

# Курс «Паскаль. Программирование на языке высокого уровня»

---

Павловская Т.А.

Павловская Т.А. Паскаль. Программирование на языке высокого уровня. Учебник. 2-е изд. — СПб.: ПИТЕР, 2010. — 464 с.

Павловская Т.А.  
(СПбГУИТМО)

1

# Лекция 3. Типы данных, определяемые программистом

---

**Описываются одномерные и двумерные массивы,  
строки, записи, множества и файлы.**

Павловская Т.А.  
(СПбГУИТМО)

---

# Простые типы данных

Павловская Т.А.  
(СПбГУИТМО)

# Описание типа данных

type

имя\_типа = описание\_типа

...

var

имя\_переменной : имя\_типа

var

имя\_переменной : описание\_типа

# Перечисляемый тип данных

**type**

**имя = (список имен констант)**

type

Menu = (READ, WRITE, EDIT, QUIT)

```
var m, n : Menu;
```

```
...
```

```
m := READ; n := m;
```

# Интервальный тип данных

**type имя = конст\_1 .. конст\_2**

```
type Hour = 0 .. 23;  
    Range = -100 .. 100;  
    Letters = 'a' .. 'z';  
    Actions = READ .. EDIT;
```

```
var r : -100 .. 100;
```

---

# Массивы

Павловская Т.А.  
(СПбГУИТМО)

# Описание массива

**type имя\_типа = array  
[тип\_индекса] of тип\_элемента**

```
type mas = array [1 .. 6] of real;  
Color = array [byte] of mas;  
A = array [Menu] of boolean;
```

```
var c : mas;  
a, b : array [1 .. n] of integer;
```

```
const a : mas = (0, 5.2, -7.1, 100, 15, 1);
```



# Пример 1

```
Program Max_Elem;                                { макс. элемент }
const n = 20;

var a : array [1 .. n] of real;
    i : integer;
    max : Real;
begin
    writeln('Введите ', n, ' элементов массива');
    for i := 1 to n do read(a[i]);

    max := a[1];
    for i := 2 to n do
        if a[i] > max then max := a[i];
    writeln('Максимальный элемент: ', max:6:2)
end
```

# Пример 2

{ **Кол-во отрицательных** и **общая сумма элементов** }

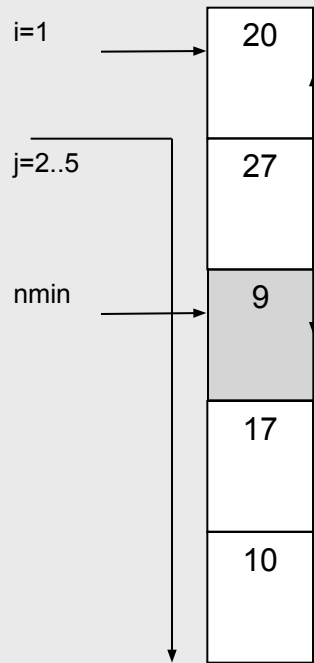
```
Program Sum_Num;  
const n = 10;  
var a : array [1 .. n] of integer;  
    i, sum, num : integer;  
begin  
    writeln('Введите ', n, ' элементов массива');  
    for i := 1 to n do read(a[i]);  
  
    sum := 0;  
    num := 0;  
    for i := 1 to n do begin  
        if a[i] < 0 then inc(num);  
        sum := sum + a[i];  
    end;
```

Павловская Т.А.  
(СПбГУИТМО)

