

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ  
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ  
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»  
(БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных  
систем

## **Лабораторная работа №12**

по дисциплине: Основы программирования

тема: «Динамические переменные»

Выполнил: ст. группы ПВ-201  
Машуров Дмитрий Русланович

Проверил:  
Притчин Иван Сергеевич

Белгород 2020 г.

## **Лабораторная работа №12**

### **«Динамические переменные»**

**Цель работы:** получение навыков работы с указателями и динамическими переменными структурированных типов.

#### **Задания для подготовки к работе:**

1. Изучите ссылочный тип и его использование для создания динамических переменных и работы с ними.
2. Рассмотрите возможные способы хранения матриц в динамически распределяемой области памяти. Изобразите схемы хранения для каждого случая.
3. Разработайте алгоритм и составьте программы для решения задачи соответствующего варианта для четырех случаев, матрицы следует разместить в "куче" при выполнении следующих условий: а) число строк и число столбцов – константы; б) число строк – константа, а число столбцов – исходное данное; в) число строк – исходное данное, число столбцов – константа; г) число строк и число столбцов – исходные данные.
4. Ввод, вывод и обработку матриц опишите отдельными подпрограммами. Для случаев а) – г), где возможно, используйте одни и те же подпрограммы.
5. В блок-схемах обработки матриц не используйте операции разыменования, а обращайтесь к элементам матрицы как к элементам двумерного массива.
6. Опишите блок-схему алгоритма решения задачи с использованием блоков «предопределенный процесс».
7. Закодируйте алгоритм.
8. Подберите наборы тестовых данных с обоснованием их выбора.

#### **Задания к работе:**

1. Наберите программу, отладьте ее и протестируйте.
2. Выполните анализ ошибок, выявленных при отладке программы

#### **Задание варианта №17:**

Дана матрица. Упорядочить ее строки по убыванию первых элементов строк, если это возможно.

## **Выполнение работы:**

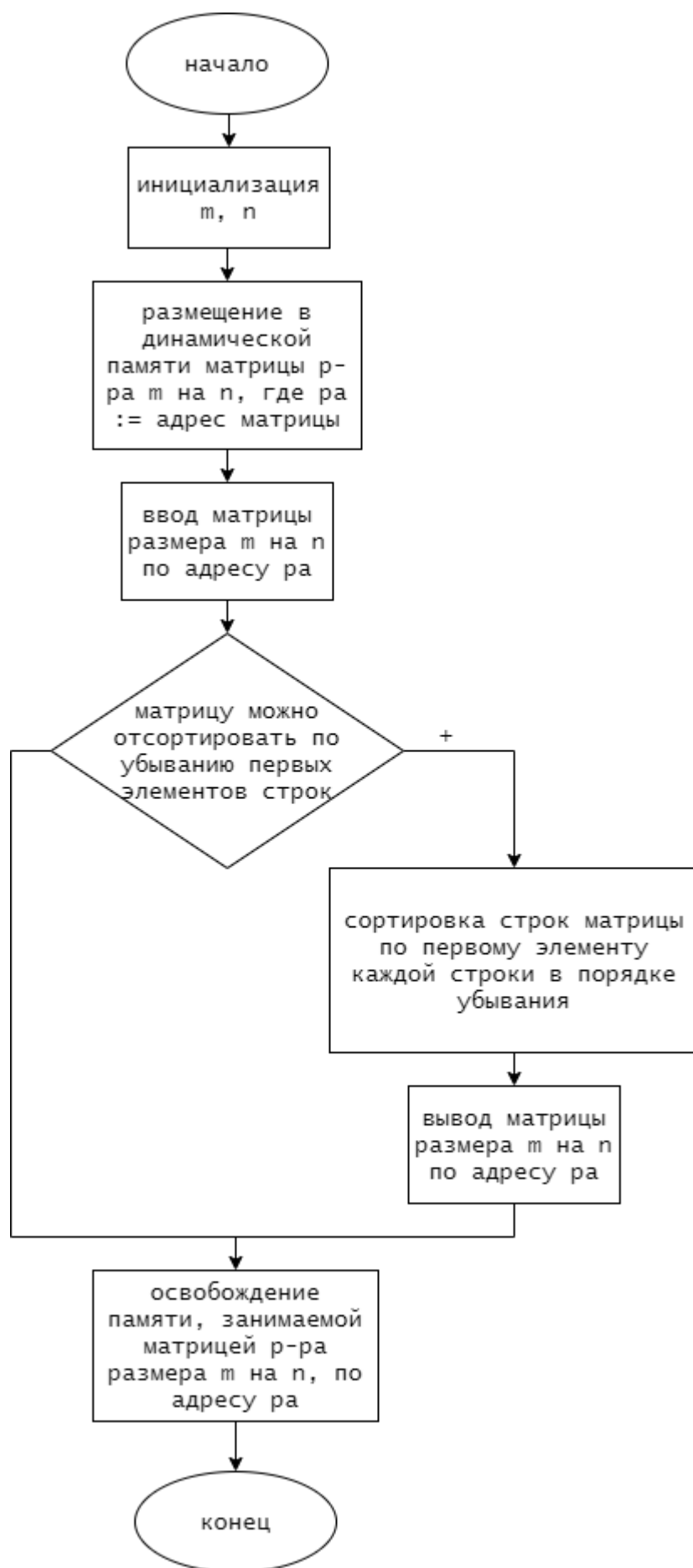
### **1. Выделение подзадач**

Выделим следующие подзадачи:

