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: a) число строк и число столбцов − константы; b) число строк − константа, а число столбцов − исходное данное; c) число строк − исходное данное, число столбцов − константа; d) число строк и число столбцов − исходные данные.
4. Ввод, вывод и обработку матриц опишите отдельными подпрограммами. Для случаев a) − d), 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 `MxN`, `pa` – адрес матрицы
   - Ввод матрицы `MxN` по адресу `pa`
   - Вывод матрицы `MxN` по адресу `pa`
   - Упорядочивание строк матрицы по первому элементу каждой строки в порядке убывания
   - Освобождение памяти занимаемой матрицей размера `MxN` по адресу `pa`

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

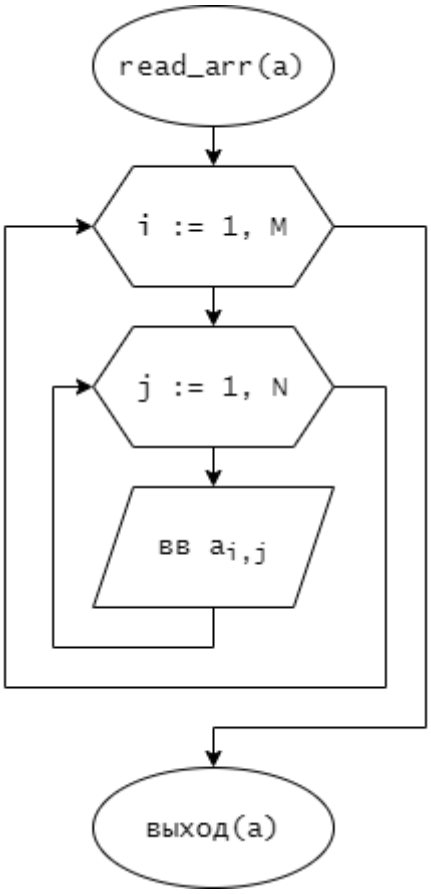
2. **Блок-схема алгоритма в укрупнённых блоках:**

3. Описание подпрограмм:

Спецификация процедуры `read_arr`

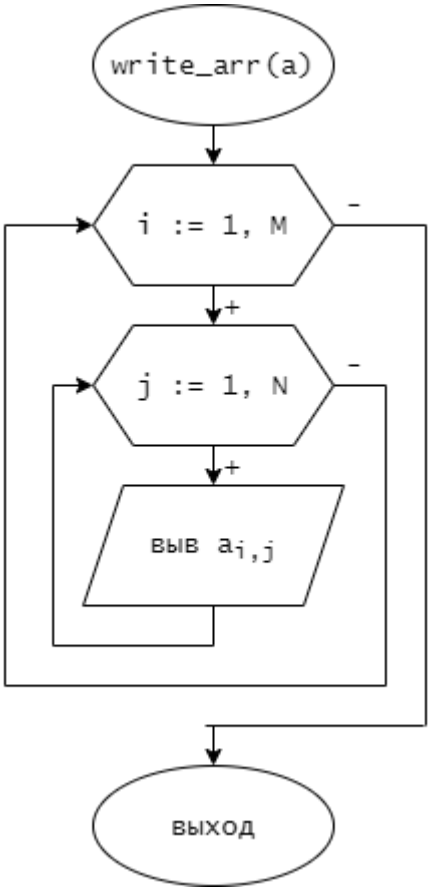
1) Заголовок: `procedure read_arr(var a: t_matr)`
2) Назначение: ввод матрицы `a` размера `MxN`
3) Входные параметры: `нет`
4) Выходные параметры: `a`

   Блок-схема:



