МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра информационных систем

ОТЧЕТ

по практической работе №3

по дисциплине «Программирование»

Тема: Указатели и многомерные статические массивы

Студентка гр. 0324 Косенко А.Р.

Преподаватель Глущенко А.Г.

Санкт-Петербург

2020

Цель работы.

Изучение структуры многомерных статических массивов, обработка данных многомерных массивов. Получение практических навыков работы с указателями. Изучение простейшей арифметики указателей.

Основные теоретические положения.

Понятие указателя Компилятор, обрабатывая оператор определения переменной, выделяет память в соответствии с типом переменной и инициализирует её указанным значением. Все обращения по имени переменной заменяются компилятором на адрес области памяти, в которой хранится значение переменной. Возможно создание собственных переменных, которые будут хранить какой-то адрес памяти. Такие переменные называются указателями.

Таблица 4.1 – Условное представление памяти

…	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	…

Указатели предназначены для хранения адресов областей памяти (табл. 4.1.). В С++ существует три вида указателей: 1) Указатели на объект, который содержит адрес области памяти, хранящей данные определенного типа. 2) Указатели на функцию. Указатель на функцию содержит адрес сегмента кода, по которому располагается исполняемый код функции. Указатели на функции используются для косвенного вызова функции (через обращение к переменной, хранящей её адрес), а также для передачи имени функции в другую функцию в качестве параметра. Указатель функции должен иметь тип «указатель функции, возвращающей значение заданного типа и имеющей аргументы заданного типа. 3) Указатели на void. Такой указатель применяется в тех случаях, когда тип объекта, адрес которого нужно хранить, не определен. Указателю типа void 46 можно присвоить значение любого типа, но перед выполнением каких-либо действий его нужно явным образом преобразовать к этому типу. Для получения адреса какого-либо программного объекта используют оператор &. Предположим, что у нас имеется две переменные: float A = 3.14 и double = 3.14 (табл. 4.2). Видно, что обе переменные занимают по 4 байта памяти. Они имеют адреса 101 и 105 соответственно.

Таблица 4.2 – Представление в памяти некоторых переменных

		float A = 3.14				double B = 3.14												
…	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	…

Если использовать оператор &, то он вернет адреса этих переменных. Правда, обычно значение адреса памяти выводится в шестнадцатеричном коде. Указатели на объекты определяются в программе следующим образом: *; Тип данных – базовый тип указателя. Имя переменной – идентификатор переменной-указателя. Признаком того, что переменная – указатель, является символ ‘*’, располагающийся перед именем переменной. Определим две переменных-указателя: float *p1; int *p2; Указатель $p1$ предназначен для хранения адресов участков памяти, размер которого соответствует типу int, а переменная $p2$ - для хранения адресов участков памяти, размер которых соответствует типу double. Указатели представляют собой обычные целые числа значения типа int и занимают в памяти 4 байта не зависимо от базового типа указателя. Значения указателей при их выводе на экран представляются как целые значения в шестнадцатеричном формате.

Инициализация указателей.

Листинг 4.1 – Пример использования указателя

```
/* pointer.cpp: Этот файл содержит функцию "main". Здесь начинается и заканчивается
выполнение программы */

#include <iostream>

using namespace std;

int main()
{
	setlocale(LC_ALL, "Russian");
	float pi = 3.14, *p;
	p = &pi;
	cout << "Значение переменной pi: " << pi << endl;
	cout << "Значение переменной pi: " << *p << endl;
	cout << "Адрес переменной pi: " << p << endl;
	return 0;
}
```

Результат выполнения этой программы:

```
Значение переменной pi: 3.14
Значение переменной pi: 3.14
Адрес переменной pi: 012FF77C
```

Присвоить указателю адрес некоторой переменной можно инструкцией присваивания и операции &. Пример инициализации указателя представлен в 47 листинге 4.1. Получить значение объекта, на который ссылается некоторый указатель можно с помощью операции * (разыменовывание указателя).

Из памяти по адресу, который хранится в указатели, берется столько байт памяти, сколько требуется базовому типу указателя. Далее с этими байтами работают, как со значением базового типа указателя. С помощью указателей можно не только получать значения, расположенные по адресам, хранящимся в указателях, но и записывать нужные значения по этим адресам: *p = 1.618; Указатели могут использоваться в различных выражениях наравне с обычными переменными и константами. При использовании указателей в выражениях важно помнить, что операция * имеет наивысший приоритет по отношению к другим операциям (за исключением операции унарный минус). Указателю можно присвоить значение другого указателю, если совпадают их базовые типы. Хотя указатели представляют собой целые значения, присваивать им произвольные целые значения нельзя. Единственным исключением является присвоение указателю нулевого значения. Нулевое значение указателя означает то, что указатель не на что не ссылается.

