Лекция 7.
Строковый тип данных

В Паскале существует 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 – это байт под номером 0. Т.е. для хранения данных типа string отводится 256 байт.

Как описываются строки в ТР?

```
type str7= string [7];

const n = 10;

var  sa : string;    { строка стандартого типа }

       sb, sc : str7;       { строка типа str7 }

       sn : string [n];
```

Операции со строками

- Присваивание

  sa := sb;

- Конкатенация

  sb := ’кара’ + ’пуз’;

- Сравнение ‘aa’ < ‘ab’

  Write(‘Мышка’ < ‘Шишка’);

  Write(‘Big’ < ‘Биг’);

Ввод-вывод строк :

1) целиком Write(sb)

2) посимвольно Write(sn[1],sn[2], ….. sn[10])

Процедуры и функции TP, используемые для работы с данными строкового типа.

Concat(s1, s2,…, sn)	Функция	Конкатенация строк s1, s2, … sn
Copy(S, m, k)	Функция	Из строки S копирует k символов, начиная с m
Delete(S, m, k)	Процедура	Удаляет k символов из строки S начиная с m
Insert(subs, s, m)	Процедура	Вставляет subs в строку S начиная с номера m

Процедуры и функции TP, используемые для работы с данными строкового типа (продолжение).

Length(S)	Фун	Возвращает длину строки S. Length(S) = Ord(S[0])
Pos(subs, S)	Фун	Отыскивает в строке S первое вхождение строки subs и возвращает номер позиции, с которой она начинается.
Str(X, S)	Проц	Преобразует число X в строку символов S.
Val(S, X, error)	Проц	Преобразует строку S в целую или вещественную переменную X. Значение error = 0 при нормальном завершении.

Примеры

Действия со строками	Результат
S:=concat(‘Сивка’ ,’-’,‘Бурка’);	S = ‘Сивка-Бурка’
S1:= copy(S, 2, 4);	S1 = ‘ивка’
s2:= ‘7 раз’; m:= pos(‘7’,s2); Delete(s2,m,1) Insert(‘семь‘, s2, m);	S2=‘семь раз’
K:=Pos(‘a’,s2)	K=7
K:=Length(s1)+K	K=11
Str(Pi:6:2, s1);	S1 = ‘ 3.14’
Val(‘5.782’,x,code)	X=5.782; code = 0;

Определить, что будет выведено на экран в результате работы программы

```
Program wrt_k;
Const  a: array[1..8] of char =‘abcrcaab’;
Var    i: integer;   k: string;
Begin
     k:= “;
For i:=8 downto 2 do
   if a[i] < ‘c’ then k:=k+a[i];
write(k)
end.
```

Ответ: «baab»

Лекция 8.
Записи и множества.

Записи (records)

Запись – структура данных, состоящая из конечного числа компонентов, называемых полями.
Поля записи могут быть различных типов. Каждое поле имеет имя. Запись, как единое целое, занимает непрерывную область памяти.

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

описание поля записи: имя_поля : тип_поля;

Данные типа «запись» (примеры).

```
type    cars = record
                name : string [25];
                 price : real;
                 number : integer;
end;

var g1, g2 : cars;
            tabl : array [1 .. 100] of cars;
          student : record
                name : string [30];
                group : byte;
                marks : array [1 .. 5] of byte;
           end;
```

Примеры действий с записями.

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

```
With g1 do  begin
            price := 200;
                  number := 12;
                  end;
```

Инициализация записей:

```
const g : cars = ( name : =’opel’;
                   price := 25000;
                   number : =10 );
```

Примеры записей

```
Type Anketa = RECORD
                    name : string[20];
                  soname : string[20];
                    gend : char;
                   b_day : 1..31;
                 b_Month :  1..12;
                 b_year    :  word;
                    kurs :  1..5;
                   group  :  word;
                              END;
Var stud : anketa;     All_Stud:array[1..200] of stud;
                       или
Var All_Stud: array [1..200] of Anketa;
```

Как обратиться к полям записи?