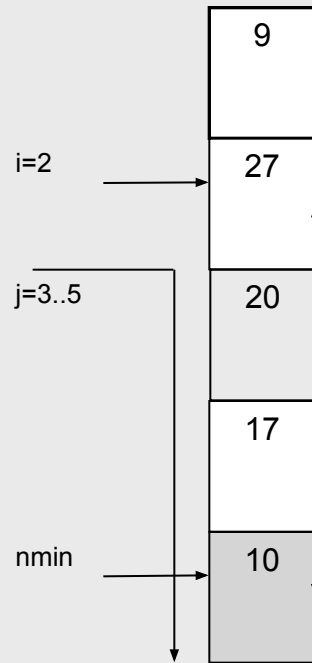
10

# Сортировка выбором

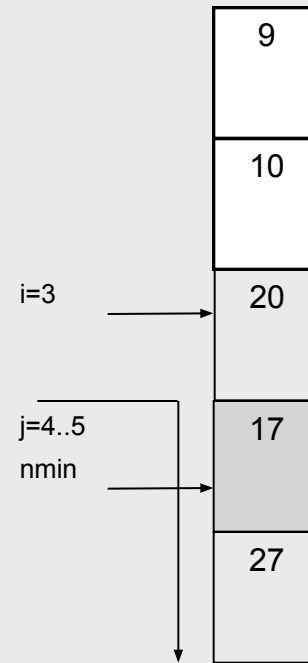
1-й просмотр:



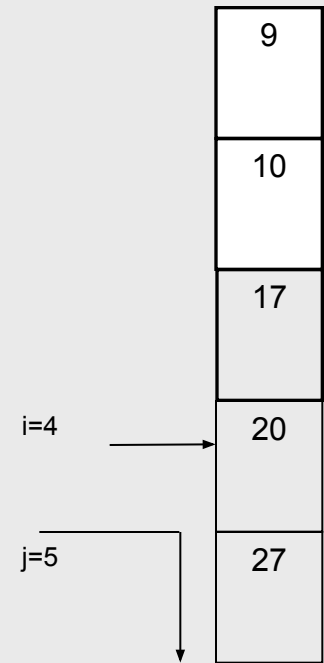
2-й просмотр:



3-й просмотр:



4-й просмотр:

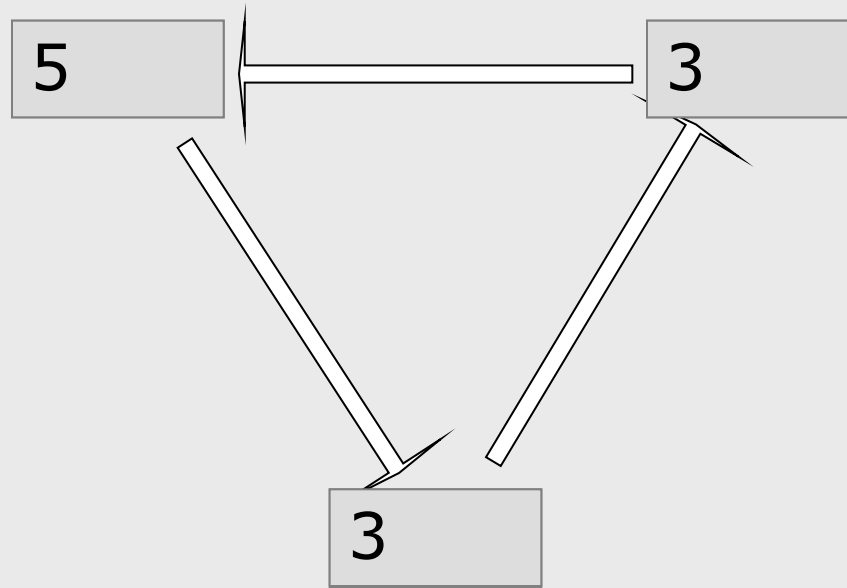


# Алгоритм сортировки

Повторить  $(n-1)$  раз ( $i := 1$  to  $n-1$ ):

- Среди элементов, начиная с  $i$ -го, найти, где расположен минимальный элемент массива
- Поменять его местами с  $i$ -м элементом.  $i$ -й элемент теперь на нужном месте.

# Обмен значений двух переменных



# Сортировка выбором

```
Program Sort;  
const n = 20;  
var a : array [1 .. n] of integer;  
    i, j, nmin, buf : integer;  
begin  
    writeln('Введите ', n, ' элементов массива');  
    for i := 1 to n do read(a[i]);  
  
    for i := 1 to n - 1 do begin  
        nmin := i;  
        for j := i + 1 to n do  
            if a[j] < a[nmin] then nmin := j;  
        buf := a[i];    a[i] := a[nmin];    a[nmin] := buf;  
    end;  
  
    writeln('Упорядоченный массив:');  
    for i := 1 to n do write(a[i], 5)  
end.
```