Арифметика указателей

К указателям можно применять некоторые арифметические операции, одни из них +, −, ++, − −. Результаты выполнения этих операций по отношению к указателям существенно отличаются от результатов соответствующих арифметических операций, выполняющихся с обычными числовыми данными.

Таблица 4.3 – Представление переменные в памяти

		int A = 20				int B = 30				int p1 = 100								
…	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	…

Указатель p1 содержит адрес переменной A. Если выполнить операцию p1 = p1 + 1; (или p1++;) Значение указателя изменится, и станет равно 104. Таким образом, добавление или вычитание 1 из указателя приводит к изменению его значения на размер базового типа указателя. При этом добавлять или вычитать из указателей можно только целые значения. Не стоит забывать о приоритете операций: *p1 + 1; и *(p1 + 1); имеют совершенно разный смысл.

Указатели и массивы

Между массивами и указателями имеется очень тесная связь. Когда в программе определяется некоторый массив. Имя переменной без индексов представляет собой указатель на первый элемент массива: int Arr[5]; 49 Если вывести на экран значение переменной Arr, то будет отображено некоторое

целое значение в шестнадцатеричном формате, которое соответствует адресу первого элемента этого массива. Именно из-за этого операция присвоения сразу всех значений одного массива в другому запрещена в C++. Попытка присвоения Arr1 = Arr2; привела бы к тому, что переменная Arr1, стала бы указывать на ту же область памяти, что и переменная Arr2. Адрес, который ранее хранился в переменной Arr1, был бы утерян, что привело бы к утечке памяти. Более того, по этой причине запрещены любые изменения значения переменной массива. Указателю, который имеет такой же базовый тип, как и элемент массива, можно присвоить указатель на массив. Но обратное присвоение выполнить невозможно, так как переменная массива – это константа, изменение которой запрещено. Так как переменная массива является указателем на первый элемент массива, появляются дополнительные возможности по работе с массивами на основе использования арифметики указателей. Например, Arr[4] эквивалентно *(Arr + 4). Первое выражение – это пример обычной индексации элементов массива. Во втором выражении использовалась арифметика указателей, и с помощи операции + был получен адрес пятого элемента массива. Здесь нельзя забывать о приоритете операций. Использование арифметики указателей при работе с массивами приводит обычно к уменьшению объема генерируемого кода программы и к уменьшению времени ее выполнения, то есть к увеличению быстродействия. Поскольку указатель и имя массива взаимосвязаны, указатели можно индексировать, как обычные массивы, а также создавать массивы указателей.

Объявление многомерных массивов Многомерные массивы определяются аналогично одномерным массивам. Количество элементов по каждому измерению указывается отдельно в квадратичных 50 скобках. Элементы многомерных массивов располагаются друг за другом в непрерывном участке памяти. При инициализации многомерного массива он представляется либо как массив из массивов, при этом каждый массив заключается в свои фигурные скобки (в этом случае левую размерность при описании можно не указывать), либо задается общий список элементов в том порядке, в котором элементы располагаются в памяти:

```
int arr[][] = { {0,0}, {1,1}, {1,0} };
int arr[3][3] = { 0, 0, 0, 1, 1, 1, 2, 2, 2 };
```

Определить, сколько памяти выделено под многомерный массив, можно аналогично одномерному массиву с помощью операции sizeof. Необходимо определить, сколько выделяется памяти на 1 элемент массива и умножить на

количество элементов массива. В листинге 4.2 представлен пример заполнения многомерного массива.