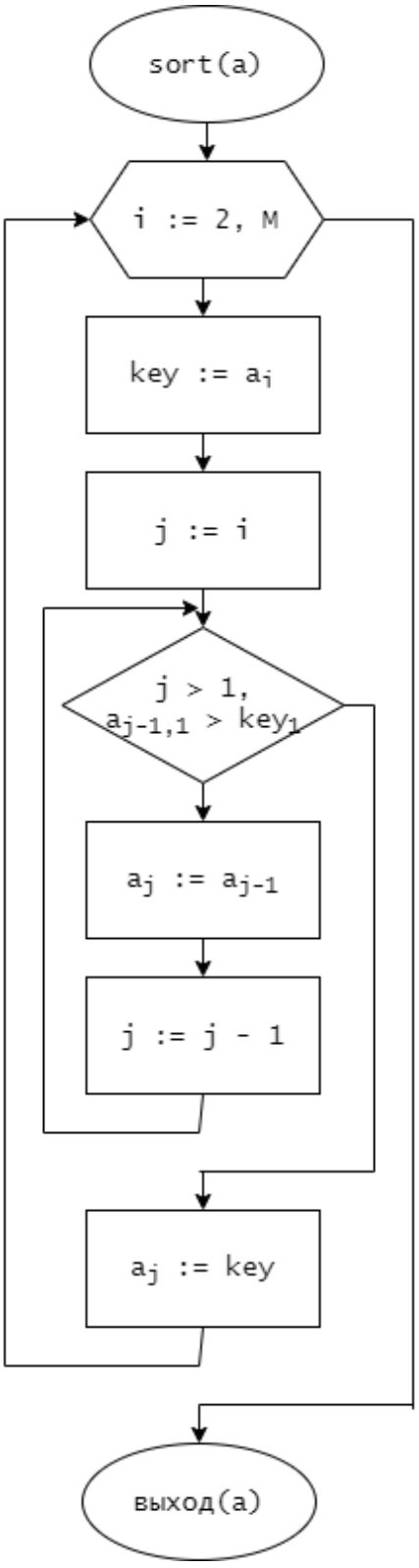
Спецификация процедуры `write_arr`

1) Заголовок: `procedure write_arr(a: t_matr)`
2) Назначение: вывод матрицы `a` размера `MxN`
3) Входные параметры: `a`
4) Выходные параметры: `нет`
   Блок-схема:



Спецификация процедуры `sort`

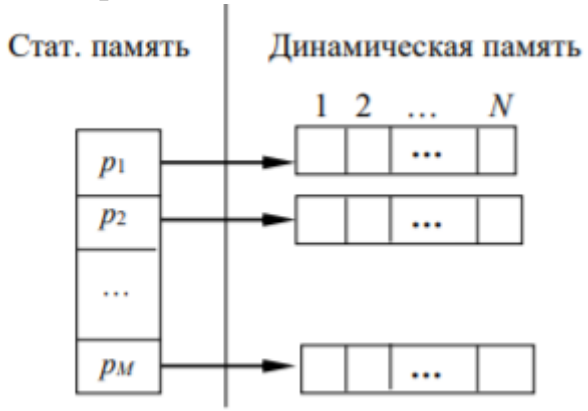
1) Заголовок: `procedure sort(var a: t_matr)`
2) Назначение: сортировка матрицы `a` по первым элементам строк в порядке убывания
3) Входные параметры: `a`
4) Выходные параметры: `a`
   Блок-схема:



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]^`
        Обращение к элементу матрицы: `a[i]^[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 10 2 3 4 122 0 3 4 4 4 4 4	122 0 3 4 10 2 3 4 9 8 7 5 4 4 4 4

h) Скриншоты программы:

```
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;

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],' ');
```

```
            write(a[i]^[j],' ');
        writeln();
        end;
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);

sort(a);

writeln('Вывод отсортированной матрицы: ');
write_matr(a);

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
```