<Имя_переменной>.<имя поля> := <выражение>

```
Stud.name:='Олег';
Stud.gend:='M';
Stud.kurs:=3;   … и т.д.
```

или

```
With Stud do   Begin
                name:='Олег';
                gend:='M';
                kurs:=3;
                End;
```

Множества

Type

имя_типа_множества = set of базовый_тип;

Пример:

```
type Caps = set of 'A'..'Z';
        Colors = set of (RED, GREEN, BLUE);
        Numbers = set of byte;
```

Количество элементов множества может меняться от 0 до 256.

Примеры множеств

```
Var A, B : Set  of  byte;
        s1, s2 ,s3: set of ‘a’ .. ‘z’;
Const D3=[1,3,6,9,12];
….
A:=[25,50,55,60];
B:=[ ];     {пустое множество}
S1:= [‘x’, ‘y’, ‘z’];
S2:= [ ‘a’, ‘b’, ‘y’];
B:=A;
```

Операции над множествами:

Операция	**Название**
A := B	Присваивание
A + B	Объединение A и B
A * B	Пересечение A и B
A – B	Вычитание B из A

Сравнение множеств(результат – логическое значение)

Операция	Название
A=B	Проверка множеств на эквивалентность; True если A и B совпадают.
A <> B	Проверка множеств на неэквивалентность; True если A и B несовпадают.
A <= B	Проверка на вхождение A в B. True, если A подмножество B.
A =>B	Проверка на вхождение B в A. True, если B подмножество A.
e in A	True, если e является элементом A

Лекция 9. Файлы в ТР.

- Общие приёмы для работы с файлами;
- Текстовые файлы;
- Типизированные файлы;

Файлы TP.

Файл – именованная область внешней памяти компьютера либо это логическое устройство – возможный приёмник или источник данных (CON, PRN, AUX,NUL).

В TP используются три типа файлов:

- Текстовые файлы
- Типизированные файлы
- Нетипизированные

Примеры:

```
VAR  <имя_ФП1>  : File of <тип>;  - типизированный файл;
     <имя_ФП2>  : File;           - нетипизирован. файл
     <имя_ФП3>  : TEXT;           - текстовый файл
<имя_ФП3>  : TEXTfile;    - текстовый файл (LAZARUS)
```

Преобразование данных при работе с файлами.

1. Преобразования данных при чтении – записи типизированных и нетипизированных файлов не происходит. Данные имеют одинаковое представление как в памяти компьютера, так и при хранении в файле;

2. При **чтении** из текстового файла данные преобразуются из символьного представления во внутреннее представление в памяти компьютера; при **записи** в текстовый файл – данные из внутреннего представления преобразуются в символьный формат.

Доступ к содержимому файлов

- Последовательный доступ – для текстовых файлов;
- Прямой доступ – для типизированных и нетипизированных файлов;

Все элементы, из которых состоит файл с прямым доступом нумеруются от 0 до N; в конце файла Находится специальный код =маркер конца файла. Для файлов с прямым доступом вводится понятие **текущей длины** – количество элементов файла. **Указатель файла** – адрес текущего элемента файла, предназначенного для обработки.

Порядок действий при работе с файлами в ТР

1. Задать файловую переменную;
2. Связать файловую переменную с файлом на диске или логическим устройством;
3. Указать направление обмена данными;
4. Осуществить чтение данных из файла или запись данных в файл;
5. Закрыть файл.

Как связать файловую переменную с файлом на диске?

После описания файловой переменной её надо связать с файлом на диске:

ASSIGN(ИФП, ИмяФайла_или_ЛогУстр);

ASSIGNFILE(ИФП, ИмяФайла_или_ЛогУстр);

пример:

```
Var     F1: File of string[60];
         F2: Text;
Begin …
Assign ( F1, ‘c:\works\F001.txt’);
Assign ( F2, ‘a:\mwk\My_abc.001’);

AssignFile ( F1, ‘c:\works\F001.txt’);       -LAZARUS !!!!
AssignFile ( F2, ‘a:\mwk\My_abc.001’);
```

fp – имя файловой переменной

Направление обмена	Текстовый файл	Типизир. файл	Нетипизир. файл
ReSet(fp) чтение из файла	Read или ReadLn	Read или Write	Read
ReWrite(fp) запись в файл	Write или WriteLn	Write	Write
Append(fp) дописать в конец файла	Write или WriteLn	X	X