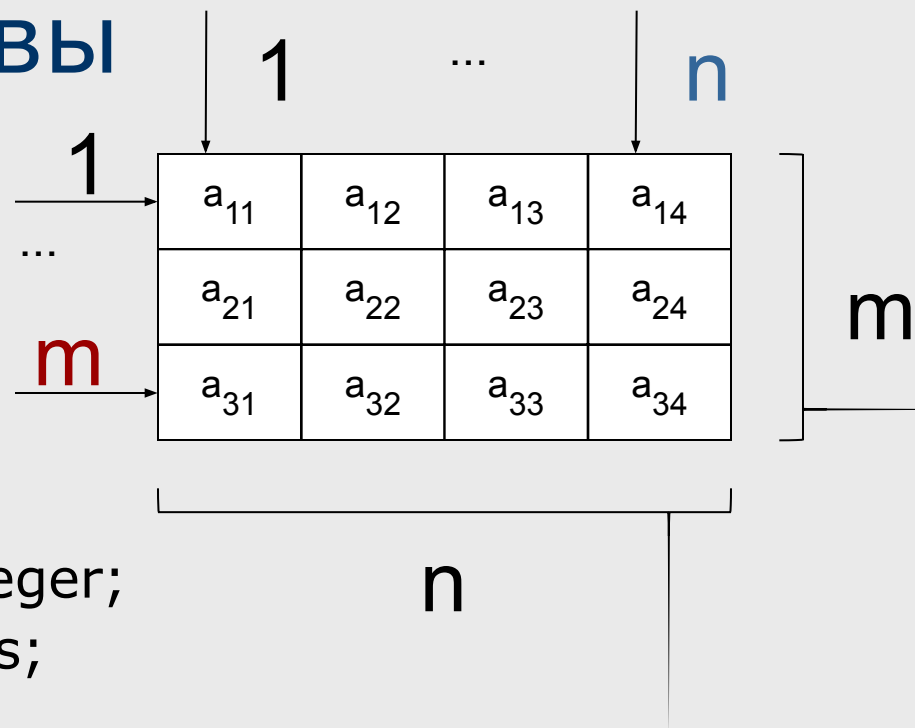
# Сортировка выбором

```
Program Sort;
const n = 20;
var a : array [1 .. n] of integer;
    i, j, nmin, buf : integer;
begin
    writeln('Введите ', n, ' элементов массива');
    for i := 1 to n do read(a[i]);

    for i := 1 to n - 1 do begin
        nmin := i;
        for j := i + 1 to n do
            if a[j] < a[nmin] then nmin := j;
        buf := a[i];    a[i] := a[nmin]; a[nmin] := buf;
    end;

    writeln('Упорядоченный массив:');
    for i := 1 to n do write(a[i]:5)
end.
```

# Двумерные массивы



```
const n = 4; m = 3;
```

```
type
```

```
  mas  = array [1 .. n] of integer;
```

```
  mas2 = array [1 .. m] of mas;
```

-----

```
type mas2 = array [1 .. m, 1 .. n] of integer;
```

```
var a, b : mas2;
```

```
a11 a12 a13 a14 a21 a22 a23 a24 a31 a32 a33 a34  
- 1-я строка - | - 2-я строка - | - 3-я строка - |
```