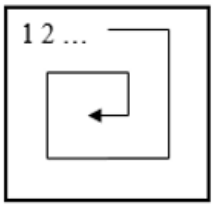
Листинг 4.2 – Пример инициализации двумерного массива

```
/* arr[NxM].cpp: Этот файл содержит функцию "main". Здесь начинается и заканчивается
выполнение программы */

#include <iostream>
#include <ctime>

using namespace std;

int main()
{
	setlocale(LC_ALL, "Russian");
	const int N = 10, M = 10; // Количество строк матрицы N = 10, столбцов – M = 10
	srand(time(0));
	int arr[N][M]; // Объявляем массив arr размерности NxM
	cout << "Исходный массив данных: ";
	for (int i = 0; i < N; i++)
	{
		for (int j = 0; j < M; j++)
		{
			arr[i][j] = rand() % 9;
			cout << arr[i][j] << " ";
		}
		cout << endl;
	}
	return 0;
}
```

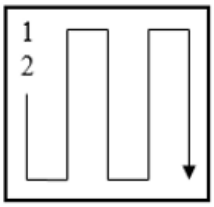
В приведенном выше примере сначала объявляется двумерный массив, затем при помощи двух циклов (один из которых вложенный) происходит заполнение массива случайными числами. Первый цикл нужен, чтобы пройти все строчки 51 матрицы, а второй цикл – пройти все строчки, заполнить их значениями и вывести на экран.

Постановка задачи.

1) Используя арифметику указателей, заполняет квадратичную целочисленную матрицу порядка *N* (6,8,10) случайными числами от 1 до N*N согласно схемам, приведенным на рисунках. Пользователь должен видеть процесс заполнения квадратичной матрицы.

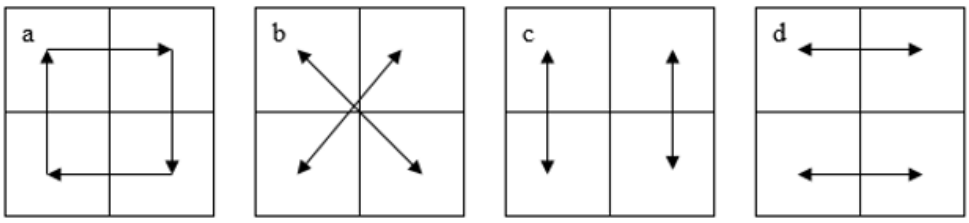


а



б

2) Получает новую матрицу, из матрицы п. 1, переставляя ее блоки в соответствии со схемами:



3) Используя арифметику указателей, сортирует элементы любой сортировкой.

4) Уменьшает, увеличивает, умножает или делит все элементы матрицы на введенное пользователем число.

Выполнение работы.

1) Для выполнения этого задания я использовала функцию **cout**. **width** , которая позволяет вашим программам указать минимальное количество символов, которые будет использовать следующее выходное значение.
2) Чтобы поменять блоки местами, я создала переменную, которая отвечала за порядок+ переменные , которые отвечали за номер строки. Далее создала многомерный массив для более эффективной работы с большими матрицами и выделила для этого память. Далее при помощи указателей я начала обменивать строки между собой. Я выполняла пункт с, в котором , напрмер в матрице 6 на 6, 1 строка должна была стать 1, 5 строка -2, 4 строка-3.
3) Даю пользователю самому создать массив любого порядка . Далее сортировка массива bubble sort при помощи указателей.
4) Через цикл for, при помощи арифметики указателей меняла (действие зависит от выбора пользователя) каждый элемент массива на введенное число.

Вывод

Была создана программа , решающая задания для двумерного массива. Узнала о работе с массивом при помощи арифметики указателей.

