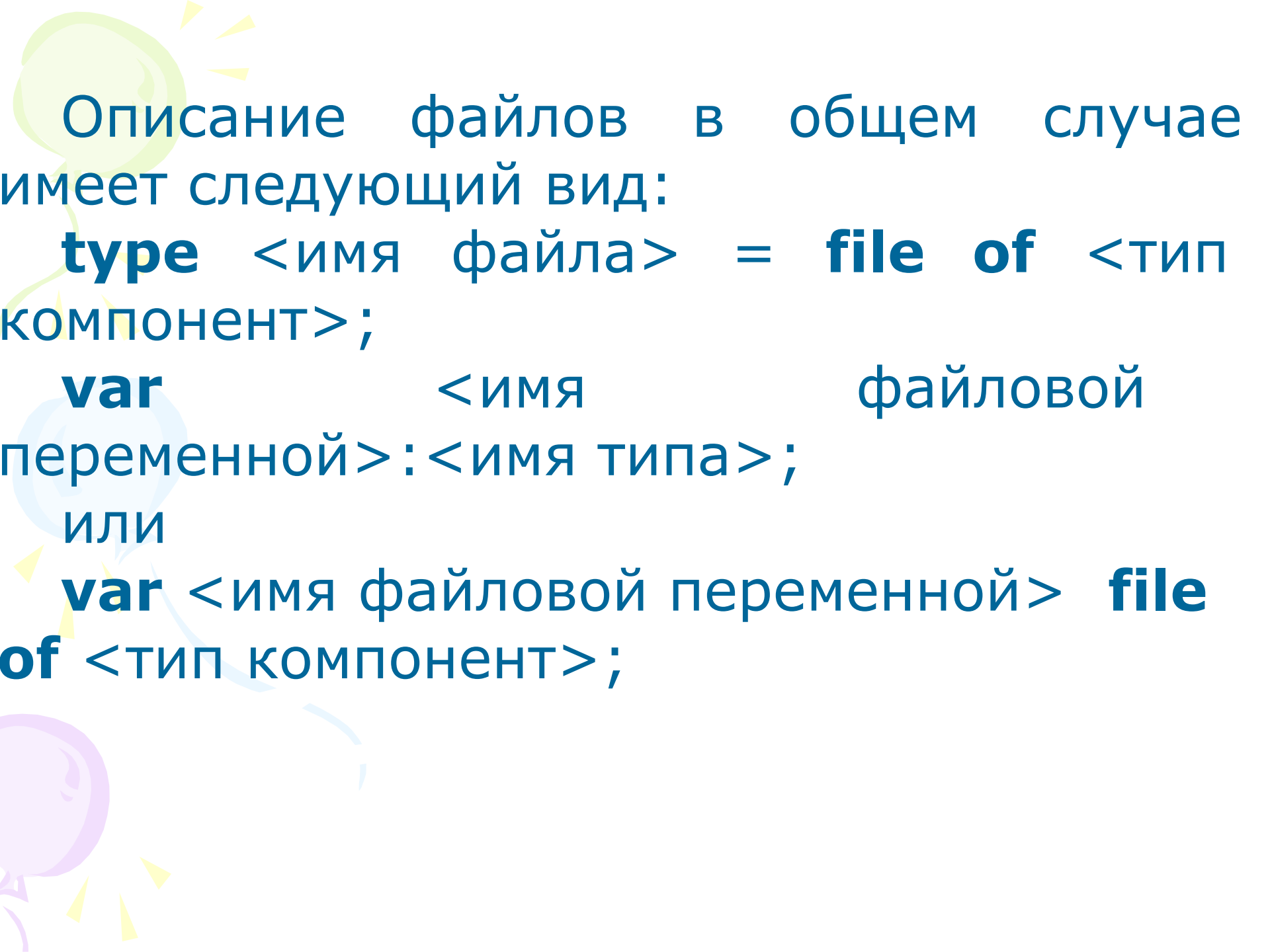
Запись удобна особенно тогда, когда объем информации велик и в дальнейшем предполагается использовать эту информацию в других программах.