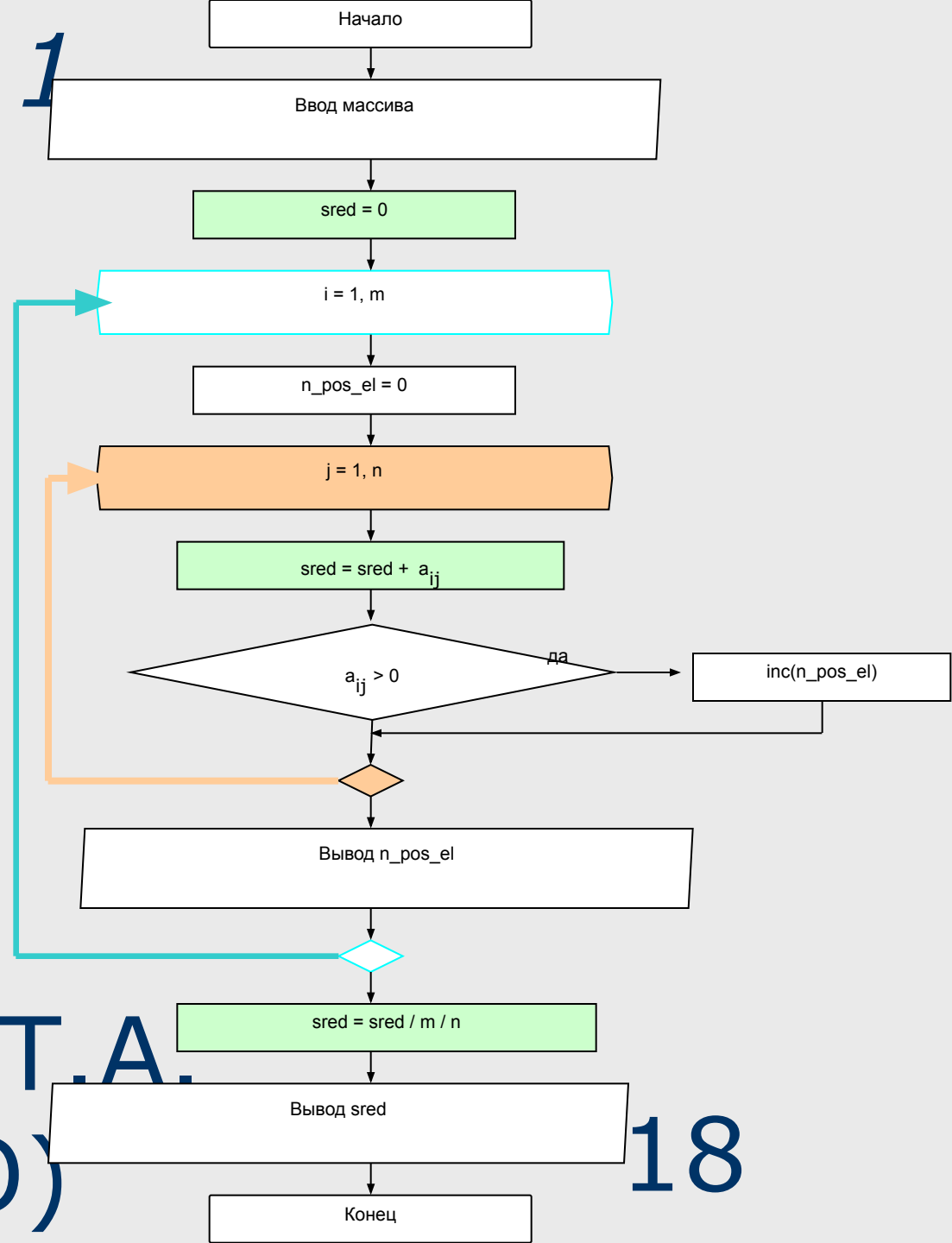
# Инициализация массива

```
const a : mas2 = ( ( 2, 3, 1, 0),  
                  ( 1, 9, 1, 3),  
                  ( 3, 5, 7, 0) );
```

```
for i := 1 to m do  
  for j := 1 to n do read (a[i, j]);
```

# Пример 1

Программа,  
которая для  
целочисленной  
матрицы 3 x 4  
определяет  
среднее  
арифметическое  
ее элементов и  
количество  
положительных  
элементов в  
каждой строке.



```

program sred_n;
const m = 3; n = 4;
var a : array [1 .. m, 1 .. n] of integer;
    i, j, n_pos_el : integer;
    sred : real;
begin
  for i := 1 to m do
    for j := 1 to n do read(a[i, j]);
    sred := 0;
    for i := 1 to m do begin
      n_pos_el := 0;
      for j := 1 to n do begin
        sred := sred + a[i, j];
        if a[i, j] > 0 then inc(n_pos_el);
      end;
      writeln('В ', i, '-й строке ', n_pos_el,
        ' положительных элементов');
    end;
    sred := sred / m / n;
    writeln('Среднее арифметическое: ', sred:6:2);
  end.

```

Павловская Т.А.

(СЛУЖИМО)

---

**Строки**

Павловская Т.А.  
(СПбГУИТМО)

20

# Типы строк

В Паскале три типа строк:

- стандартные (`string`);
- определяемые программистом на основе `string`;
- строки в динамической памяти.