Примеры работы

1-3) Вывод спиралью

3 – 6) Переставление блоков

7) Сортировка

8 – 11) Изменение массива на заданное число

```
Матрица до заполнения
 20  35  10   8   6   8
 11  13  34   2  15  32
 15   8  24  23  25  19
 17  29  36   9  16  34
 27  27  27  32  20  20
  6  19  19  31   6   4
Так выглядит массив размерности 6*6 в пункте а:
  1  2  3  4  5  6
 20 21 22 23 24  7
 19 32 33 34 25  8
 18 31 36 35 26  9
 17 30 29 28 27 10
 16 15 14 13 12 11
```

```
Матрица до заполнения
 50   25   36   27   38   30    6   24
 25   42   31   21   44   51   14    7
 28   53   21   18   15    3   53    2
 34   62   29    8   49   42   63   34
  2   34   61   40   63    2   63   24
 43   29   44   23   16   57   29   43
 45   49   60   60   51   48   61   21
 45   25   28   29    3   27   63    4
Так выглядит массив размерности 8*8 в пункте а:
  1    2    3    4    5    6    7    8
 28   29   30   31   32   33   34    9
 27   48   49   50   51   52   35   10
 26   47   60   61   62   53   36   11
 25   46   59   64   63   54   37   12
 24   45   58   57   56   55   38   13
 23   44   43   42   41   40   39   14
 22   21   20   19   18   17   16   15
Выберите размер матрицы:
```

```
Матрица до заполнения
 96  71  35  79  68   2  98   3  18  93
 53  57   2  81  87  42  66  90  45  20
 41  30  32  18  98  72  82  76  10  28
 68  57  98  54  87  66   7  84  20  25
 29  72  33  30   4  20  71  69   9  16
 41  50  97  24  19  46  47  52  22  56
 80  89  65  29  42  51  94   1  35  65
 25  15  88  57  44  92  28  66  60  37
 33  52  38  29  76   8  75  22  59  96
 30  38  36  94  19  29  44  12  29  30
Так выглядит массив размерности 10*10 в пункте а:
  1    2    3    4    5    6    7    8    9   10
 36   37   38   39   40   41   42   43   44   11
 35   64   65   66   67   68   69   70   45   12
 34   63   84   85   86   87   88   71   46   13
 33   62   83   96   97   98   89   72   47   14
 32   61   82   95  100   99   90   73   48   15
 31   60   81   94   93   92   91   74   49   16
 30   59   80   79   78   77   76   75   50   17
 29   58   57   56   55   54   53   52   51   18
 28   27   26   25   24   23   22   21   20   19
Выберите размер матрицы:
```

```
Матрица до перемещения блоков
 21   29    8    8   18    3
 15   17   17   23   13    1
 34   10   30    1   21    4
  7   12   27   24   29   30
 29    7   30   12    3   35
 22    3    7    9   26    4
Матрица после перемещения блоков
 22   3   7   9  26   4
 29   7  30  12   3  35
  7  12  27  24  29  30
 34  10  30   1  21   4
 15  17  17  23  13   1
 21  29   8   8  18   3
```

```
Матрица до перемещения блоков
 60   5  29  20  55  21  52  48
 35  49  37  31  16  51  22  10
 62   3  15  42   9  49  21  51
 11  42  21   9  27  11  61  22
 15  25  41   5  45  28  52  15
 13  24  46  28  10   3  38   7
  5  52  48  14  36   5  64  46
 46  21  55   8  31  51  29  45
Матрица после перемещения блоков
 46  21  55   8  31  51  29  45
  5  52  48  14  36   5  64  46
 13  24  46  28  10   3  38   7