- Размещение в динамической памяти матрицы размера  $M \times N$ ,  $ra$  – адрес матрицы
- Ввод матрицы  $M \times N$  по адресу  $ra$
- Вывод матрицы  $M \times N$  по адресу  $ra$
- Проверка на то, можно ли упорядочить матрицу по убыванию первых элементов строк
- Упорядочивание строк матрицы по первому элементу каждой строки в порядке убывания
- Освобождение памяти занимаемой матрицей размера  $M \times N$  по адресу  $ra$

Далее описание алгоритма приводится в блок-схеме с укрупнёнными блоками в терминах выделенных подзадач

### **2. Блок-схема алгоритма в укрупнённых блоках:**

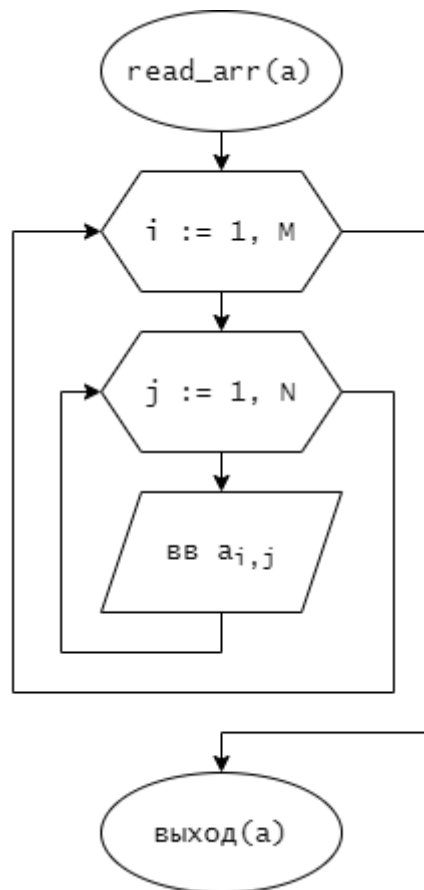


### 3. Описание подпрограмм:

Спецификация процедуры read\_arr

- 1) Заголовок: procedure read\_arr(var a: t\_matr)
- 2) Назначение: ввод матрицы a размера MxN
- 3) Входные параметры: нет
- 4) Выходные параметры: a

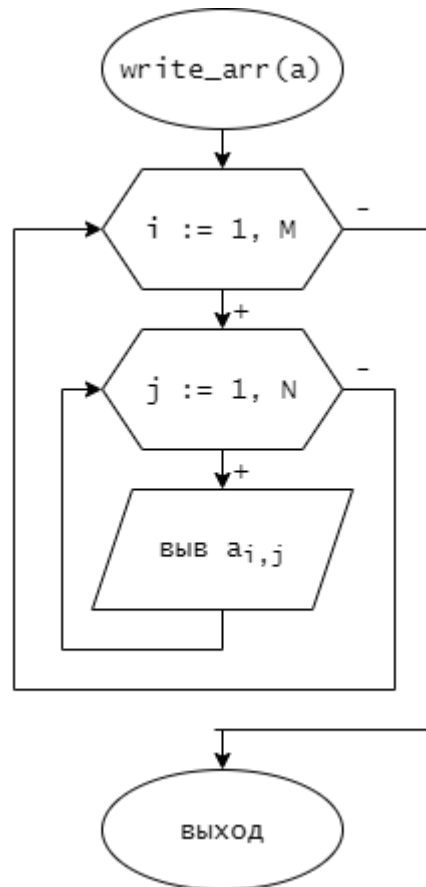
Блок-схема:



## Спецификация процедуры write\_arr

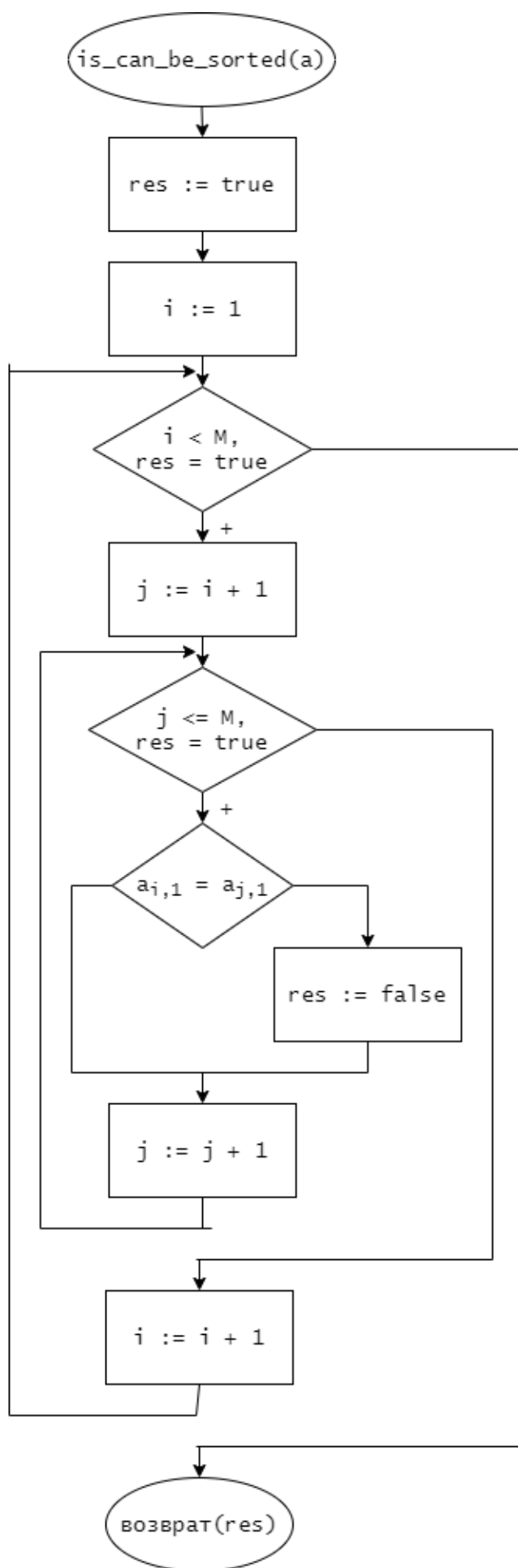
- 1) Заголовок: `procedure write_arr(a: t_matr)`
- 2) Назначение: вывод матрицы `a` размера  $M \times N$
- 3) Входные параметры: `a`
- 4) Выходные параметры: нет

Блок-схема:



## Спецификация функции `is_can_be_sorted`

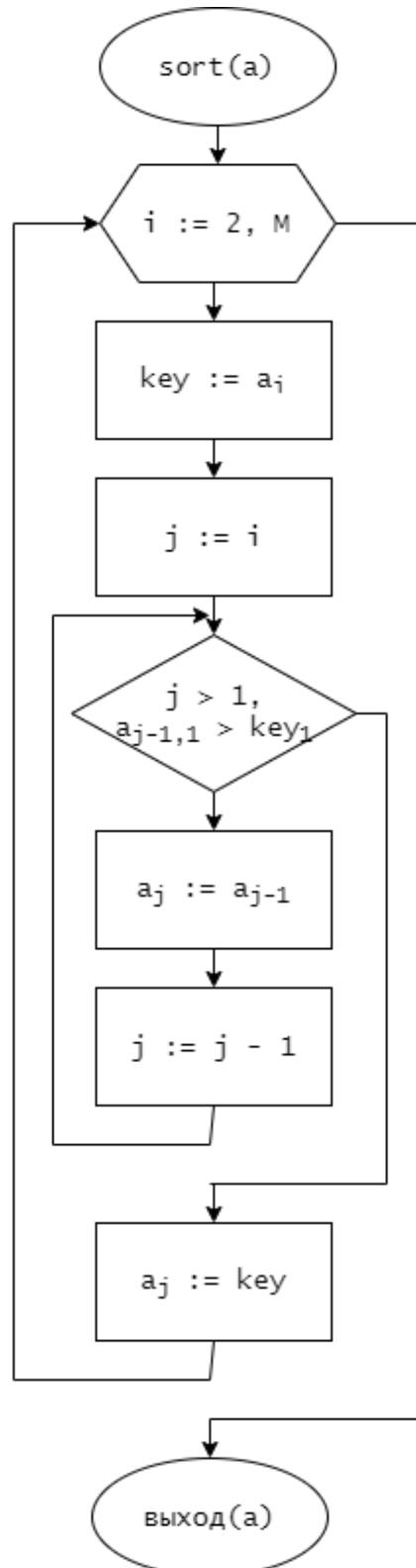
- 1) Заголовок: `function is_can_be_sorted(a: t_matr): boolean`
  - 2) Назначение: возвращает значение «истина», если матрицу `a` размера  $M \times N$  можно отсортировать в порядке убывания по первым элементам строк, иначе – «ложь». Применяется перед использованием процедуры `sort`
  - 3) Входные параметры: `a`
  - 4) Выходные параметры: нет
- Блок-схема:





## Спецификация процедуры sort

- 1) Заголовок: `procedure sort(var a: t_matr)`
  - 2) Назначение: сортировка матрицы `a` по первым элементам строк в порядке убывания
  - 3) Входные параметры: `a`
  - 4) Выходные параметры: `a`
- Блок-схема:



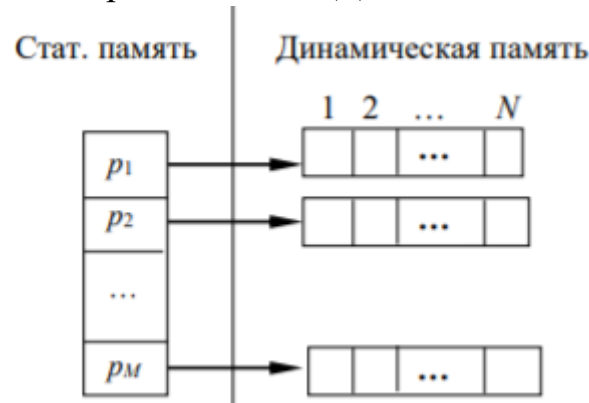
#### 4. Различные случаи алгоритма

##### 1) Число строк и число столбцов – константы

a) Описание структур данных:

```
const M = 4;  
      N = 4;  
type t_row = array[1..N] of integer;  
      t_p_row = ^t_row;  
      t_matr = array[1..M] of t_p_row;
```

b) Схема размещения в ДП:



c) Подпрограмма размещения матрицы в ДП:

```
procedure create_matr(var a: t_matr);  
var i: byte;  
begin  
  for i := 1 to M do  
    New(a[i]);  
end;
```

d) Обращение к строке и к элементу матрицы:

Поскольку матрица – массив указателей на строки:

Обращение к строке матрицы:  $a[i]^{\wedge}$

Обращение к элементу матрицы:  $a[i]^{\wedge}[j]$

e) Заголовок процедуры ввода матрицы:

```
procedure read_matr(a: t_matr)
```

f) Подпрограмма освобождения памяти:

```
procedure del_matr(var a: t_matr);  
var i: byte;  
begin  
  for i := 1 to M do  
    Dispose(a[i]);  
end;
```

g) Тестовые данные:

№	Вход	Выход
1	10 2 3 4 15 2 3 4 3 9 8 4 11 2 3 4	15 2 3 4 11 2 3 4 10 2 3 4 3 9 8 4
2	9 8 7 5 9 2 3 4 122 0 3 4 4 4 4 4	«Матрицу нельзя отсортировать в порядке убывания»

h) Скриншоты программы:

```
{ $CODEPAGE UTF-8 }

const M = 4;
      N = 4;

type t_row = array[1..N] of integer;
      t_p_row = ^t_row;
      t_matr = array[1..M] of t_p_row;

procedure create_matr(var a: t_matr);
var i: byte;
begin
  for i := 1 to M do
    New(a[i]);
  end;

procedure read_matr(a: t_matr);
var i, j: byte;
begin
  for i := 1 to M do
    for j := 1 to N do
      read(a[i]^j);
    end;
  end;
```

```

procedure write_matr(a: t_matr);
var i,j: byte;
begin
  for i := 1 to M do
    begin
      for j := 1 to N do
        write(a[i]^j, ' ');
      writeln();
    end;
  end;

```

```

function is_can_be_sorted(a: t_matr) : boolean;
var i,j: byte;
    res: boolean = true;
begin
  i := 1;

  while (i < M) and (res = true) do
    begin
      j := i + 1;

      while (j <= M) and (res = true) do
        begin
          if (a[i]^1 = a[j]^1) then
            res := false;

          j := j + 1;
        end;

        i := i + 1;
      end;

      is_can_be_sorted := res;
    end;

```

```

procedure sort(var a: t_matr);
var i,j: byte;
    key: t_p_row;
begin
    for i := 2 to M do
        begin
            key := a[i];
            j := i;
            while (j > 1) and (a[j-1]^1 < key^1) do
                begin
                    a[j] := a[j-1];
                    j := j - 1;
                end;
            a[j] := key;
        end;
    end;

procedure del_matr(var a: t_matr);
var i: byte;
begin
    for i := 1 to M do
        Dispose(a[i]);
    end;

```

```

var a: t_matr;
    l: integer;

begin
create_matr(a);

writeln('Ввод матрицы: ');
read_matr(a);

if (is_can_be_sorted(a)) then
begin
    sort(a);

    writeln('Вывод отсортированной матрицы: ');
    write_matr(a);
end
else
    writeln('Матрица не может быть отсортирована в порядке убывания');

del_matr(a);

read(l);
end.

