Лабораторная работа №8
Указатели и динамические массивы

Цель работы: получение навыков работы с указателями и динамическими массивами.

Указатели

Указатель *(pointer)* – переменная, диапазон значений которой состоит из адресов ячеек памяти или специального значения – нулевого адреса. Если переменная объявлена как указатель, то она содержит адрес памяти, по которому может находиться скалярная величина любого типа. При объявлении переменной типа указатель, необходимо определить тип объекта данных, адрес которых будет содержать переменная, и имя указателя с предшествующей звездочкой. Формат объявления указателя:

```
type* ptrname;
```

Нулевой указатель − это указатель, хранящий специальное значение (**0** или **NULL**), используемое для того, чтобы показать, что данная переменная-указатель не ссылается (не указывает) ни на какой объект.

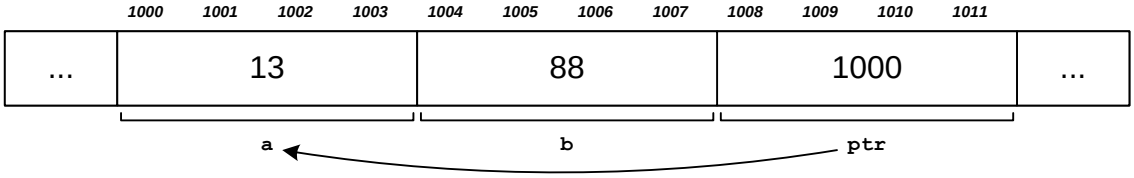
Основные действия над указателями

Для того чтобы присвоить указателю адрес некоторой переменной нужно использовать унарную операцию **&**. С использованием этой операции мы уже раньше встречались в функции **scanf()**. Присвоение указателю адреса переменной можно осуществлять как в месте его объявления, так и отдельным оператором.

Следующий пример демонстрирует объявление двух целочисленных переменных типа **int** и указателя на тип **int**, которому присваивается адрес переменной **a**.

```
int a = 13;
int b = 88;
int* ptr = &a;
```

На рисунке показано представление этих переменных в памяти компьютера. Адреса переменных и их взаимное расположение показано условно и при выполнении на разных компьютерах будут отличаться.



Следующий оператор выведет на экран значения переменных **a** и **b**, значение указателя **ptr** (адрес в памяти) и значение, на которое указывает **ptr**. Для вывода значения, на которое указывает указатель, используется унарная операция разыменования указателя ***.

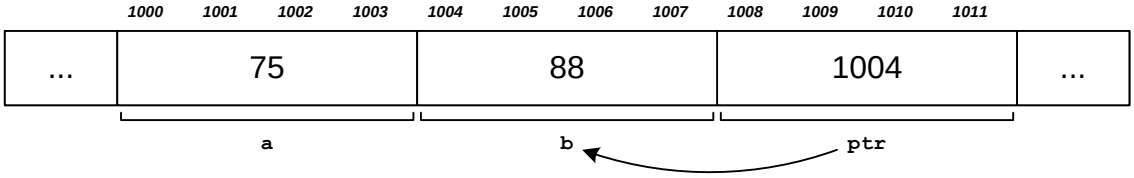
```
printf("a = %d, b = %d, ptr = %d, *ptr = %d\n",a, b, ptr, *ptr);
```

Так как указатель **ptr** указывает на переменную **a**, то значения, выводимые переменной **a** и ***ptr**, должны совпадать (см. рисунок **Ошибка! Источник ссылки не найден.**).

Операцию разыменования указателя можно использовать и в левой части оператора присваивания. В следующем примере переменной, на которую указывает указатель **ptr**, присваивается значение разности переменных **b** и **a** (88–13=75).

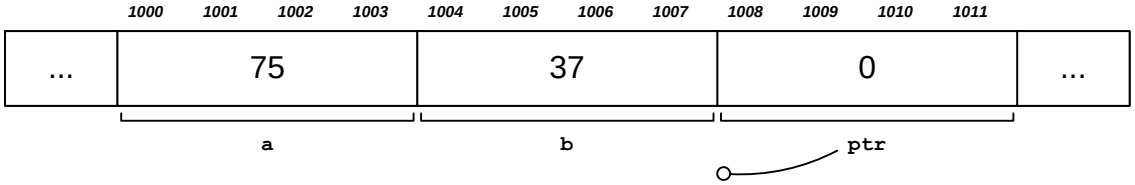
```
*ptr = b - a;
ptr = &b;
```

Затем указателю присваивается адрес переменной **b**. Представление переменных в памяти компьютера после этих действий показано на рисунке.