Строка типа `string` может содержать до 255 символов. Под каждый символ отводится по 1 байту, в котором хранится код символа. Еще один байт отводится под фактическую длину строки

# Описание строк

```
type str5 = string [5];
```

```
const n = 10;
```

```
var s : string; { строка стандартного типа }
```

```
    s1 : str5;   { строка типа str5 }
```

```
    s2 : string [n]; { описание типа задано при  
описании переменной }
```

# Операции со строками

- Присваивание  
    `s := s1;`
- Конкатенация  
    `s1 := 'кар' + 'туз';`
- Сравнение  
    `if s3 > s1 then ...`
- Ввод-вывод – как целиком, так и посимвольно.

# Процедуры и функции

- Функция `Concat(s1, s2, ..., sn)`
- Функция `Copy(s, start, len)`
- Процедура `Delete(s, start, len)`
- Процедура `Insert(subs, s, start)`
- Функция `Length(s)`
- Функция `Pos(subs, s)`
- Процедура `Str(x, s)`
- Процедура `Val(s, x, errcode)`



# Пример 1

*Написать программу, которая определяет, встречается ли в заданном текстовом файле заданная последовательность СИМВОЛОВ.*

*Текст не содержит переносов слов, длина строки текста не превышает 80 символов.*

# Алгоритм

1. Построчно считывать текст из файла.
2. Для каждой строки проверять, содержится ли в ней заданная последовательность.
3. Если да, напечатать сообщение о наличии заданной последовательности и завершить программу.
4. При нормальном выходе из цикла напечатать сообщение об отсутствии заданной последовательности и завершить программу.

# Программа

```
program search_substr;
const len = 80;                                { 1 }
var
  word, line : string[len];                    { 2 }
  fin : text;
begin
assign(fin, 'text.txt'); reset(fin);
writeln('Введите слово для поиска:'); readln(word);

  while not eof(fin) do begin                  { 3 }
    readln(fin, line);
    if pos(word, line) <> 0 then begin         { 4 }
      writeln('Присутствует!'); exit end;
    end;
  writeln('Отсутствует!');
end.
```

# Пример 2

*Программа, которая читает текст из файла и выводит его на экран, заменяя заданную с клавиатуры последовательность символов на многоточие.*

# Программа

```
Program Change_word;
var s, str : string[80];    f : text;    i, dl : integer;

begin
  assign(f, 'primer.txt'); reset(f);
  writeln('Какое слово заменять?'); readln(s);
  dl := length(s);
  while not Eof(f) do begin
    readln(f, str);
    i := 1;
    while i <> 0 do begin
      i := Pos(s, str);
      if i <> 0 then begin    Delete(str, i, dl);
                            Insert('...', str, i); end;
    end;
    writeln(str);
  end;
  close(f);
end;
```

Павловская Т.А.  
(СПбГУИТМО)

---

# Записи и множества

Павловская Т.А.  
(СПбГУИТМО)

30

# Описание записи

```
type имя_типа = record  
    описание 1-го поля записи;  
    описание 2-го поля записи;  
    ...  
    описание n-го поля записи;  
end;
```

# Примеры описания записей

```
type goods = record  
  name : string [20];  
  price : real;  
  number : integer;  
end;
```

```
var g1, g2 : goods;
```

```
  stock : array [1 .. 100] of goods;
```

```
  student : record  
    name : string [30];  
    group : byte;  
    marks : array [1 .. 4] of byte;
```

```
end;
```



# Действия с записями

```
g1 := g2;  
g2 := stock[3];  
g1.price := 200;
```

```
with g1 do begin  
    price := 200; number := 10  
end;
```

Инициализация записей:

```
const g : goods = ( name : 'boots';  
                    price : 200;  
                    number : 10 );
```

# Пример использования записей

Сведения о товарах на складе хранятся в текстовом файле. Для каждого товара отводится одна строка, в первых 20 позициях которой записано наименование товара, а затем через произвольное количество пробелов его цена и количество единиц.