 15  25  41   5  45  28  52  15
 11  42  21   9  27  11  61  22
 62   3  15  42   9  49  21  51
 35  49  37  31  16  51  22  10
 60   5  29  20  55  21  52  48
```

```
Матрица до перемещения блоков
 91   85   77   43   37    8   46   57   80   19
 88   13   49   73   60   10   37   11   43   88
  7    2   14   73   22   56   20  100   22    5
 40   12   41   68    6   29   28   51   85   59
 21   25   23   70   97   82   31   85   93   73
 73   51   26   86   23  100   41   43   99   14
 99   91   25   91   10   82   20   37   33   56
 95    5   80   70   74   77   51   56   61   43
 80   85   94    6   22   68    5   14   62   55
 27   60   45    3    3    7   85   22   43   69
Матрица после перемещения блоков
 27  60  45   3   3   7  85  22  43  69
 80  85  94   6  22  68   5  14  62  55
 95   5  80  70  74  77  51  56  61  43
 99  91  25  91  10  82  20  37  33  56
 73  51  26  86  23 100  41  43  99  14
 21  25  23  70  97  82  31  85  93  73
 40  12  41  68   6  29  28  51  85  59
  7   2  14  73  22  56  20 100  22   5
 88  13  49  73  60  10  37  11  43  88
 91  85  77  43  37   8  46  57  80  19
```

```
>>> 3
Введите порядок матрицы: 5
Введите максимальное число для генерации рандомных чисел: 25
Матрица до сортировки:
  9  12   3  16  19
 11  12  18  25  22
 13   3  16  10  14
  2  16   2  23  12
 12  19  18   5   8
Матрица после сортировки:
  2   2   3   3   5
  8   9  10  11  12
 12  12  12  13  14
 16  16  16  18  18
 19  19  22  23  25
```

```
Введите порядок матрицы: 5
Введите максимальное число для генерации рандомных чисел: 25
Матрица до вычитания:
  6  13  24  18  11
  5   3  23   9  20
 18  19   7  12  18
  5  24  22  20  10
 13  24  25  16  21
Введите вычитаемое: 2
  4  11  22  16   9
  3   1  21   7  18
 16  17   5  10  16
  3  22  20  18   8
 11  22  23  14  19
```

```
Введите порядок матрицы: 6
Введите максимальное число для генерации рандомных чисел: 64
Матрица до сложения:
 14   7  28  53  21  18
 15   3  53   2  34  62
 29   8  49  42  63  34
  2  34  61  40  63   2
 63  24  43  29  44  23
 16  57  29  43  45  49
Введите слагаемое: 3
 17  10  31  56  24  21
 18   6  56   5  37  65
 32  11  52  45  66  37
  5  37  64  43  66   5
 66  27  46  32  47  26
 19  60  32  46  48  52
```

```
>>> 6
Введите порядок матрицы: 5
Введите максимальное число для генерации рандомных чисел: 25
Матрица до умножения:
  4   2   5  25   8
 11   2  19  15  13
  2  12  20  15  21
 21  10   4  18   2
 23   3  18  18   3
Введите множетель: 2
  8   4  10  50  16
 22   4  38  30  26
  4  24  40  30  42
 42  20   8  36   4
 46   6  36  36   6
Выберите задание:
```

```
Введите порядок матрицы: 5
Введите максимальное число для генерации рандомных чисел: 25
Матрица до деления:
  7   2   6  12  17
 16  15  20  20  16
  5   7  18  23  22
  7   1  10   3  18
  7  23   4  12  16
Введите делитель: 2
  3.5     1     3     6   8.5
    8   7.5    10    10     8
  2.5   3.5     9  11.5    11
  3.5   0.5     5   1.5     9
  3.5  11.5     2     6     8
Выберите задание:
```