Присвоим переменной, на которую указывает указатель **ptr**, значение 37, а самому указателю нулевой адрес с помощью специальной константы **NULL**. Представление переменных в памяти компьютера после этих действий показано на рисунке.

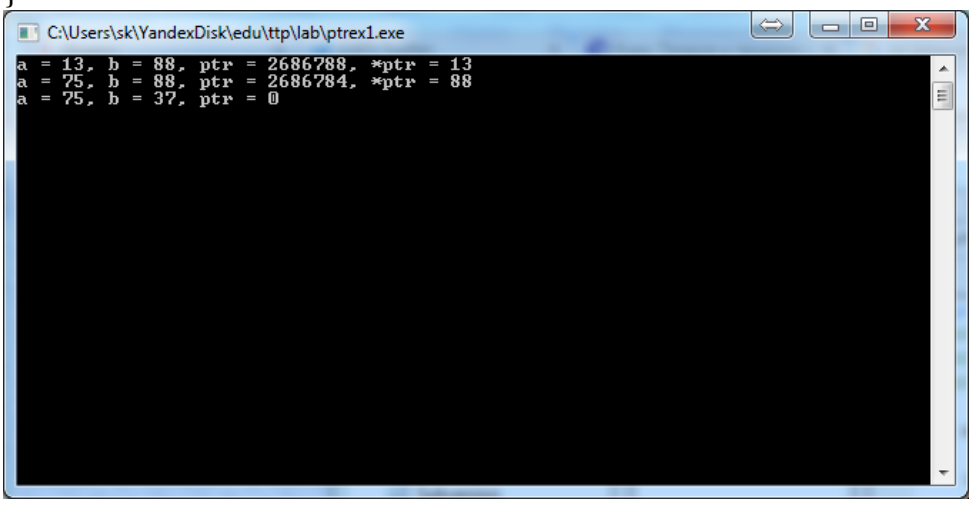
***ptr = 37;**
ptr = NULL;



Разыменование нулевого указателя приводит к неопределенному результату. Например, попытка вывести значение нулевого указателя, скорее всего, приведет к аварийному завершению программы:

```
ptr = NULL;
printf(“*ptr = %d\n”, *ptr);
```

```
/* ptrex1.c - пример использования указателей */
#include <stdio.h>
#include <locale.h>
#include <conio.h>
int main()
{    int a = 13;
   int b = 88;
   int* ptr = &a;
   setlocale(LC_ALL, "");
   printf("a = %d, b = %d, ptr = %d, *ptr = %d\n", a, b, ptr, *ptr);
   ptr = b - a;   ptr = &b;
   printf("a = %d, b = %d, ptr = %d, *ptr = %d\n", a, b, ptr, *ptr);
   *ptr = 37;    ptr = NULL;
   printf("a = %d, b = %d, ptr = %d\n", a, b, ptr);
    getch();
   return 0;
}
```



ДИНАМИЧЕСКИЕ МАССИВЫ

Динамические массивы могут быть созданы двумя способами : либо с помощью операций **new[ ]** из языка **C++**, либо с помощью функции **malloc( )** из библиотеки языка **C**, при этом нужно указать тип и количество элементов массива.

Пример:

```
int n = 100;
float *p = new float[n];
```

Создается указатель **p** на вещественный тип , в операционной памяти выделяется непрерывная область смежных ячеек памяти для размещения 100 элементов вещественного типа, и при этом адрес начальной ячейки записывается в указатель **p**

Аналогично функция **malloc(m)** из библиотеки языка **C** выделяет непрерывную область памяти, длиной **m** байтов, поэтому для создания динамического массива из предыдущего примера необходимо записать следующие операторы:

```
int n = 100; m = sizeof (float);
float *f = (float*) malloc(n*m);
```

В переменной **m** вычислено и записано количество байт, необходимых для размещения одной переменной вещественного типа. В аргументе функции **malloc ( )** указано общее количество байт: **n*m** для размещения **n=100** элементов вещественного типа.

Доступ к элементам динамического массива осуществляется точно так же как и к статическим, так как массив и указатель – одно и то же: **p[5]**.

Динамические массивы *не обнуляются* при создании.