В Паскале предусмотрен специальный тип данных – файлы, операции над которыми сводятся к работе с внешними носителями.



Файл – это поименованная область памяти на каком-либо носителе информации, предназначенная для хранения данных. Этим носителем может быть гибкий или жесткий диск (магнитная лента).

Внешние файлы должны быть описаны в разделе описаний программы.



Описание файлов в общем случае имеет следующий вид:

type <имя файла> = **file of** <тип компонент>;

var <имя файловой переменной> : <имя типа>;

или

var <имя файловой переменной> **file of** <тип компонент>;

Файловая переменная (обозначим ее как f) служит для доступа к файлу.

В Турбо-Паскале существуют следующие категории файлов:

- типизированные;
- нетипизированные;
- текстовые.

В зависимости от категории
объявление файлов соответственно
будет:

```
var f1 : file of <ТИП КОМПОНЕНТ>;
```

```
f2 : file;
```

```
f3 : text.
```

Например:

```
var f1:file of char;
```

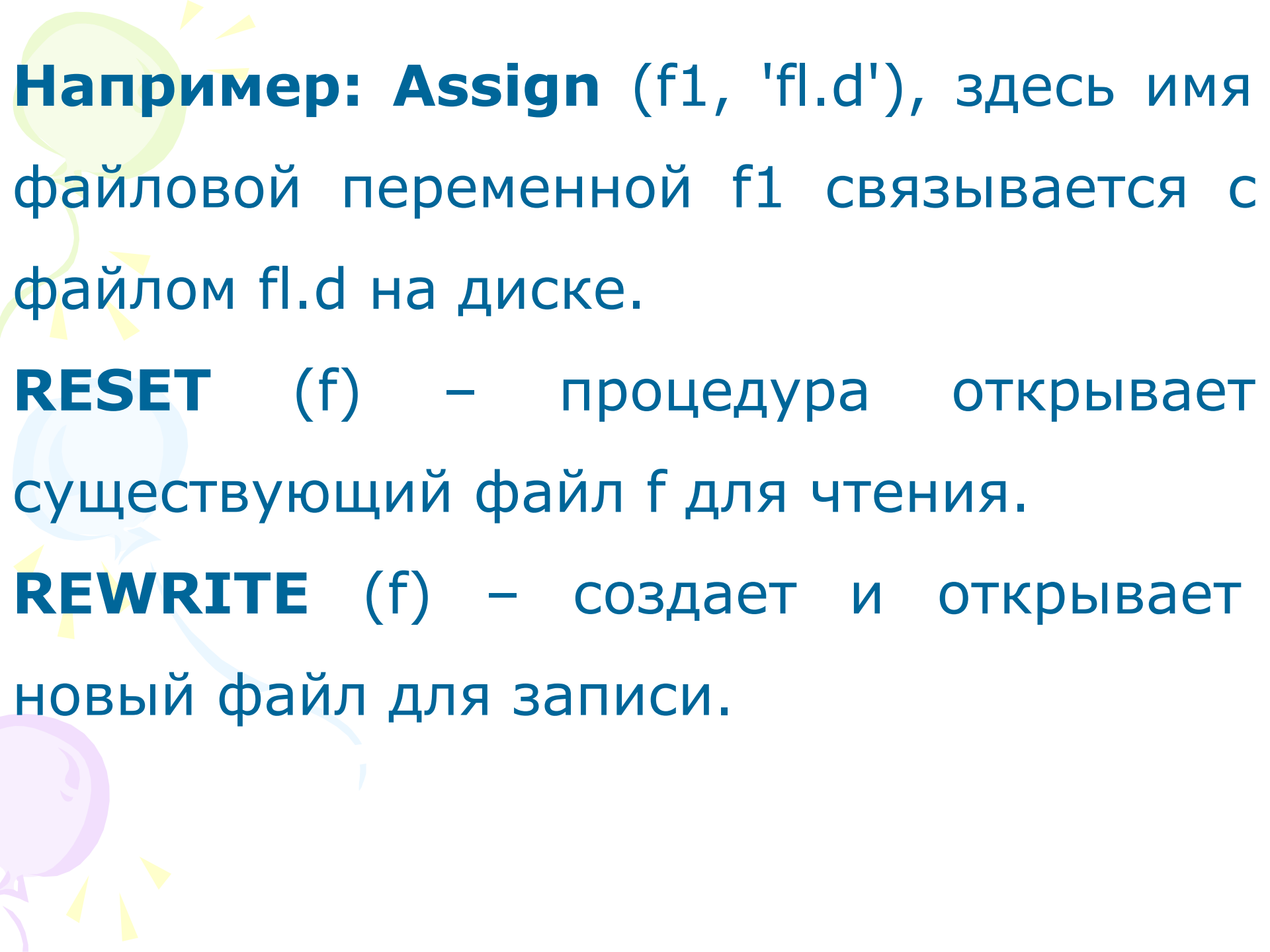
```
f2:file;
```

```
f3:text.
```

Стандартные процедуры для работы с файлами.

Работа с файлами производится посредством специальных стандартных процедур. Рассмотрим некоторые из них.

`ASSIGN (f, '<имя внешнего файла>')` – эта процедура связывает файловую переменную `f` с именем внешнего файла на диске.



Например: Assign (f1, 'fl.d'), здесь имя файловой переменной f1 связывается с файлом fl.d на диске.

RESET (f) – процедура открывает существующий файл f для чтения.

REWRITE (f) – создает и открывает новый файл для записи.

APPEND (f) – открывает существующий файл для добавления данных.