Пример:

```
Program ExmpFile;
Var Fa : Text;
      Tx : String[50];
Const S='C:\MyDoc\abc001.txt';
Begin
Assign(Fa, S);
Reset(Fa);
While Not( EOF(fa)) do begin ReadLn(Fa, Tx);
                             WriteLn(Tx);
                       end;
Close(Fa);
End.
```

Файлы в LAZARUS (чтение текстового файла)

```
procedure TForm1.Button1Click(Sender: TObject);
Var fp:TextFile;
a,b,c:integer;
begin
  if OpenDialog1.Execute and FileExists(Opendialog1.FileName)
  then
  begin
    AssignFile(fp,Opendialog1.FileName);
    Reset(fp);
    Memo1.Lines.Clear;
    while not eof(fp) do begin
      Readln(fp,a,b,c);
      Memo1.Lines.Add(InttoStr(a)+'  '+InttoStr(b)+'  '+InttoStr(c));
    end;
    CloseFile(fp);
  end;
end;
```

Основные функции для работы с файлами

fp- файловая переменная

- Close(fp) –закрыть файл;
- Rename(fp,<Новое_имя>) – переименовать файл;
- Erase(fp) – удаляет файл (сначала его надо закрыть);
- Flush(fp) – запись всех изменений в файл на диске;
- EOF(fp) – логическая функция = True, если достигнут конец файла, иначе EOF = False;
- IOResalt – признак последней операции ввода-вывода. При успешном завершении IOResalt = 0.

Текстовые файлы.

- совокупность строк переменной длины. Последовательный доступ.
- Var fp : text;
- В конце каждой строки EOLN = #13#10
- В конце Файла EOF = #26
- Чтение: Read(fp, List) или ReadLn(fp, List)
- List содержит Char, String, <целые>, <вещественные>
- Запись: Write(fp,List) или WriteLn(fp,List)
- List содержит Char, String, BOOLEAN; <целые>, <вещественные>

Специальные функции для текстовых файлов

- EOLN(fp) – возвращает True, если в текстовом файле, из которого осуществляется ввод данных, достигнут маркер конца строки.
- SeekEOLN(fp) – ищет маркер конца строки и возвращает True, если до конца строки остались только пробелы.
- SeekEOF(fp) – ищет маркер конца файла и возвращает True, если до конца файла остались строки заполненные пробелами.

Типизированные файлы.

Предназначены для хранения однотипных элементов во внутренней форме представления. Тип элементов задаётся после ключевых слов FILE OF.
Var fp : file of <тип_элементов>;

Основные процедуры для работы с типизированными файлами

- Seek(fp,N) – делает текущим компонент N;
- FileSize(fp) – возвращает количество компонентов файла;
- FilePose(fp) – возвращает номер компонента, который будет обрабатываться при следующей операции;

Чтение-запись

Read(fp, <список>); Write(fp,<список>).

Типизированные файлы. Пример.

```
Program TPZ_f;
Const n=10;
Type student = record  name : string [30];    group : word;
                              marks : array [1 .. 5] of byte;
                    end;
 s_mas = array[1..n] of student;

Var    s1: student;   sall : s_mas;  f1: file of s_mas; i:word;
Begin  assign(f1,'stdata.tpz'); rewrite(f1);
for i:=1 to n do begin  with s1  do begin
write('name = ');  readln(name);  write('group=');
readln(group);
write(' marks=');
readln(marks[1], marks[2], marks[3], marks[4], marks[5]);
                                                  end;
   sall [ i ]:=s1; end;
    write(f1, sall);  Close(f1); end.
```

Лекция 10.

Примеры: некоторые алгоритмы и их программная реализация.