Память, выделенная для динамического массива, после использования должна быть освобождена. Для первого способа (для операции **new[ ]**) с помощью оператора **delete[ ]**, для второго способа (для функции **malloc( ))** посредством функции **free( ).**

Пример:

```
delete[ ] p; //освобождение памяти для указателя p
free ( f ); // освобождение памяти для указателя f
```

Пример: Вычислить сумму элементов динамического массива.

```
#include<malloc.h>
#include<stdlib.h>
#include<time.h>
#include<iostream.h>
void main( )
{int *p;
int sum = 0, i, n;
cout<< ''Введите длину массива'';
cin >> n;
p = (int*) malloc( n*sizeof (int));/* выделение памяти для n элементов целого типа*/
for (i = 0; i < n; i ++ ) {
```

```
            p[i] = rand ()%10 – 5 ; sum+=p[i];
}
cout<< ''Sum='' << sum;
free(p); // освобождения выделенной памяти
}
```

ДВУМЕРНЫЕ ДИНАМИЧЕСКИЕ МАССИВЫ

Для создания динамического двумерного массива необходимо выполнить следующую последовательность действий:

1) **float **mass**; //описать указатель на указатель;

2) **mass = (float**) malloc(n*sizeof (float*))**; /* выделить память для одномерного массива указателей на будущие строки двумерного массива, состоящего из **n** элементов, где **n** – количество строк (см. рис. 1) */

3) **for (int i = 0; i < n; i ++)**

{mass [i]= (float*) malloc(m*sizeof(float));}/* в цикле для каждого элемента массива указателей выделяется память под каждую строку двумерного массива. Причем каждая строка будет содержать **m** элементов вещественного типа, где **m** – количество столбцов (см. рис. 1) */

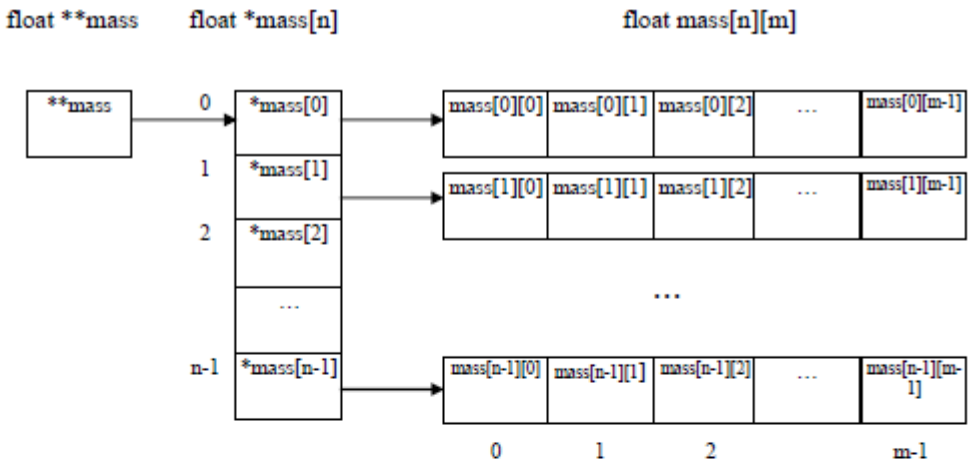


Рис. 1. « Выделение памяти для динамического двумерного массива»

Создание двумерного динамического массива с помощью средств языка С++ происходит аналогичным способом, только изменится синтаксис операторов:

```
float **mass;
mass = new float *[n];
for ( int i = 0; i < n; i ++)
{ mass[i] = new float [m]; }
```

Пример: Вычислить след матрицы, то есть произведение элементов главной диагонали матрицы с использованием двумерного динамического массива.

```
#include<stdlib.h>
#include<time.h>
#include<iostream.h>
void main ( )
{
int **matr;
int n;
cout << "Введите n";
cin >> n;
matr = (int*) malloc (n*sizeof(int*));
 for( int i = 0; i < n ; i ++) matr[i] = (int*) malloc (n*sizeof(int));
/* заполним матрицу случайными числами */
for( int i = 0 ; i < n ; i ++)
for ( int j = 0 ; j < n ; j ++)
matr [i][j] = rand()%10;
int sled = 1;
for(i=0;i<n;i++)
{sled* = matr [i][i]; }
cout << "sled =" << sled;
}
```

Операции с указателями

В таблице собраны примеры основных операций и функций, применяемых в языке Си с указателями. В примерах подразумевается, что указатели типа **int***, но эти операции применимы и для других типов данных.