```

```

Ввод матрицы:
10 2 3 4
15 2 3 4
3 9 8 4
11 2 3 4
Вывод отсортированной матрицы:
15 2 3 4
11 2 3 4
10 2 3 4
3 9 8 4

```

```

Ввод матрицы:
9 8 7 5
9 2 3 4
122 0 3 4
4 4 4 4
Матрица не может быть отсортирована в порядке убывания

```

**2) Число строк - константа, число столбцов – исходное данные**

a) Описание структур данных

```
const MAX = 20000;
```

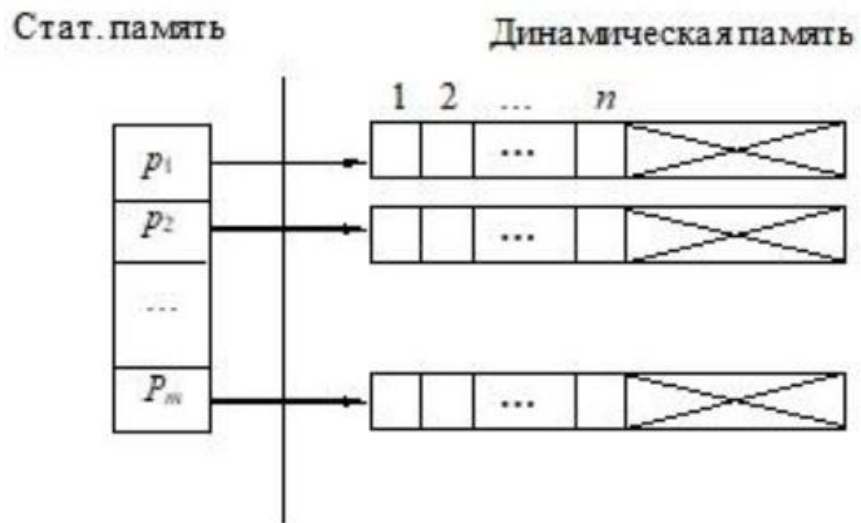
```
M = 4;
```

```
type t_max_row = array[1..MAX div  
  SizeOf(integer)] of integer;
```

```
t_p_max_row = ^t_max_row;
```

```
t_matr = array[1..M] of t_p_max_row;
```

b) Схема размещения матрицы в ДП:



c) Подпрограмма размещения матрицы в ДП:

```
procedure create_matr(var a: t_matr; n: byte);  
var i: byte;  
begin  
  for i := 1 to M do  
    GetMem(a[i], n*SizeOf(integer));  
end;
```

d) Обращение к строке и элементу матрицы:

Поскольку матрица – массив указателей на строки:

Обращение к строке матрицы:  $a[i]^{\wedge}$

Обращение к элементу матрицы:  $a[i]^{\wedge}[j]$

e) Заголовок программы ввода матрицы:

```
procedure read_matr(a: t_matr; n: byte);
```

f) Подпрограмма освобождения памяти:

```
procedure del_matr(var a: t_matr; n: byte);  
var i: byte;  
begin  
  for i := 1 to M do  
    FreeMem(a[i], n*SizeOf(integer));
```

end;

g) Тестовые данные:

№	Вход	Выход
1	$n = 3$ 4 5 2 10 13 2 2 3 1 1 0 1	10 13 2 4 5 2 2 3 1 1 0 1
2	$n = 2$ 1 2 3 4 5 6 3 2	«Матрица не может быть отсортирована в порядке убывания»

h) Скриншоты:

```
{ $CODEPAGE UTF-8 }

const MAX = 20000;
      M = 4;

type t_max_row = array[1..MAX div SizeOf(integer)] of integer;
      t_p_max_row = ^t_max_row;
      t_matr = array[1..M] of t_p_max_row;

procedure create_matr(var a: t_matr; n: byte);
var i: byte;
begin
  for i := 1 to M do
    GetMem(a[i], n * SizeOf(integer));
  end;

procedure read_matr(a: t_matr; n: byte);
var i, j: byte;
begin
  for i := 1 to M do
    for j := 1 to n do
      read(a[i]^j);
    end;
  end;
```



```

procedure write_matr(a: t_matr; n: byte);
var i,j: byte;
begin
  for i := 1 to M do
    begin
      for j := 1 to n do
        write(a[i]^j, ' ');
      writeln();
    end;
  end;
end;

```

```

function is_can_be_sorted(a: t_matr) : boolean;
var i,j: byte;
    res: boolean = true;
begin
  i := 1;

  while (i < M) and (res = true) do
    begin
      j := i + 1;

      while (j <= M) and (res = true) do
        begin
          if (a[i]^1 = a[j]^1) then
            res := false;
          j := j + 1;
        end;
      i := i + 1;
    end;

    is_can_be_sorted := res;
  end;
end;