Код.

//Код писался на 4(это значит, что в каждом задании я выбирала 1 схему),

```
#include <iostream>

using namespace std;


int main()

{
```

```
int variant , razmer ,deistvie ;
razmer = 0;
variant = 0;
deistvie = 0;
while (variant != 8)
{
int variant;
cout << "Выберите задание: " << endl;
cout << endl;
cout << "1. Заполнить матрицу по спирали" << endl;
cout << "2. Переставить блоки матрицы (пункт в)'" << endl;
cout << "3. Сортировать элементы матрицы " << endl;
cout << "4. Уменьшить элементы матрицы на введенное число " << endl;
cout << "5. Увеличить элементы матрицы на введенное число " << endl;
cout << "6. Умножить элементы матрицы на введенное число " << endl;
cout << "7. Разделить элементы матрицы на введенное число " << endl;
cout << "8. ";
cout <<"Выход " << endl;
cout << ">>> ";
cin >> variant;
switch (variant)
{
case (1):
{
do
{
cout << "Выберите размер матрицы: " << endl;
```

```
cout << endl;
cout << "1. Матрица 6x6" << endl;
cout << "2. Матрица 8x8" << endl;
cout << "3. Матрица 10x10" << endl;
cout << "4. Выход" << endl;
cout << ">>> ";
cin >> razmer;

 switch (razmer)
{
 case (1):

{
int a=6;
int A[a][a];
int argc , i=1,j,k, p=a/2;
for (int i = 0; i < a; i++)
{
for (int j = 0; j < a; j++)
{
A[i][j] =1 + rand() % 36 ;//создаю рандомные числа для матрицы
}
}
cout << "Матрица до заполнения\n";
for (int i = 0; i < 6; i++)
{
for (int j = 0; j <6; j++)
```

```
{
cout.width(3);// минимальное количество символов, которые будет использовать следующее выходное значение.
cout <<A[i][j]<<" ";
}
cout <<"\n";
}
for(k=1;k<=p;k++)//k-это номер витка
{
for (j=k-1;j<a-k+1;j++) A[k-1][j]=i++;//Определение значений верхнего столбца
for (j=k;j<a-k+1;j++) A[j][a-k]=i++;// По правому вертикальному столбцу
for (j=a-k-1;j>=k-1;--j) A[a-k][j]=i++;//по нижнему горизонтальному столбцу
for (j=a-k-1;j>=k;j--) A[j][k-1]=i++;// по левому вертикальному столбцу
}
cout << "Так выглядит массив размерности 6*6 в пункте а: \n";
if (a%2==1) A[p][p]=a*a;
for(i=0;i<a;i++)
for(j=0;j<a;j++)
{
cout.width(3);
cout << A[i][j];
if(j==a-1)
cout << "\n";
}
break;
}
case (2):
```

```
{
int b=8;
int B[b][b];
int argc , i=1,j,k, p=b/2;
for (int i = 0; i < b; i++)
{
for (int j = 0; j < b; j++)
{
B[i][j] =1 + rand() % 64 ;
}
}
cout << "Матрица до заполнения\n";
for (int i = 0; i < 8; i++)
{
for (int j = 0; j <8; j++)
{
cout.width(3);
cout <<B[i][j]<<" ";
}
cout <<"\n";
}
for(k=1;k<=p;k++)/*Цикл по номеру витка*/
{
for (j=k-1;j<b-k+1;j++) B[k-1][j]=i++;/*Определение значений верхнего гор столбца*/
for (j=k;j<b-k+1;j++) B[j][b-k]=i++;/* --//-- По правому вертикальному столбцу*/
```

```
for (j=b-k-1;j>=k-1;--j) B[b-k][j]=i++;/* --//-- по нижнему горизонтальному столбцу*/
for (j=b-k-1;j>=k;j--) B[j][k-1]=i++;/* --//-- по левому вертикальному столбцу*/
}
cout << "Так выглядит массив размерности 8*8 в пункте а: \n";
if (b%2==1) B[p][p]=b*b;
for(i=0;i<b;i++)
for(j=0;j<b;j++)
{
cout.width(3);
cout << B[i][j]<< " ";
if(j==b-1)
cout << "\n";
}
break;
}
case (3):
 {
int c=10;
int C[c][c];
int argc , i=1,j,k, p=c/2;
for (int i = 0; i < c; i++)
{
for (int j = 0; j < c; j++)
{
C[i][j] =1 + rand() % 100 ;
}
}
```

```
cout << "Матрица до заполнения\n";
for (int i = 0; i < 10; i++)
{
for (int j = 0; j <10; j++)
{
cout.width(3);
cout <<C[i][j]<<" ";
}
cout <<"\n";
}
for(k=1;k<=p;k++)/*Цикл по номеру витка*/
{
for (j=k-1;j<c-k+1;j++) C[k-1][j]=i++;/*Определение значений верхнего гор столбца*/
for (j=k;j<c-k+1;j++) C[j][c-k]=i++;/* --//-- По правому вертикальному столбцу*/