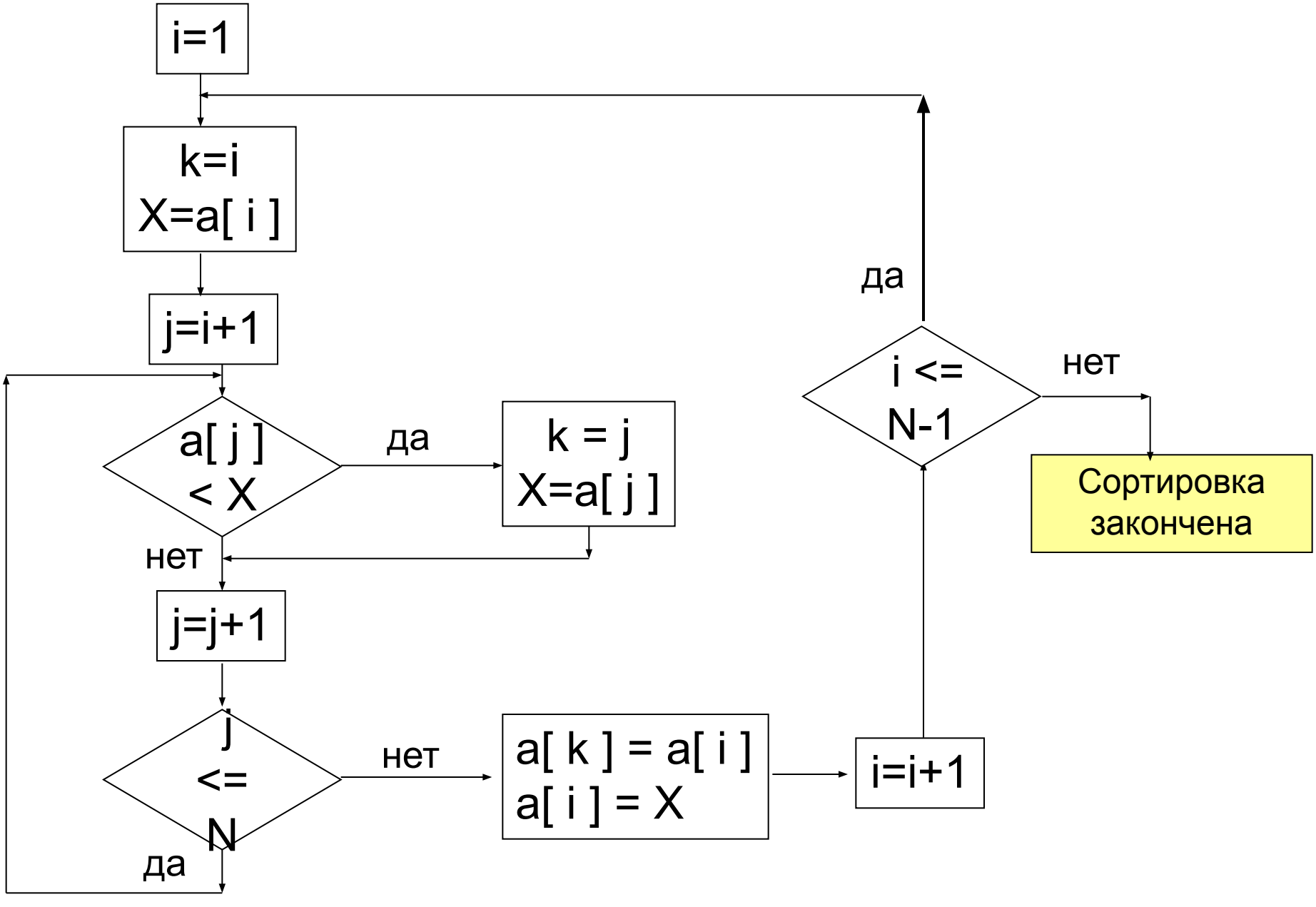
- Алгоритмы сортировки;
- Суммирование рядов;

Сортировка простым выбором.

Задача: Упорядочить числовой массив в порядке возрастания элементов. Заданы элементы массива и их количество.

Алгоритм: выберем наименьший элемент среди всех, начиная с первого, и поменяем его местами с первым элементом. Повторим эту процедуру начиная со второго элемента массива, и т.д.

Блок-схема сортировки простым выбором (по возрастанию).