```

```

procedure sort(var a: t_matr);
var i,j: byte;
    key: t_p_max_row;
begin
    for i := 2 to M do
        begin
            key := a[i];
            j := i;
            while (j > 1) and (a[j-1]^[1] < key^[1]) do
                begin
                    a[j] := a[j-1];
                    j := j - 1;
                end;
            a[j] := key;
        end;
    end;

procedure del_matr(var a: t_matr; n: byte);
var i: byte;
begin
    for i := 1 to M do
        FreeMem(a[i],n * SizeOf(integer));
    end;

```

```

var a: t_matr;
    l,n: integer;

begin
writeln('Введите кол-во элементов в строке');
readln(n);

create_matr(a,n);

writeln('Ввод матрицы: ');
read_matr(a,n);

if (is_can_be_sorted(a)) then
begin
    sort(a);

    writeln('Вывод отсортированной матрицы: ');
    write_matr(a,n);
end
else
    writeln('Матрица не может быть отсортирована в порядке убывания');

del_matr(a,n);

```

```

Введите кол-во элементов в строке
3
Ввод матрицы:
4 5 2
10 13 2
2 3 1
1 0 1
Вывод отсортированной матрицы:
10 13 2
4 5 2
2 3 1
1 0 1

```

```

Введите кол-во элементов в строке
2
Ввод матрицы:
1 2
3 4
5 6
3 2
Матрица не может быть отсортирована в порядке убывания

```

**3) Число строк – исходное данные, число столбцов – константа:**

a) Описание структур данных:

```
const MAX = 20000;
```

```
N = 4;
```

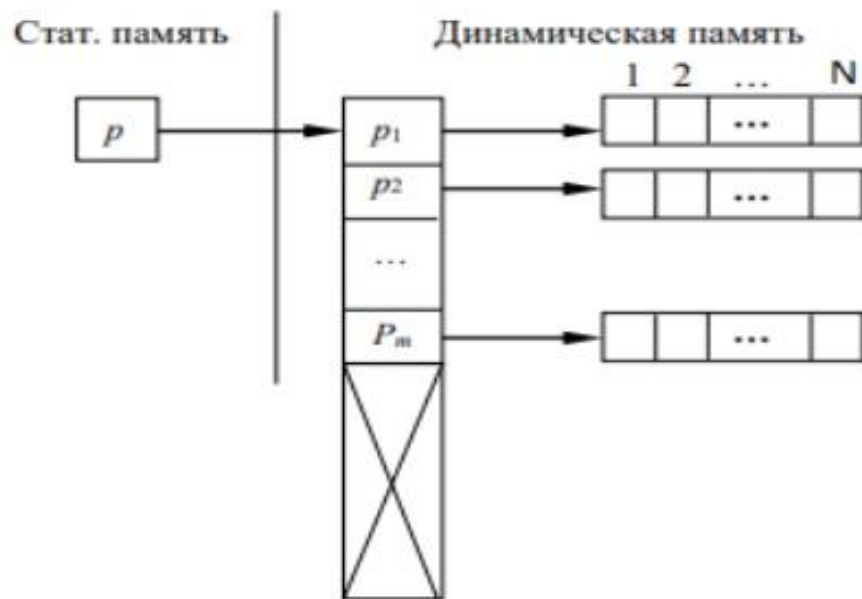
```
type t_row = array[1..N] of integer;
```

```
t_p_row = ^t_row;
```

```
t_max_matr = array[1..MAX div  
SizeOf(pointer)] of t_p_row
```

```
t_p_max_matr = ^t_max_matr;
```

b) Схема размещения матрицы в ДП:



c) Подпрограмма размещения матрицы в ДП:

```
procedure create_matr(var pa: t_p_max_matr; m:  
byte);
```

```
var i: byte;
```

```
begin
```

```
    GetMem(pa, m*SizeOf(pointer));
```

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

```
        New(pa^[i]);
```

```
    end;
```

d) Обращение к строке и к элементу матрицы:

Поскольку матрица – указатель на массив указателей:

pa^[i]^ - обращение к строке

pa^[i]^[j] – обращение к элементу

e) Заголовок процедуры ввода матрицы:

```
procedure read_matr(pa: t_p_max_matr; m:  
byte);
```

f) Подпрограмма освобождения памяти:

```

procedure del_matr(var pa: t_p_max_matr; m:
byte);
var i: byte;
begin
    for i := 1 to m do
        Dispose(pa^[i]);
        FreeMem(pa, m*SizeOf(pointer));
    end;

```

g) Тестовые данные:

№	Вход	Выход
1	$m = 2$ 1 2 3 4 5 6 7 8	5 6 7 8 1 2 3 4
2	$m = 3$ 8 9 7 7 1 1 0 0 8 2 3 10	«Матрица не может быть отсортирована в порядке убывания»

h) Скриншоты:

```

{$CODEPAGE UTF-8}

const MAX = 20000;
      N = 4;

type t_row = array[1..MAX div SizeOf(integer)] of integer;
      t_p_row = ^t_row;
      t_max_matr = array[1..MAX div SizeOf(pointer)] of t_row;
      t_p_max_matr = ^t_max_matr;

procedure create_matr(var pa: t_p_max_matr; m: byte);
var i: byte;
begin
    GetMem(pa, m*SizeOf(pointer));
    for i := 1 to m do
        New(pa^[i]);
    end;

```

```
procedure read_matr(pa: t_p_max_matr; m: byte);
var i,j: byte;
begin
    for i := 1 to m do
        for j := 1 to N do
            read(pa^[i]^[j]);
        end;
    end;

procedure write_matr(pa: t_p_max_matr; m: byte);
var i,j: byte;
begin
    for i := 1 to m do
        begin
            for j := 1 to N do
                write(pa^[i]^[j], ' ');
            end;
            writeln();
        end;
    end;
```

```

function is_can_be_sorted(pa: t_p_max_matr; m: byte) : boolean;
var i,j: byte;
    res: boolean = true;
begin
    i := 1;

    while (i < m) and (res = true) do
        begin
            j := i + 1;

            while (j <= m) and (res = true) do
                begin
                    if (pa^[i]^[1] = pa^[j]^[1]) then
                        res := false;
                    j := j + 1;
                end;
            i := i + 1;
        end;
    is_can_be_sorted := res;
end;

```

```

procedure sort(var pa: t_p_max_matr; m: byte);
var i,j: byte;
    key: t_p_row;
begin
    for i := 2 to m do
        begin
            key := pa^[i];
            j := i;
            while (j > 1) and (pa^[j-1]^[1] < key^[1]) do
                begin
                    pa^[j] := pa^[j-1];
                    j := j - 1;
                end;
            pa^[j] := key;
        end;
    end;
end;

```

```

procedure del_matr(var pa: t_p_max_matr; m: byte);
var i: byte;
begin
    for i := 1 to m do
        Dispose(pa^[i]);
        FreeMem(pa, m*SizeOf(pointer));
    end;

var pa: t_p_max_matr;
    l, m: integer;

```

```

begin
writeln('Введите кол-во строк');
readln(m);

create_matr(pa, m);

writeln('Ввод матрицы: ');
read_matr(pa, m);

if (is_can_be_sorted(pa, m)) then
begin
    sort(pa, m);

    writeln('Вывод отсортированной матрицы: ');
    write_matr(pa, m);
end
else
writeln('Матрица не может быть отсортирована в порядке убывания');

del_matr(pa, m);

read(l);
end.

```

```

Введите кол-во строк
2
Ввод матрицы:
1 2 3 4
5 6 7 8
Вывод отсортированной матрицы:
5 6 7 8
1 2 3 4

```



```

Введите кол-во строк
3
Ввод матрицы:
8 9 7 7
1 1 0 0
8 2 3 10
Матрица не может быть отсортирована в порядке убывания

```

4) Число строк и число столбцов – исходные данные:

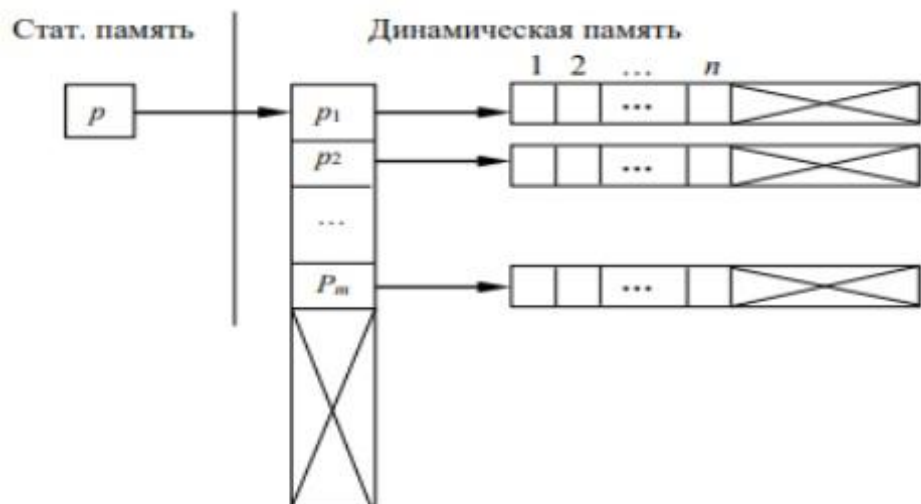
a) Описание структур данных:

```

const MAX = 20000;
type t_max_row = array[1..MAX div
  SizeOf(integer)] of integer;
      t_p_max_row = ^t_max_row;
      t_max_matr = array[1..MAX div
  SizeOf(pointer)] of t_p_max_row;
      t_p_max_matr = ^t_max_matr;