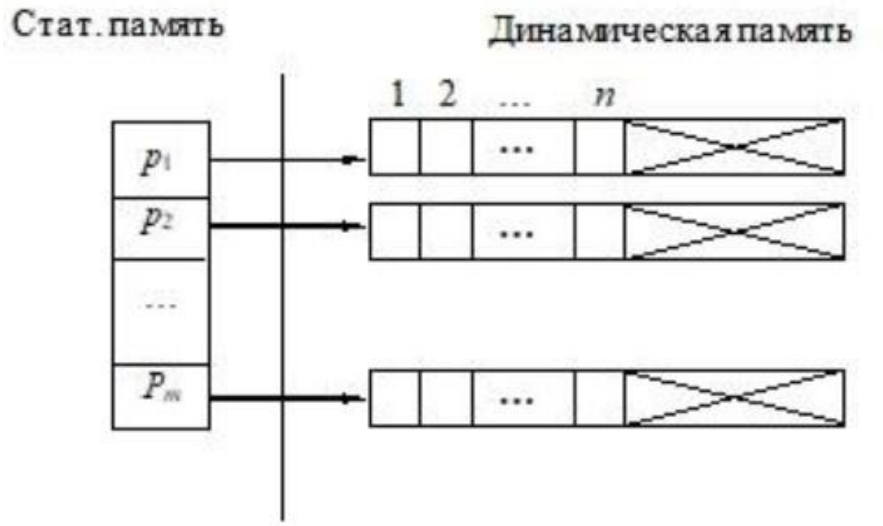
```
Ввод матрицы:
9 8 7 5
10 2 3 4
122 0 3 4
4 4 4 4
Вывод отсортированной матрицы:
122 0 3 4
10 2 3 4
9 8 7 5
4 4 4 4
```

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

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]^`
Обращение к элементу матрицы: `a[i]^[j]`

e) Заголовок программы вводы 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 2 2	5 6 3 4 2 2 1 2

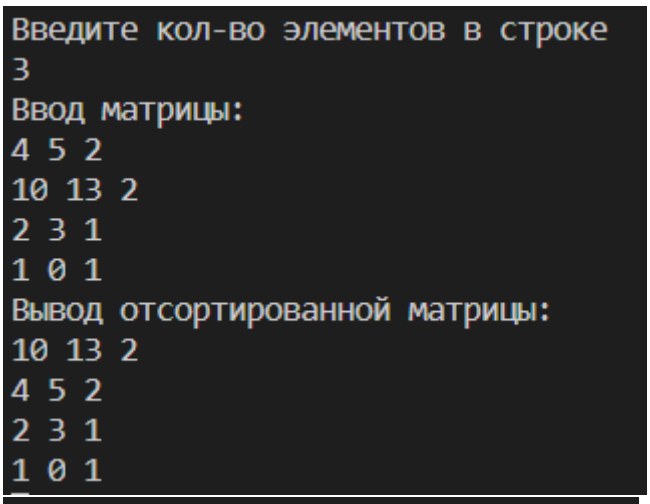
h) Скриншоты:

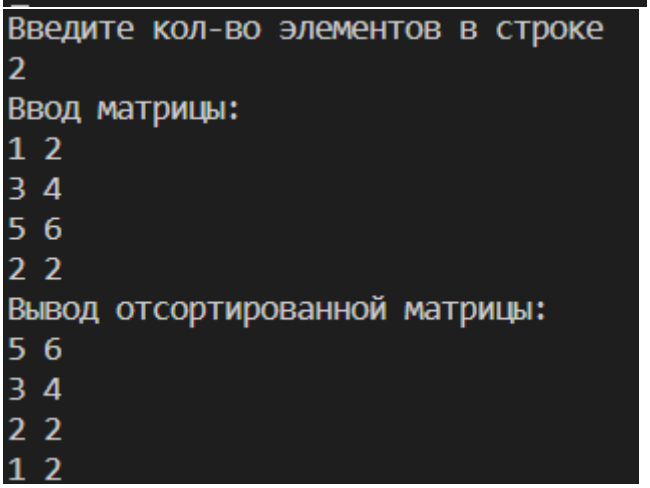
```
{$CODEPAGE UTF-8}

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;
```

```
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;

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);
```

```
sort(a);

writeln('Вывод отсортированной матрицы: ');
write_matr(a,n);

del_matr(a,n);

read(l);
end.
```