Пример операции	**Описание**
ptr = &var;	Присвоить указателю **ptr** адрес переменной **var**
***ptr = 5;**	Разыменование указателя и присвоение значения **5**
a = *ptr;	Разыменование указателя и присвоение значения переменной **a**
printf("%d", *ptr);	Разыменование указателя и вывод его значения на экран.
arr = (int*)malloc(5 * sizof(int));	Динамическое выделение блока памяти на **5** элементов типа **int** и присвоение его адреса указателю **arr**.
ptr++;	Увеличение значения указателя на размер его типа. Для динамических массивов — переход указателя к следующему элементу.
ptr--;	Уменьшение значения указателя на размер его типа. Для динамических массивов — переход указателя к предыдущему элементу.
ptr2 = ptr1 + 3;	Присваивает указателю **ptr2** значение указателя **ptr1**, увеличенное на **3** размера его типа, т.е. для типа **int** на 12 байт.
ptr2 = ptr1 – 7;	Присваивает указателю **ptr2** значение указателя **ptr1**, уменьшенное на **7** размеров его типа, т.е. для типа **int** на 28 байт.
n = ptr1 – ptr2;	Вычисляет разницу между значениями указателей в байтах и присваивает это значение переменной **n**.

	Операция сложения для двух указателей смысла не имеет.
free(arr);	Освобождение ранее выделенной памяти.

Пример. Написать программу, осуществляющую динамическое выделение памяти для массива произвольного размера, заполнение его псевдослучайными числами от 0 до 999 и сортировку по возрастанию.

В начале программы осуществляется ввод при помощи функции **scanf()** необходимого количества элементов для динамического массива в целочисленную переменную **size**.

Затем при помощи функции **malloc()** осуществляется выделение памяти под динамический массив. Размер в байтах задается с использованием операции **sizeof**. Результат, возвращаемый функцией **malloc()**, приводится к типу указателя на **int** и присваивается переменной **arr**.

Далее производится инициализация генератора псевдослучайных чисел вызовом функции **srand()**. В качестве начального значения используется текущее время **time()**.

Заполнение динамического массива псевдослучайными целыми числами из диапазона от 0 до 999 осуществляется в цикле **while**. До цикла указателю **ptr** присваивается адрес начала динамического массива **arr**. Цикл выполняется пока значение указателя **ptr** не выйдет за пределы массива. В цикле текущему элементу массива, на который указывает **ptr**, присваивается псевдослучайное число, а сам указатель перемещается на следующий элемент с помощью операции **++**.

В следующем цикле **while** таким же образом значения элементов динамического массива выводятся на экран.

Сортировка массива осуществляется, так же как и в предыдущей лабораторной работе. Внешний цикл реализован при помощи оператора **for**, а внутренний при помощи оператора **while**. Обратите внимание, как при помощи указателя осуществляется доступ к значению следующего элемента массива с помощью операции **+** и разыменования *****: ***(ptr+1)**.

После сортировки значения массива выводятся на экран в цикле **while**. А затем осуществляется освобождение памяти при помощи функции **free()**.

```
#include <stdio.h>  /* printf, scanf */
#include <locale.h> /* setlocale */
#include <conio.h>  /* getch */
#include <stdlib.h> /* malloc, free, srand, rand */
#include <time.h>   /* time */

int main()
{
   int size;     /* размер динамического массива */
   int* arr;     /* указатель на начало массива */
   int* ptr;     /* указатель для работы с массивом */
   int i;       /* вспомогательная переменная */
   int tmp;      /* для перестановки элементов массива */

   setlocale(LC_ALL, "");

   /* запрос размера для динамического массива */
   printf("Введите размер массива: ");
   scanf("%d", &size);

   /* выделение памяти под массив */
   arr = (int*)malloc(size * sizeof(int));

   /* инициализация генератора случайных чисел */
```

```
srand(time(NULL));

/* заполнение массива случайными числами от 0 до 999 */
ptr = arr;  /* установка указателя ptr в начало массива */
while (ptr < arr + size)
   *ptr++ = rand() % 1000;

/* вывод исходного массива */
ptr = arr;
printf("Исходный массив:\n");
while (ptr < arr + size)
   printf("%5d", *ptr++);

/* сортировка */
for (i = 0; i < size; i++) {
   ptr = arr;
   while (ptr < arr + size - 1) {
      // если текущий элемент меньше следующего
      if (*ptr > *(ptr + 1)) {
         // перестановка местами
         // текущего и следующего элементов массива
         tmp = *ptr;
         *ptr = *(ptr + 1);
         *(ptr + 1) = tmp;
      }
      ptr++;       }    }

/* вывод результатов */
ptr = arr;
printf("\n\nОтсортированный массив:\n");
while (ptr < arr + size)
   printf("%5d", *ptr++);

/* освобождение памяти */
free(arr);

getch();
return 0;}
```

Задания для самостоятельной работы

Задание 1. Указатели и адреса.