```

b) Схема размещения матрицы в ДП:



c) Подпрограмма размещения матрицы в ДП:

```

procedure create_matr(var pa: t_p_max_matr;
  m,n: byte);
var i: byte;
begin
  GetMem(pa,m*SizeOf(pointer));
  for i := 1 to m do
    GetMem(pa^[i],n*SizeOf(integer));
  end;

```

d) Обращение к строке и к элементу матрицы:

Поскольку матрица – указатель на массив указателей:  
 $pa^i$  - обращение к строке

pa^[i]^[j] – обращение к элементу

e) Заголовок процедуры ввода матрицы:

```
procedure read_matr(pa: t_p_max_matr; m,n:
byte);
```

f) Подпрограмма освобождения памяти:

```
procedure del_matr(var pa: t_p_max_matr; m,n:
byte);
var i: byte;
begin
  for i := 1 to m do
    FreeMem(pa^[i],n*SizeOf(integer));
    FreeMem(pa,m*SizeOf(pointer));
  end;
```

g) Тестовые данные:

№	Вход	Выход
1	$m = 3, n = 2$ 4 5 10 9 1 0	10 9 4 5 1 0
2	$m = 5, n = 3$ 4 5 6 13 20 3 100 2 3 5 6 7 13 9 9	«Матрица не может быть отсортирована в порядке убывания»

h) Скриншоты:

```
{ $CODEPAGE UTF-8 }
```

```
const MAX = 20000;
```

```
type t_max_row = array[1..MAX div SizeOf(integer)] of integer;  
|   t_p_max_row = ^t_max_row;  
|   t_max_matr = array[1..MAX div SizeOf(pointer)] of t_p_max_row;  
|   t_p_max_matr = ^t_max_matr;
```

```
procedure create_matr(var pa: t_p_max_matr; m,n: byte);  
var i: byte;  
begin  
    GetMem(pa,m*SizeOf(pointer));  
    for i := 1 to m do  
        GetMem(pa^[i],n*SizeOf(integer));  
    end;
```

```
procedure read_matr(pa: t_p_max_matr; m,n: byte);  
var i,j: byte;  
begin  
    for i := 1 to m do  
        for j := 1 to n do  
            read(pa^[i]^[j]);  
        end;  
    end;
```

```
procedure write_matr(pa: t_p_max_matr; m,n: byte);  
var i,j: byte;  
begin  
    for i := 1 to m do  
        begin  
            for j := 1 to n do  
                write(pa^[i]^[j], ' ');  
            writeln();  
        end;  
    end;
```

```

function is_can_be_sorted(pa: t_p_max_matr; m: byte) : boolean
var i,j: byte;
    res: boolean = true;
begin
    i := 1;

    while (i < m) and (res = true) do
        begin
            j := i + 1;

            while (j <= m) and (res = true) do
                begin
                    if (pa^[i]^[1] = pa^[j]^[1]) then
                        res := false;

                    j := j + 1;
                end;

                i := i + 1;
            end;

            is_can_be_sorted := res;
        end;
    end;
end;

```

```

procedure sort(var pa: t_p_max_matr; m: byte);
var i,j: byte;
    key: t_p_max_row;
begin
    for i := 2 to m do
        begin
            key := pa^[i];
            j := i;
            while (j > 1) and (pa^[j-1]^[1] < key^[1]) do
                begin
                    pa^[j] := pa^[j-1];
                    j := j - 1;
                end;
            pa^[j] := key;
        end;
    end;

procedure del_matr(var pa: t_p_max_matr; m,n: byte);
var i: byte;
begin
    for i := 1 to m do
        FreeMem(pa^[i],n*SizeOf(integer));
        FreeMem(pa,m*SizeOf(pointer));
    end;

```

```

var a: t_p_max_matr;
    l,m,n: integer;

begin
writeln('Введите кол-во строк');
readln(m);

writeln('Введите кол-во элементов в строке');
readln(n);

create_matr(a,m,n);

writeln('Ввод матрицы: ');
read_matr(a,m,n);

if (is_can_be_sorted(a,m)) then
begin
    sort(a,m);

    writeln('Вывод отсортированной матрицы: ');
    write_matr(a,m,n);
end
else
writeln('Матрица не может быть отсортирована в порядке уб');

del_matr(a,m,n);

read(l);
end.

```

```

Введите кол-во строк
3
Введите кол-во элементов в строке
2
Ввод матрицы:
4 5
10 9
1 0
Вывод отсортированной матрицы:
10 9
4 5
1 0

```

Введите кол-во строк

5

Введите кол-во элементов в строке

3

Ввод матрицы:

4 5 6

13 20 3

100 2 3

5 6 7

13 9 9

Матрица не может быть отсортирована в порядке убывания