Сортировка простым выбором.

```
Program Simpl_Sort;
const n = 10;
var a : array [1 .. n] of integer;
        i, j, k, x : integer;
begin
    writeln('Введите ', n, ' элементов массива');
    for i := 1 to n do read(a[i]);

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

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

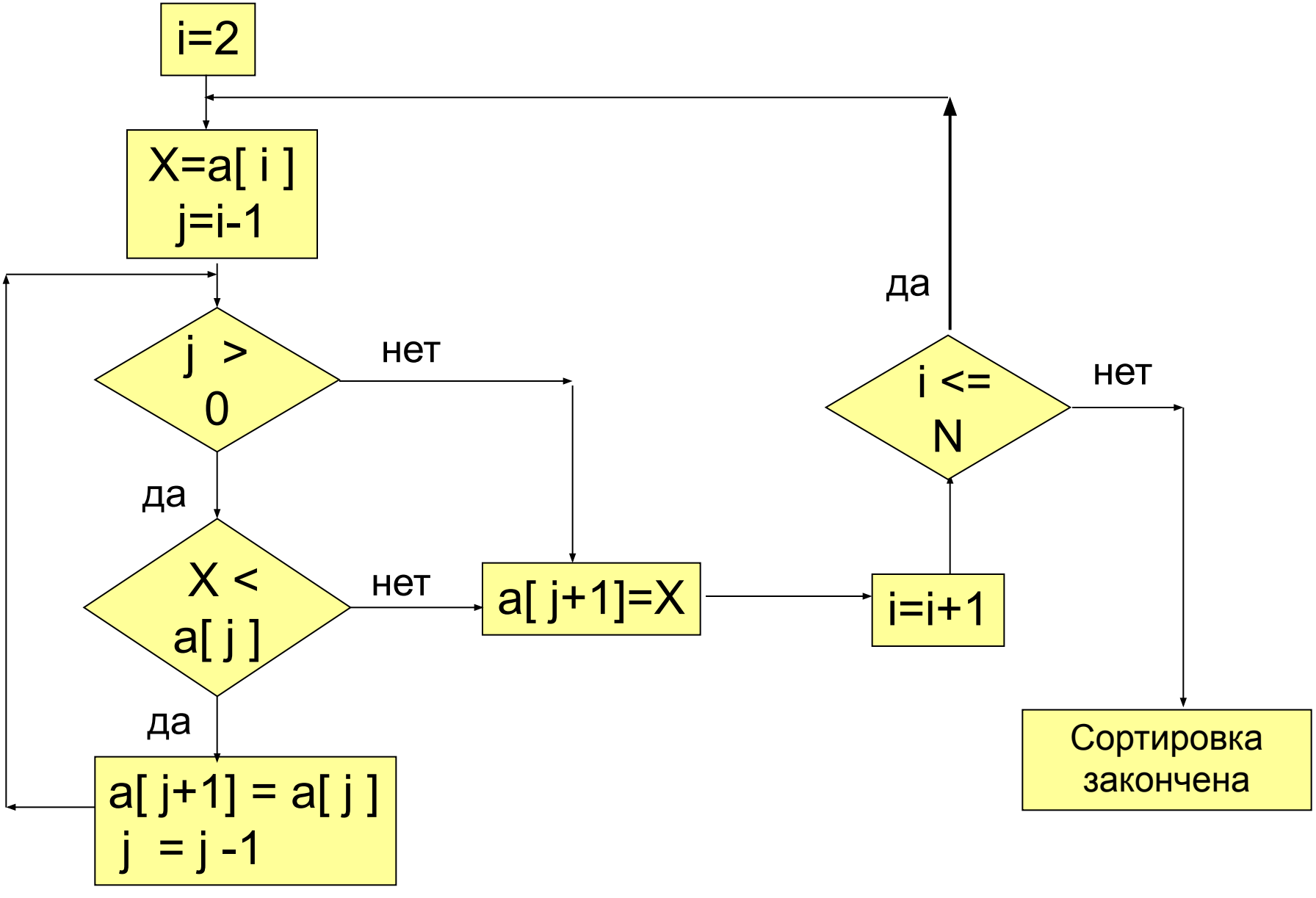
Сортировка простым включением.

Задача:
Упорядочить по возрастанию элементы массива.

Алгоритм решения задачи:

Пусть на некотором этапе работы алгоритма левая часть массива с 1-ого по (i-1) элементы уже отсортирована,
а правая часть с i –ого по N-ый осталась в прежнем виде.
Берём первый элемент правой (неотсортированной) части и вставляем в левую часть так, чтобы упорядоченность левой части сохранилась.
В начале алгоритма левая часть состоит
из одного элемента A[1].

Блок-схема сортировки простым включением (по возрастанию)



Сортировка простым включением.