READ (f, X1,...,Xn) или **READLN** (f, X1,...,Xn) – считывает из файла f значения переменных X1 ... Xn.

WRITE (f, X1,...,Xn) или **WRITELN** (f, X1,...,Xn) – записывает в файл f значения переменных X1 ... Xn.

CLOSE (f) – закрывает файл f после окончания работы с ним.



Особым типом файлов являются ***текстовые файлы.***

Эти файлы содержат некоторый текст, который состоит из обычных символов (например, букв алфавита и цифр).

Символы текстового файла разбиты на строки.

Описание текстового файла:

Var <имя файла>: text;

Текстовый файл состоит из последовательности строк различной длины. Для определения конца строки используется функция

Eoln(var F:text) : Boolean;

Она принимает значение **True**, если достигнут конец строки, и значение **False** – в противном случае.

Для чтения из текстового файла или записи в текстовый файл можно использовать процедуры **Write (f, X1,...,Xn), WriteLn(f, X1,...,Xn) Read(f, X1,...,Xn) , ReadLn (f, X1,...,Xn) .**

Следует отметить, что, несмотря на то, что текстовый файл является набором символьных значений, он может использоваться (и часто используется) для хранения численных значений.

При считывании или записи значений в файл происходит автоматическое преобразование из числового формата в символьный и наоборот.

Продемонстрируем работу с файлами на примере.

Пример. Составить программу пересчета концентраций химических веществ, заданных в мольных долях, в весовые :

$$BD_i = \frac{MD_i \cdot MB_i}{\sum MD_i \cdot MB_i} ; i = 1, \dots, 5,$$

где

BD_i – концентрация в весовых долях;

MD_i – концентрация в мольных долях;

MB_i – молекулярный вес веществ.

Исходные данные ввести из файла,
результат вычислений поместить в
файл.

