Тема 6: Двумерные массивы. Указатели. Метод «сверху вниз»

Задание 1.

1) К чему приведет выполнение следующего фрагмента кода?

```
{
   int c[3][3] = { {10, 11, 12}, {13}, {14, 15, 16} };
   std::cout << c[1][1] << '\n';
}
```

Вывод элемента 0

2) Пусть определена матрица:

```
int cc[3][4] = {0};
```

Укажите правильный способ записи через указатели оператора `cc[p][q] = 3?` *(*(cc+p)+q) = 3

3) Есть функция с прототипом: `void test1(int a[4][5]);` и массивы:

```
int c[4][5] ={0};
int b[5][4] = {0};
int a[20] = {0};
```

Выберите правильный вызов.

- `test1(c[4][5]);`
- `test1(a);`
- `test1(&c[0][0]);`
- `test1(c);`

4) Пусть определена матрица:

```
int** bb = new int* [3] {nullptr};

for (int i = 0; i < 2; ++i) {
    *(bb + i) = new int[3] {0};
}
```

Укажите правильный способ записи оператора `bb[k][m] = 2;` ?

- `**(bb + k * 3 + m) = 2;`
- `*(*(bb + m) + k) = 2;`
- `*(*(bb + k) + m) = 2;`
- нет правильного ответа

5) Есть функция с прототипом: `void test4(int *a, int m);` и массивы:

```
int** dd = new int* [3] {nullptr};


for (int i = 0; i < 3; ++i) {
    *(dd + i) = new int[4] {0};
}
```

```
int a[4][5] = {0};
```

Какие из вызовов функции правильные?

- `test4(dd + 1, 4);`
- `test4(dd[0][0], 3);`
- `test4(&a[0][0], 5);`
- `test4(&dd[0][0], 3);`
- нет правильного вызова

Задание 2.

В каждой функции обращение к элементам массива записать двумя способами: традиционно, используя индексы, и через указатели. Один из вариантов указать в комментарии.

1) Напишите функцию inArray, имеющую три параметра – адрес матрицы (целые числа, **встроенный массив**), количество строк и количество столбцов, и выполняющую ввод всех элементов массива из потока cin. Функция должна инициировать исключение при обнаружении ошибки ввода.
2) Напишите функцию inArray, имеющую три параметра – адрес матрицы (целые числа, **массив в динамической памяти**), количество строк и количество столбцов, и выполняющую ввод всех элементов массива из потока cin. Функция должна инициировать исключение при обнаружении ошибки ввода.
3) Напишите функцию outArray, имеющую три параметра – адрес матрицы (целые числа, **встроенный массив**), количество строк и количество столбцов, и выполняющую вывод всех элементов массива в поток cout. Каждую строку матрицы надо выводить с новой строки.
4) Напишите функцию outArray, имеющую три параметра – адрес матрицы (целые числа, **массив в динамической памяти**), количество строк и количество столбцов, и выполняющую вывод всех элементов массива в поток cout. Каждую строку матрицы надо выводить с новой строки.
5) Напишите функцию getMaxOfArray, имеющую два параметра – адрес матрицы (целые числа, **встроенный массив**), количество элементов в матрице. Учесть расположение элементов встроенной матрицы в памяти.
6) Напишите функцию getNumberOfOrderedRows, имеющую три параметра – адрес матрицы (целые числа, **массив в динамической памяти**), количество строк и количество столбцов, и возвращающую **количество строк, элементы которых упорядочены по возрастанию**. Для проверки является ли строка упорядоченной использовать функцию **isOrderedArray** (см. Тема 5). Функция isOrderedArray, имеет два параметра – адрес одномерного массива (целые числа) и количество элементов, возвращающую значение true, если элементы массива упорядочены по возрастанию, и false –в противном случае.
7) Напишите функцию arithmeticMeansOfPositive, имеющую четыре параметра – адрес матрицы (целые числа, **массив в динамической памяти**), адрес вектора (i-ый элемент будет содержать **среднее арифметическое положительных элементов i-той строки матрицы**), количество строк и столбцов. Функция должна инициировать исключение при невозможности построения вектора.

8) Напишите функции для тестирования функций 1..7:
   - опишите встроенный массив (размер задается поименованными константами) и массив в динамической памяти;
   - введите количество строк и столбцов, выделите память для массива, предусмотрите обработку исключений, если количество задано некорректно и если не память выделена;
   - вызовите функции и выведите ответы.