Программа по запросу выдает сведения о товаре или сообщение о том, что товар не найден.

```
Program SKLAD;  
const Max_n = 100;  
type  
    str20 = string [20];  
    goods = record  
        name : str20;  
        price : real;  
        number : integer;  
    end;
```

```
var stock : array[1 .. Max_n] of goods;
    i, j, len : integer;
    name : str20;
    found : boolean;
    f : text;

begin
    assign(f, 'stock.txt'); reset(f);
    i := 1;
    while not Eof(f) do begin
        with stock[i] do readln(f, name, price, number);
        inc(i);

        if i > Max_n then begin
            writeln('Переполнение массива'); exit end;
        end;
```



```
while true do begin
    writeln('Введите
    наименование');
    Readln(name); len := length(name);
    if len = 0 then break;
    for j := len + 1 to 20 do
        name := name + ' ';
    found := false;
    for j := 1 to i - 1 do begin
        if name <> stock[j].name then continue;
        with stock[j] do
            writeln (name:22, price:7:2, number:5);
            found := true;
            break;
        end;
    if not found then writeln ('Товар не найден');
end;
end;
```

# Записи с вариантной частью

```
type contact = record
  name : string [40];
  tel : string [15];
  case i : integer of
    0: (post: string [20]);
    1: (date: string [10]; code: word);
  end;
```



# Пример вариантной записи

type

```
figure = (rect, triangle, circle);
```

```
shape = record
```

```
x, y : real;
```

```
case kind : figure of
```

```
    rect      : (height, width : real);
```

```
    triangle  : (x2, y2, x3, y3 : real);
```

```
    circle    : (radius : real);
```

```
end;
```

# Множества

## Type

**имя\_типа = set of базовый\_тип;**

type Caps = set of 'A'..'Z';

Colors = set of (RED, GREEN, BLUE);

Numbers = set of byte;

var oct : set of 0..7;

# Константы и переменные

```
['A', 'D']    [1, 3, 6]
```

```
[2, 3, 10 .. 13]  []
```

```
var m: set of 1 .. 3;
```

```
[ ] [1]  [2]  [3]  [1,2] [1,3] [2,3] [1,2,3]
```



# Операции с множествами

Знак	Название
$:=$	присваивание
$+$	объединение
$*$	пересечение
$-$	вычитание
$=$	тождественность
$\langle \rangle$	нетождественность
$\leq$	содержится в
$\geq$	содержит
$\in$	принадлежность

# Пример работы с множествами

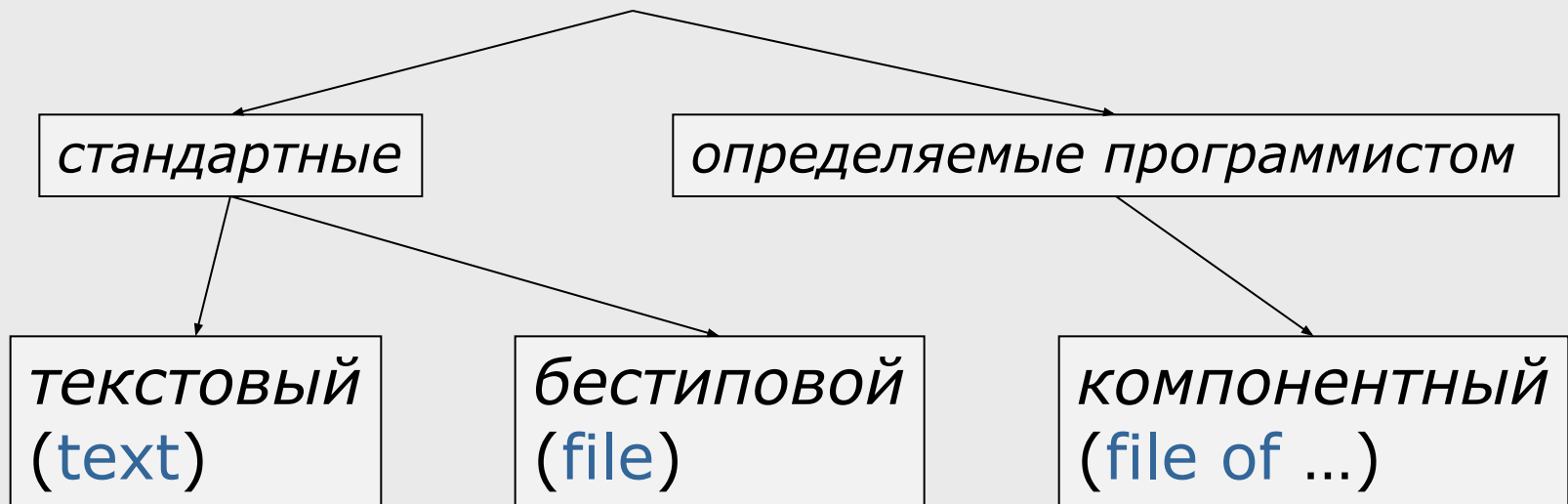
```
type Caps = set of 'A'..'Z';
var a, b, c : Caps;
begin
  a := ['A', 'U' .. 'Z'];    b := [ 'M' .. 'Z'];
  c := a; { присваивание }
  c := a + b; { объединение, результат ['A', 'M' .. 'Z'] }
  c := a * b; { пересечение, результат ['U' .. 'Z'] }
  c := b - a; { вычитание, результат ['M' .. 'T'] }
  c := a - b; { вычитание, результат ['A'] }
  if a = b then writeln ('тождественны'); { не выполнится }
  if a <> b then writeln ('не тождественны'); { выполнится }
  if c <= a then writeln ('c содержится в a'); { выполнится }
  if 'N' in b then writeln ('в b есть N'); { выполнится }
end.
```