1. Ввести значение 2-х целых переменных *a* и b. Направить два указателя на эти переменные. С помощью указателя увеличить значение переменной *a* в 2 раза. Затем поменять местами значения переменных a и b через их указатели.
2. Ввести значение 2-х целых переменных *a* и b. Направить два указателя на эти переменные. С помощью указателя увеличить значение переменной a в 2 раза если a>b иначе b уменьшить в 2 раза
3. Ввести значение 2-х вещественных переменных a и b. Направить два указателя на эти переменные. С помощью указателя увеличить значение переменной a в 3 раза. Затем поменять местами значения переменных a и b через их указатели.
4. Ввести значение 2-х вещественных переменных a и b. Направить два указателя на эти переменные. Если a>b, то с помощью указателя увеличить значение переменной a на 3 и b уменьшить в 3 раза, в противном случае a уменьшить в 2 раза и b увеличить на 3.
5. Ввести значение 2-х символьных переменных a и b. Направить два указателя на эти переменные. С помощью указателя изменить значение переменной *a* . Затем поменять местами значения переменных a и b через их указатели.
6. Ввести значение 2-х целых переменных a и b. Направить два указателя на эти переменные. Большее из них с помощью указателя увеличить в 5 раз и меньшее уменьшить на 5.
7. Ввести значение 3-х целых переменных a и b и c. Направить указатели на эти переменные. С помощью указателя увеличить значение переменной a в 2 раза. Затем поменять местами значения переменных c и b через их указатели.
8. Ввести значение 3-х вещественных переменных a и b и c. Направить указатели на эти переменные. С помощью указателя увеличить значение переменной c в 3 раза. Затем поменять местами значения переменных a и c через их указатели.
9. Ввести значение 2-х вещественных переменных a и b. Направить два указателя на эти переменные. Большее из них с помощью указателя увеличить на7 и меньшее уменьшить на 3.
10. Ввести значение 2-х символьных переменных a и b. Направить два указателя на эти переменные. Затем поменять местами значения переменных a и b через их указатели.
11. Ввести значение 2-х целых переменных a и b. Направить два указателя на эти переменные. Затем поменять местами значения переменных a и b через их указатели.
12. Ввести значение 2-х вещественных переменных a и b. Направить два указателя на эти переменные. Затем поменять местами значения переменных a и b через их указатели.
13. Ввести значение 2-х целых переменных a и b. Направить два указателя на эти переменные. С помощью указателя увеличить значение переменной a в 2 раза, a b уменьшить в 2 раза
14. Ввести значение 2-х вещественных переменных a и b. Направить два указателя на эти переменные. С помощью указателя увеличить значение переменной a в 3 раза , a b уменьшить в 3 раза
15. Ввести значение 2-х вещественных переменных a и b. Направить два указателя на эти переменные. С помощью указателя увеличить значение переменной a в 3 раза , a b уменьшить в 3 раза

Задание 2. Решить задачи с использованием указателей, динамической памяти и функций. В виде функций оформить ввод, вывод и обработку массива согласно варианта. В main должно быть только объявление массива, выделение динамической памяти и вызов объявленных функций.

1. Создать динамические массивы, используя указатели. Задан одномерный массив a (n). Найти количество, все номера и произведение элементов массива меньших 1.

2. Создать динамические массивы, используя указатели. Дано 2 массива x(n) и y(m). Сколько раз встречается второй элемент первого массива x(n) во втором массиве y(m).

3. Создать динамические массивы, используя указатели. В каком из двух данных массивов p (n) q(n) больше отрицательных элементов?

4. Создать динамические массивы, используя указатели. Дан массив p(n). Каждый положительный элемент в нем возвести в квадрат. Остальные элементы оставить прежними.

5. Создать динамические массивы, используя указатели. Задан одномерный массив a (n). Найти номер последнего элемента меньшего заданного числа betta, количество положительных элементов и сумму элементов больших 3.

6. Создать динамические массивы, используя указатели. В каком из двух данных массивов p (n) q(n) больше положительных элементов?

7. Создать динамические массивы, используя указатели. Дано 2 массива x(n) и y(m). Сколько раз встречается первый элемент первого массива x(n) во втором массиве y(m).

8. Создать динамические массивы, используя указатели. Дан массив r(n). Каждый элемент равный 0 в нем заменить на 1. Остальные оставить прежними.