Программный файл.

```
Program Conz;
```

```
  type mas=array[1..10]of real;
```

```
  var BD,MD,MB:mas;
```

```
    s:real;
```

```
    i:integer;
```

```
    f1,f2:text;    {объявление
```

```
файлов }
```

Begin

```
Assign (f1, 'dat');  
Assign (f2, 'rez');  
Reset (f1); Rewrite (f2);  
{ввод данных из файла dat}  
for i:=1 to 5 do  
read (f1, MD[i]);  
readln (f1);  
for i:=1 to 5 do  
read (f1, MB[i]);  
s:=0.0;
```



```
for i:=1 to 5 do
```

```
  s:=s+MD[i]*MB[i];
```

```
for i:=1 to 5 do
```

```
  BD[i]:=MD[i]*MD[i]/s;
```

```
  {вывод результатов в файл
```

```
rez}
```

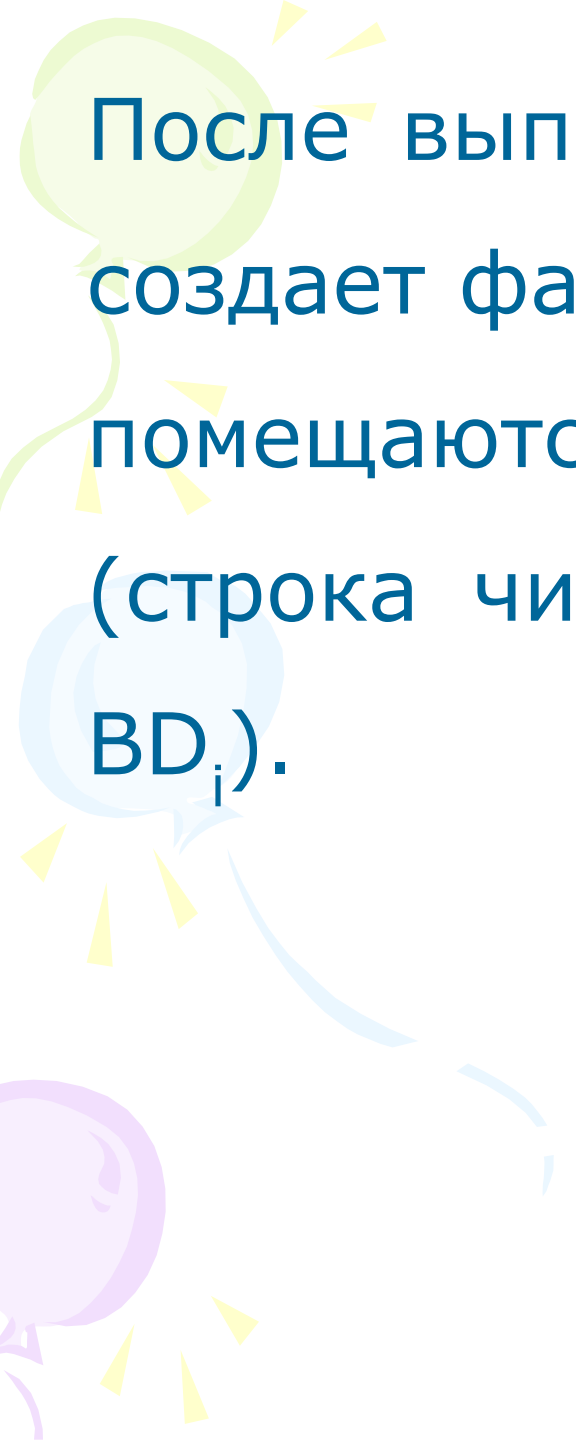
```
for i:=1 to 5 do
```

```
  write(f2,BD[i]:6:2,' ');
```

```
  close(f1); close(f2);
```

```
End.
```

После написания и сохранения программного файла в новый файл согласно последовательности ввода данных в программе (1-я строка – массив значений концентраций MD_i ; 2-я строка – массив значений молекулярных весов MB_i) записывают через пробел исходные численные данные и сохраняют файл под именем **dat**.



После выполнения вычислений ПЭВМ создает файл с именем **rez**, в который помещаются результаты расчетов (строка численных значений массива BD_i).