---

# Файлы

Павловская Т.А.  
(СПбГУИТМО)

# Файлы



Пример описания файлов:

```
var ft : text;  
    fb : file;  
    fc : file of real;
```

Павловская Т.А.  
(СПбГУИТМО)

↑  
Компоненты могут быть любого типа, кроме  
файлового.

# Хранение данных:

- **Текстовые файлы** хранят данные в виде строк символов. При выводе данные преобразуются из внутренней формы представления в символьную, при вводе выполняется обратное преобразование.

- **Бестиповые и компонентные файлы** хранят данные в том же виде, в котором они представлены в оперативной памяти, то есть при обмене с файлом происходит побитовое копирование информации.

## Доступ к файлам:

- Последовательный
- Прямой

Файл:	текстовый	бестиповой	компонентный
Преобразование	+	-	-
Прямой доступ	-	+	+

# Организация ввода-вывода

1. объявить файловую переменную  
`var f : text;`
2. связать ее с физическим файлом  
`assign(f, 'vasia.txt');`
3. открыть файл для чтения и/или записи  
`rewrite(f);`
4. выполнить операции ввода-вывода  
`writeln(f, 'Здесь был Вася');`
5. закрыть файл  
`close(f);`

# Процедуры и функции

для работы со всеми типами файлов:

- `assign(var f; filename : string)`
- `close(var f)`
- `erase(var f)`
- `rename(var f; newname : string)`
- `reset(var f)`
- `rewrite(var f)`
- `eof(var f) : boolean`
- `IOresult : integer`

# Текстовые файлы

- Текстовый файл - последовательность строк символов переменной длины.
- Каждая строка заканчивается символами перевода строки и возврата каретки (коды — 13 и 10).
- Текстовый файл можно открыть не только для чтения или записи, но и для добавления информации в конец:  
`append(var f)`



# Подпрограммы для текстовых файлов

- `read(f, <список>)`
- `readln(f, [<список>])`
- `write(f, <список>)`
- `writeln(f, [<список>])`

# Подпрограммы для текстовых файлов

`flush(var f : text)`

применяется к открытым выходным файлам, принудительно записывает данные из буфера в файл независимо от степени его заполнения.

`settextbuf(var f : text; var buf; bufsize : word)`

служит для увеличения или уменьшения буфера ввода - вывода текстового файла `f`. Значение размера буфера для текстовых файлов по умолчанию равно 128 байтам. Увеличение размера буфера сокращает количество обращений к диску.

# Подпрограммы для текстовых файлов

`seekEof(var f : Text): boolean`

возвращает значение `True`, если до конца файла остались строки, заполненные пробелами.

`seekEoln(var f : text): boolean`

возвращает значение `True`, если до конца строки остались только пробелы.