for (j=c-k-1;j>=k-1;--j) C[c-k][j]=i++;/* --//-- по нижнему горизонтальному столбцу*/
for (j=c-k-1;j>=k;j--) C[j][k-1]=i++;/* --//-- по левому вертикальному столбцу*/
}
cout << "Так выглядит массив размерности 10*10 в пункте а: \n";
if (c%2==1) C[p][p]=c*c;
for(i=0;i<c;i++)
for(j=0;j<c;j++)
{
cout.width(3);
cout << C[i][j]<< " ";
if(j==c-1)
```

```
cout << "\n";
}
break;
}
 }
}
while (razmer!= 4);
break;
}
case(2): {
 do {
 cout << "Выберите размер матрицы: " << endl;
 cout << endl;
 cout << "1. Матрица 6x6" << endl;
 cout << "2. Матрица 8x8" << endl;
 cout << "3. Матрица 10x10" << endl;
 cout << "4. Выход" << endl;
 cout << ">>> ";
 cin >> razmer;

 switch (razmer)
 {
 case (1):
{
 int n=6,n1=1, n2=2, n3=3, n4=4, n5=5,n6=6;//создаю переменную, которая будет отвечать за порядок+ переменные , которые отвечают за номер строки.
int arr[6][6];
```

```
double **f = new double*[6];//Создаю многомерный массив для более эффективной работы с большими матрицами.
for (int i = 0; i < 6; i++)
f[i] = new double[n];//выделяю память
for (int i = 0; i < n; i++)
{
for (int j = 0; j < n; j++)
{
f[i][j] =1 + rand() % 36 ;
}
}

cout << "Матрица до перемещения блоков\n";
for (int i = 0; i < n; i++)
{
for (int j = 0; j <n; j++)
{
cout.width(3);
cout <<f[i][j]<<" ";
}
cout <<"\n";
}

double *temp, *tomp,*timp ;//создаю указатели для обмена строками.
temp=f[n1-1];
tomp=f[n2-1];
timp=f[n3-1];
```

```
f[n1-1]=f[n6-1];
f[n2-1]=f[n5-1];
f[n3-1]=f[n4-1];
f[n4-1]=timp;
f[n5-1]=tomp;
f[n6-1]=temp;
cout << "Матрица после перемещения блоков\n";
for (int i = 0; i < n; i++)
{
for (int j = 0; j <n; j++)
{
cout.width(3);
cout <<f[i][j]<<" ";
}
cout <<"\n";
}
break;
}
case (2):
 {
int n=8,n1=1, n2=2, n3=3, n4=4, n5=5,n6=6, n7=7 ,n8=8;
int arr[8][8];
double **f = new double*[8];
for (int i = 0; i < 8; i++)
f[i] = new double[n];
for (int i = 0; i < n; i++)
{
```

```
for (int j = 0; j < n; j++)
{
f[i][j] =1 + rand() % 64 ;
}
}

cout << "Матрица до перемещения блоков\n";
for (int i = 0; i < n; i++)
{
for (int j = 0; j <n; j++)
{
cout.width(3);
cout <<f[i][j]<<" ";
}
cout <<"\n";
}



double *temp, *tomp,*timp, *tamp ;
temp=f[n1-1];
tomp=f[n2-1];
timp=f[n3-1];
tamp=f[n4-1];
f[n1-1]=f[n8-1];
f[n2-1]=f[n7-1];
f[n3-1]=f[n6-1];
f[n4-1]=f[n5-1];
```

```
f[n5-1]=tamp;
f[n6-1]=timp;
f[n7-1]=tomp;
f[n8-1]=temp;
cout << "Матрица после перемещения блоков\n";
for (int i = 0; i < n; i++)
{
for (int j = 0; j <n; j++)
{
cout.width(3);
cout <<f[i][j]<<" ";
}
cout <<"\n";
}
break;
}

 case (3):
{
int n=10,n1=1, n2=2, n3=3, n4=4, n5=5,n6=6, n7=7 ,n8=8,n9=9,n10=10;
int arr[10][10];
double **f = new double*[10];
for (int i = 0; i < 10; i++)
f[i] = new double[n];
for (int i = 0; i < n; i++)
{
for (int j = 0; j < n; j++)
```

```
{
f[i][j] =1 + rand() % 100 ;
}
}

cout << "Матрица до перемещения блоков\n";
for (int i = 0; i < n; i++)
{
for (int j = 0; j <n; j++)
{
cout.width(3);
cout <<f[i][j]<<" ";
}
cout <<"\n";
}


double *temp, *tomp,*timp, *tamp , *tump ;
temp=f[n1-1];
tomp=f[n2-1];
timp=f[n3-1];
tamp=f[n4-1];
tump=f[n5-1];
f[n1-1]=f[n10-1];
f[n2-1]=f[n9-1];
f[n3-1]=f[n8-1];
f[n4-1]=f[n7-1];
```

```
f[n5-1]=f[n6-1];
f[n6-1]=tump;
f[n7-1]=tamp;
f[n8-1]=timp;
f[n9-1]=tomp;
f[n10-1]=temp;
cout << "Матрица после перемещения блоков\n";
for (int i = 0; i < n; i++)
{
for (int j = 0; j <n; j++)
{
cout.width(3);
cout <<f[i][j]<<" ";
}
cout <<"\n";
}
break;
}
 }
}
while (razmer != 4);
break;
}
case(3):
{
 int por;
 cout <<"Введите порядок матрицы: ";
```

```
 cin>> por;
 int Arr[por][por], k1;//k1 является максимальным числом для рандомных чисел в матрице