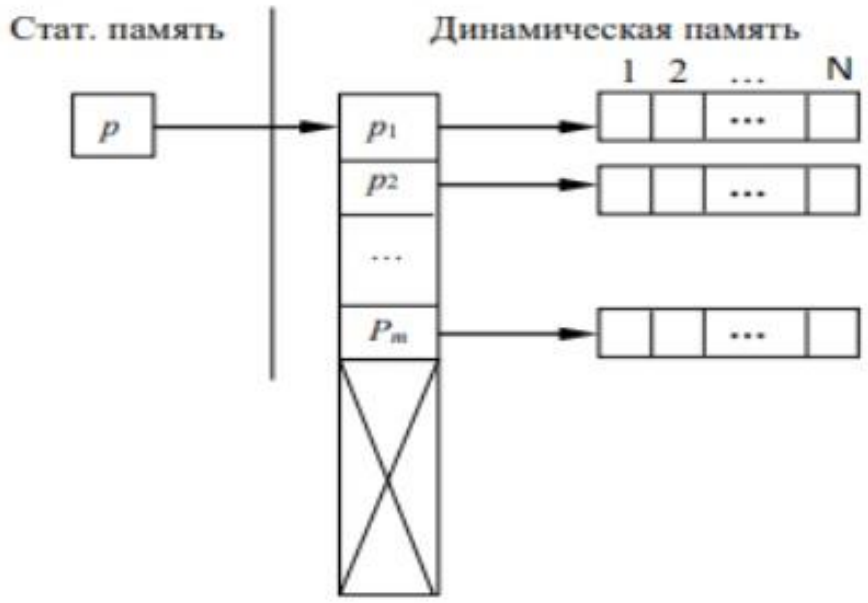




3) **Число строк – исходное данное, число столбцов – константа:**
   a) Описание структур данных:

```
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 0 2 3 10	8 9 7 7 1 1 0 0 0 2 3 10

h) Скриншоты:

```
{$CODEPAGE UTF-8}

const MAX = 20000;
      N = 4;

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

```
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],' ');
    writeln();
    end;
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;

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);

sort(pa,m);

writeln('Вывод отсортированной матрицы: ');
write_matr(pa,m);

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
```

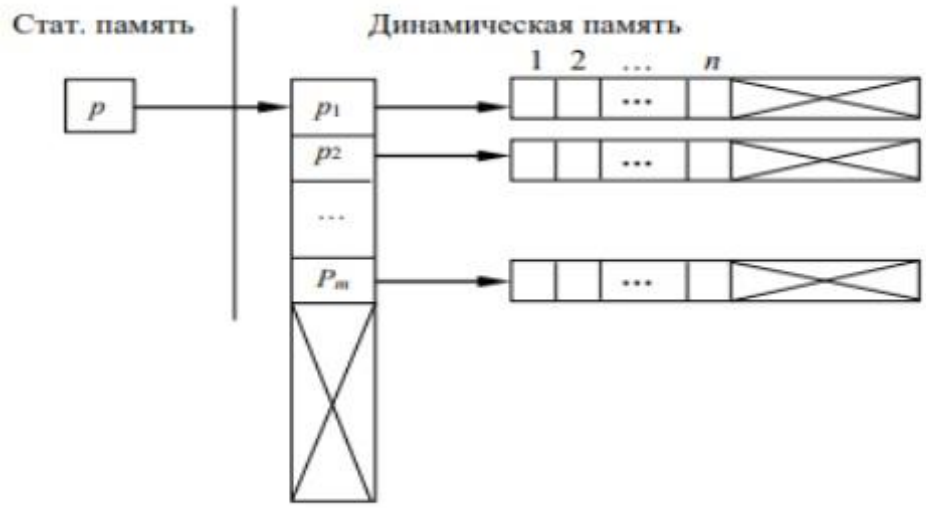
```
Введите кол-во строк
3
Ввод матрицы:
8 9 7 7
1 1 0 0
0 2 3 10
Вывод отсортированной матрицы:
8 9 7 7
1 1 0 0
0 2 3 10
```

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 9 9 9	100 2 3 13 20 3 9 9 9 5 6 7 4 5 6

h) Скриншоты:

```
{$CODEPAGE UTF-8}

const MAX = 20000;

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

```
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;

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;
```

```
        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);

sort(a,m);

writeln('Вывод отсортированной матрицы: ');
write_matr(a,m,n);

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
9 9 9
Вывод отсортированной матрицы:
100 2 3
13 20 3
9 9 9
5 6 7
4 5 6
```