---

# **Бестиповые и компонентные файлы**

Павловская Т.А.  
(СПбГУИТМО)

52

# Бестиповые файлы

Предназначены для хранения участков оперативной памяти на внешних носителях. После описания файловой переменной

```
var имя : file;
```

ее требуется связать с физическим файлом с помощью процедуры assign. Обмен производится через буфер «порциями», равными размеру буфера. Размер буфера, отличающийся от стандартного (128 байт), можно задать в reset или rewrite (от 1 до 64К):

```
reset(var f : file; bufsize : word)
```

```
rewrite(var f : file; bufsize : word)
```

Чтение и запись:

```
blockread(var f : file; var x; count : word; var num : word);
```

```
blockwrite(var f : file; var x; count : word; var num : word);
```

Чтение данных из бестипового файла осуществляется процедурой

```
BlockRead( var f: File; var X; Count: Word; var QuantBlock: Word );
```

Эта процедура осуществляет за одно обращение чтение в переменную X количества блоков, заданное параметром Count, при этом длина блока равна длине буфера. Значение Count не может быть меньше 1. За одно обращение нельзя прочесть больше, чем 64 К байтов.

Необязательный параметр QuantBlock возвращает число блоков (буферов), прочитанных текущей операцией BlockRead. В случае успешного завершения операции чтения QuantBlock = Count, в случае аварийной ситуации параметр QuantBlock будет содержать число удачно прочитанных блоков. Отсюда следует, что с помощью параметра QuantBlock можно контролировать правильность выполнения операции чтения.

Запись данных в бестиповой файл выполняется процедурой

```
BlockWrite( var f: File; var X; Count: Word; var QuantBlock:  
Word );
```

которая осуществляет за одно обращение запись из переменной X количества блоков, заданное параметром Count, при этом длина блока равна длине буфера.

Необязательный параметр QuantBlock возвращает число блоков (буферов), записанных успешно текущей операцией BlockWrite.

# Пример

Программа создает бестиповой файл,  
читая четверки вещественных чисел  
из текстового файла

```
Program Create_bfile;  
  
var   buf   : array[1 .. 4] of real;  
      f_in  : text;  f_out: file;  
      i, k  : integer;  
  
begin  
  
  assign(f_in, 'name_in'); reset(f_in);  
  assign(f_out, 'name_out');  
  rewrite(f_out, sizeof(real) * 4);
```



```
i := 0;

while not eof(f_in) do begin      inc(i);
read(f_in, buf[i]);

    if i = 4 then begin

        blockwrite(f_out, buf, 1);

        i := 0; end;
end;

if i <> 0 then begin

    for k := i + 1 to 4 do buf[k] := 0;

    blockwrite(f_out, buf, 1);

end;

close(f_in); close(f_out);

end.
```

# Компонентные файлы

Применяются для хранения однотипных элементов в их внутренней форме представления. Тип компонент задается после ключевых слов file of:

```
var имя : file of тип_компонент;
```

Компоненты могут быть любого типа, кроме файлового.

В операциях ввода-вывода могут участвовать только величины того же типа, что и компоненты файла:

```
type mas = array [1 .. 100] of real;  
var a, b : mas;  
    f : file of mas;  
begin  
    assign(f, 'some_file.dat'); rewrite(f);  
    ..  
    write(f, a, b);  
    close(f);  
end.
```

# Прямой доступ

- При последовательном доступе чтение/запись очередного элемента файла возможно только после аналогичной операции с предыдущим элементом.
- Бестиповые и компонентные файлы состоят из **блоков** одинакового размера. В бестиповом файле размер блока равен длине буфера, а в компонентном — длине компоненты. Это позволяет применить к таким файлам прямой доступ, при котором операции выполняются с заданным блоком.
- С помощью стандартной процедуры **seek** производится установка текущей позиции в файле на начало заданного блока, и следующая операция чтения/записи выполняется, начиная с этой позиции. Первый блок файла имеет номер 0.

# Пример

Программа считывает из бестипового файла, сформированного в предыдущем примере, требуемую запись

```
Program Get_bfile;  
  
varbuf : array[1 .. 4] of real;  
      f : file; i, k : integer;  
  
begin  
  
  assign(f, 'filename');  
  
  reset(f, sizeof(real) * 4);
```

```
while true do begin
    writeln('Введите номер или -1 для окончания');
    readln(k);
    if (k > filesize(f)) or
        (k < 0) then begin
        writeln('такой записи нет');
        exit end;
    seek(f, k);
    blockread(f, buf, 1);
    for i:= 1 to 4 do
        write(buf[i]:6:1);
    end;
    close(f);
end.
```