9. Создать динамические массивы, используя указатели. Задан одномерный массив a (n). Найти номер последнего элемента равного 5 и переставить его с первым элементом массива. Найти среднее арифметическое элементов массива больших заданного числа alfa.

10. Создать динамические массивы, используя указатели. Задан одномерный массив a (n). Найти номер последнего положительного элемента и переставить его с первым элементом массива. Найти количество и сумму элементов отрицательных массива.

11. Создать динамические массивы, используя указатели. Дано 2 массива x(n) и y(m). Сколько раз встречается последний элемент первого массива x(n) во втором массиве y(m).

12. Создать динамические массивы, используя указатели. В каком из двух данных массивов p (n) q(n) больше нулевых элементов?

13. Создать динамические массивы, используя указатели. Задан одномерный массив a (n). Найти все номера и среднее арифметическое отрицательных элементов массива

14. Создать динамические массивы, используя указатели. Дано 2 массива x(n) и y(m). Сколько раз встречается второй элемент второго массива y(m) в первом массиве x(n).

15. Создать динамические массивы, используя указатели. Дано 2 массива x(n) и y(m). Сколько раз встречается первый элемент второго массива y(m) в первом массиве x(n).

16. Создать динамические массивы, используя указатели. В каком из двух данных массивов p (n) q(n) больше элементов, равных 1?

Задание 3. Решить задачи с использованием указателей, динамической памяти и функций. В виде функций оформить ввод, вывод и обработку массива согласно номеру варианта. В main должно быть только объявление массива, выделение динамической памяти и вызов объявленных функций.

1. Создать динамические массивы, используя указатели. Дан двумерный массив a, размером (n×m). Заполнить одномерный массив, найдя сумму положительных элементов в каждом столбце матрицы.
2. Создать динамические массивы, используя указатели. Дан двумерный массив a, размером (n×m). Заполнить одномерный массив, найдя произведение положительных элементов в каждом столбце матрицы.
3. Создать динамические массивы, используя указатели. Дан двумерный массив a, размером (n×m). Найти минимальный элемент в каждой строке матрицы. Затем каждую строку матрицы разделить на минимальный элемент строки.
4. Создать динамические массивы, используя указатели. Дан двумерный массив a, размером (n×m). Найти минимальный элемент в каждой строке матрицы среди положительных элементов.
5. Создать динамические массивы, используя указатели. Дан двумерный массив a, размером (n×m). Заполнить одномерный массив, найдя количество положительных элементов в каждом столбце матрицы
6. Создать динамические массивы, используя указатели. Дан двумерный массив a, размером (n×m). Найти минимальный элемент в каждой строке матрицы среди отрицательных элементов.
7. Создать динамические массивы, используя указатели. Дан двумерный массив a, размером (n×m). Заполнить одномерный массив, найдя среднее арифметическое положительных элементов в каждом столбце матрицы

8. Создать динамические массивы, используя указатели. Дан двумерный массив a, размером (n×m). Найти минимальный элемент в каждой строке матрицы. Затем каждую строку матрицы умножить на минимальный элемент строки.
9. Создать динамические массивы, используя указатели. Дан двумерный массив a, размером (n×m). Заполнить одномерный массив, найдя среднее геометрическое положительных элементов в каждом столбце матрицы
10. Создать динамические массивы, используя указатели. Дан двумерный массив a, размером (n×m). Найти минимальный элемент в каждой строке матрицы. Затем к каждому элементу каждой строки прибавить минимальный элемент строки.
11. Создать динамические массивы, используя указатели. Дан двумерный массив a, размером (n×m). Найти минимальный элемент и его номер в каждой строке матрицы. Затем из каждого элемента каждой строки вычесть номер минимального элемента строки.
12. Создать динамические массивы, используя указатели. Дан двумерный массив a, размером (n×m). Заполнить одномерный массив, найдя сумму отрицательных элементов в каждом столбце матрицы.
13. Создать динамические массивы, используя указатели. Дан двумерный массив a, размером (n×m). Найти номер минимального элемента в каждой строке матрицы. Затем к каждому элементу каждой строки прибавить номер минимального элемента строки.
14. Создать динамические массивы, используя указатели. Дан двумерный массив a, размером (n×m). Заполнить одномерный массив, найдя произведение отрицательных элементов в каждом столбце матрицы
15. Создать динамические массивы, используя указатели. Дан двумерный массив a, размером (n×m). Заполнить одномерный массив, найдя количество отрицательных элементов в каждом столбце матрицы