 cout << "Введите максимальное число для генерации рандомных чисел: ";
 cin >> k1;
 for (int i = 0; i < por; i++)
{
for (int j = 0; j < por; j++)
{
Arr[i][j] =1 + rand() % k1 ;
}
}

cout << "Матрица до сортировки: \n";
for (int i = 0; i < por; i++)
{
for (int j = 0; j <por; j++)
{
cout.width(3);
cout <<Arr[i][j]<<" ";
}
cout <<"\n";
}
cout <<"Матрица после сортировки: \n";
for (int i = 0; i < por*por; i++)//Начинаю сортировку
for (int j = 0; j < por*por-1; j++)
if (*(*Arr + j) > *(*Arr + j + 1))
swap(*(*Arr + j), *(*Arr + j + 1));
```

```
for (int i = 0; i < por; i++)
{
 for (int j = 0; j < por; j++)
 {
cout.width(3);
cout << *(*(Arr + i) + j) << " ";
}
cout << endl;
}
 break;
 }

 case (4):
{
 int por;
 cout <<"Введите порядок матрицы: ";
 cin>> por;
 int Arr[por][por],chislo, k1;//k1 является максимальным числом для рандомных чисел в матрице
 cout << "Введите максимальное число для генерации рандомных чисел: ";
 cin >> k1;
 for (int i = 0; i < por; i++)
{
for (int j = 0; j < por; j++)
{
Arr[i][j] =1 + rand() % k1 ;
}
}
```

```
cout << "Матрица до вычитания: \n";
for (int i = 0; i < por; i++)
{
for (int j = 0; j <por; j++)
{
cout.width(3);
cout <<Arr[i][j]<<" ";
}
cout <<"\n";
}
cout <<"Введите вычитаемое: ";
cin >> chislo;
for (int i = 0; i < por; i++)
{
 for (int j = 0; j < por; j++)
{
cout.width(3);
cout << Arr[i][j]-chislo << " ";
}
 cout << endl;
}

 break;
}
case (5):
{
```

```
 int por;
 cout <<"Введите порядок матрицы: ";
 cin>> por;
 int Arr[por][por],chislo, k1;//k1 является максимальным числом для рандомных чисел в матрице
 cout << "Введите максимальное число для генерации рандомных чисел: ";
 cin >> k1;
 for (int i = 0; i < por; i++)
{
for (int j = 0; j < por; j++)
{
Arr[i][j] =1 + rand() % k1 ;
}
}
cout << "Матрица до сложения: \n";
for (int i = 0; i < por; i++)
{
for (int j = 0; j <por; j++)
{
cout.width(3);
cout <<Arr[i][j]<<" ";
}
cout <<"\n";
}
cout <<"Введите слагаемое: ";
cin >> chislo;
for (int i = 0; i < por; i++)
{
```

```
for (int j = 0; j < por; j++)
{
cout.width(3);
cout << Arr[i][j]+chislo << " ";
}
 cout << endl;
}
break;
}
case (6):
{
 int por;
 cout <<"Введите порядок матрицы: ";
 cin>> por;
 int Arr[por][por],chislo, k1;//k1 является максимальным числом для рандомных чисел в матрице
 cout << "Введите максимальное число для генерации рандомных чисел: ";
 cin >> k1;
 for (int i = 0; i < por; i++)
{
for (int j = 0; j < por; j++)
{
Arr[i][j] =1 + rand() % k1 ;
}
}
cout << "Матрица до умножения: \n";
for (int i = 0; i < por; i++)
{
```

```
for (int j = 0; j <por; j++)
{
cout.width(3);
cout <<Arr[i][j]<<" ";
}
cout <<"\n";
}
cout <<"Введите множетель: ";
cin >> chislo;
for (int i = 0; i < por; i++)
 {
for (int j = 0; j < por; j++)
{
cout.width(3);
cout << Arr[i][j]*chislo << " ";
}
cout << endl;
 }
break;
}
case (7):
{
 int por;
 cout <<"Введите порядок матрицы: ";
 cin>> por;
 int Arr[por][por],chislo, k1;//k1 является максимальным числом для рандомных чисел в матрице
 cout << "Введите максимальное число для генерации рандомных чисел: ";
```

```
 cin >> k1;
 for (int i = 0; i < por; i++)
{
for (int j = 0; j < por; j++)
{
Arr[i][j] =1 + rand() % k1 ;
}
}
cout << "Матрица до деления: \n";
for (int i = 0; i < por; i++)
{
for (int j = 0; j <por; j++)
{
cout.width(3);
cout <<Arr[i][j]<<" ";

}
cout <<"\n";
}
cout <<"Введите делитель: ";
cin >> chislo;
for (int i = 0; i < por; i++)
 {
for (int j = 0; j < por; j++)
 {
cout.width(5);
 cout << (double)Arr[i][j]/(double)chislo <<" ";
```

```
 }
cout << endl;
}
break;
}
 }
}
return 0;
}
```