Лабораторная работа 3.2
Динамические массивы

Задание:
Изучить теоретический материал по темам:
1. Выделение и освобождение памяти
2. Динамические массивы

Написать программу с меню, которая по запросу пользователя выполняет одно из следующих действий
1. Ввод массива
2. Вывод массива в консоль
3. Сортировка по сумме цифр, стоящих на четных местах
4. Отсортировать массив вначале по возрастанию последней цифры, а при совпадении последних цифр - по убыванию.
5. выход

Критерии оценивания: сделано меню и пункты 1,2,5 – удовлетворительно, предыдущее плюс 3 или 4 – хорошо, все выполнено – отлично.

Форма представления выполненных заданий: файл *Ivanov_Ivan_111_111_lab32.cpp* содержащий решение поставленных задач.

Теоретические сведения:
Для выделения памяти используется оператор new. Если память выделяется посредством оператора new? То освобождаться она должна посредством delete.
Пример: где-то ранее в программе пользователь ввел значение arr_size, тогда можно выделить блок памяти на динамический массив arr размера (arr_size*<размер int>):

```
arr = new int[arr_size];
```

Затем этот кусок нужно удалить:

```
delete[] arr;
```

В качестве примера работы с динамическим массивом рассмотрим операции над матрицами произвольного размера. Матрица размера ширины columns и высоты rows будет храниться по строкам в виде одномерного массива. Тогда справедливо

$$a_{ij} = Matr[i * columns + j]$$

Тогда для реализации ввода матрицы можно использовать следующий код:

```
void inputMatrix(int * &matr, int * matr_col, int * matr_rows)
{
	cout << "Input size of matrix: ";
	cin >> *matr_rows >> *matr_col;
	matr = new int[(*matr_col)*(*matr_rows)];
	for (int i = 0; i < (*matr_col)*(*matr_rows); i++)
		cin >> matr[i];
}
```

Выделенные элементы говорят о том, что аргументы передаются в функцию по ссылке, т.е. их значения могут быть изменены внутри функции.

Для вывода матрицы на экран можно использовать следующую функцию:

```
void printMatrix(int* matr, int matr_col, int matr_rows)
{
	for (int i = 0; i < matr_rows; i++)
	{
```

```
                for (int j = 0; j < matr_col; j++)
                {
                        cout << matr[i*matr_col + j] << "\t";
                }
                cout << std::endl;
        }
}
```

Здесь уже все аргументы передаются по значению. Причем, прошу обратить внимание, что в функцию передается значение указателя на массив, где хранится матрица. Из этого следует, что значение элементов матрицы внутри функции изменять можно, а вот размер матрицы менять нельзя.

Перейдем к сложению матриц. Сумма матриц считается поэлементно:

$$mat1 + mat2 = mat3$$
$$mat3_{ij} = mat1_{ij} + mat2_{ij}$$

Операцию сложения можно применять только для матриц одинакового размера.
Рассмотрим функцию сложения матриц, причем результирующая матрица передается по ссылке (то есть можно менять размер матрицы и значение элементов), а матрицы-слагаемые передаются по значению.

```
void matrixsum(int * mat1, int mat1col,int mat1rows,
        int * mat2, int mat2col, int mat2rows,
        int * &mat3, int *mat3col, int *mat3rows)
{
        if (mat1col==mat2col && mat1rows==mat2rows)
        {
                *mat3rows = mat2rows;
                *mat3col = mat2col;
                mat3 = new int[mat2rows*mat2col];
                for (int i = 0; i < mat2rows*mat2col; i++)
                        //mat3[i,j] = mat1[i, j]+mat2[i,j]
                        mat3[i] = mat1[i] + mat2[i];
        }
        else
        {
                cout << "Error of sum: sizes are not equal.";
                *mat3col = *mat3rows = 0;
        }
}
```

Перейдем к произведению матриц. Произведение матриц вычисляется как произведение строки на столбец:

$$mat3 = mat1 * mat2$$
$$mat3_{ij} = \sum_{k} mat1_{ik} * mat2_{kj}$$

Произведение матриц применимо лишь для таких матриц, что ширина первой матрицы совпадает с высотой второй, причем высота результирующей матрицы равна высоте первого сомножителя, а ширина результирующей матрицы совпадает с шириной второго сомножителя.
Рассмотрим реализацию функции произведения матриц, причем результирующая матрица передается по ссылке (то есть можно менять размер матрицы и значение элементов), а матрицы-слагаемые передаются по значению.

```
void matrixmult(int * mat1, int mat1col,int mat1rows,
        int * mat2, int mat2col, int mat2rows,
        int * &mat3, int *mat3col, int *mat3rows)
```

```
{
	if (mat1col== mat2rows)
	{
		*mat3rows = mat1rows;
		*mat3col = mat2col;
		mat3 = new int[mat3rows*mat3col];
		for (int i = 0; i < mat1rows; i++)
			for (int j = 0; j < mat2col; j++)
			{
				mat3[i*mat2col+j]=0;
				for (int k = 0; k < mat1col; k++)
					mat3[i*mat2col+j]+= mat1[i*mat1col+k] + mat2[k*mat2col+j];
			}
	}
	else
	{
		cout << "Error of mult: sizes are not equal.";
		*mat3col = *mat3rows = 0;
	}
}
```