```
Program Simpl_Ins_Sort;
const n = 10;
        a : array [1 .. n] of integer=(2, -3, 5, 9, …)
var   i, j, x : integer;
begin
    for i := 2 to n  do begin
        x:=a[ i ];     j:= i -1;
    while (x < a[ j ]) and ( j > 0) do
Begin   a [ j+1] := a[ j ];    j := j-1;  end;
                    a[ j+1]:=x
                        end;
    writeln('Упорядоченный массив:');
    for i := 1 to n do write(a[ i ]:5)
end.
```

Суммирование рядов с заданным числом слагаемых.

Задана формула для вычисления суммы заданного количества слагаемых.

$$S_n = \sum_{k=0}^{n} \frac{(-1)^k * x^{2k}}{k!} = 1 - x^2 + \frac{x^4}{2!} - \frac{x^6}{3!} + ...(-1)^n \frac{x^{2n}}{n!}$$

```
Program SUM_N;
Var S,x,C :  real;  k,n :  word;
Begin  write('Введите кол-во слагаемых n и X=');
readln(n, x);
 S:=1; C:=1;
For k := 1 to n do begin
                    C:=(-1)*x*x*C/k; S:=S+C; end;
Writeln('для заданного x сумма S(x) = ', S:10:5);
End.
```

Суммирование рядов с заданной точностью.

Признак Лейбница сходимости знакочередующихся рядов.

Задан знакочередующийся ряд:

$$S = U1 – U2 + U3 – U4 + U5 \ldots. \qquad (1)$$

Если члены знакочередующегося ряда

1) монотонно убывают: $U1 > U2 > U3 \ldots$
2) $\lim Un = 0$ при $n \to \infty$,

то ряд сходится и его сумма S отличается от n-ой частичной суммы не более чем на величину модуля первого отбрасываемого члена ряда.

Если в формуле (1) слагаемые U1, U2 … Un зависят от x, то ряд называется функциональным:

$$S(x) = U1(x) – U2(x) + U3(x) – U4(x) + U5(x) \ldots.$$

Алгоритм суммирования знакочередующихся рядов с заданной точностью.

- Попытаться выразить (n+1) слагаемое через (n) – ое:
  C(n+1) = M(x,n)*C(n)
- Вводим значение x, при котором требуется вычислить сумму ряда, и точность eps, с которой надо вычислить сумму.
- «Накапливаем» слагаемые ряда в переменной S до тех пор, пока очередное слагаемое по модулю не будет меньше или равно eps.
- Выводим на экран найденное значение суммы и количество слагаемых, которое потребовалось для достижения заданной точности.

Пример.

$$S = (x-1) - \frac{(x-1)^2}{2} + \frac{(x-1)^3}{3} - \frac{(x-1)^4}{4} + \ldots, \quad 0 < x \le 2.$$

$$A_1 = (x-1);$$
$$A_2 = A_1 * (-1) * (x-1)/2;$$
$$A_3 = A_2 * (-1) * (x-1) * 2/3; \ldots.$$

$$S \approx A_1 + A_2 + A_3 + \ldots . A_n, \quad \left|A_{n+1}\right| < eps.$$

```
Program row1;
var S,A,x,eps,m:real; k: byte;
begin
S:=0;       A:=-1;
m:=1.0;    k:=1;
write('  x   =');      readln(x);
If(x<=0) or (x>2) then begin
                                writeln(' bad value of x'); Halt(111) end;
write('  eps = ');    readln(eps);
while abs(a)>=eps do begin
A:=(-1)*A*(x-1)*m;   s:=s+a;
writeln(' k = ', k:3, '  S = ',s:8:5);
m:=k/(k+1); inc(k);      end;
writeln(' exact value  Ln(x) = ', ln(x):8:5);
readln;
end.
```