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»

Студентка гр. 0324 _____ Жигалова Д.А.

Преподаватель _____ Глущенко А.Г

Санкт-Петербург
2020

Цель работы

Знакомство с указателями и способами работы с ними, а также изучение арифметики указателей.

Основные теоретические положения

Указатель – переменная, значением которой является адрес ячейки памяти. То есть указатель ссылается на блок данных из области памяти, причём на самое его начало. Указатель может ссылаться на переменную или функцию. Для этого нужно знать адрес переменной или функции. Так вот, чтобы узнать адрес конкретной переменной в C++ существует унарная операция взятия адреса &. Такая операция извлекает адрес объявленных переменных, для того, чтобы его присвоить указателю.

Указатели используются для передачи по ссылке данных, что намного ускоряет процесс обработки этих данных (в том случае, если объём данных большой), так как их не надо копировать, как при передаче по значению, то есть, используя имя переменной. В основном указатели используются для организации динамического распределения памяти, например при объявлении массива, не надо будет его ограничивать в размере. Любой указатель необходимо объявить перед использованием, как и любую переменную. При использовании простых переменных каждой области памяти для хранения данных соответствует свое имя. Если с группой величин одинакового типа требуется выполнить однообразные действия, им дают одно имя, а различают по порядковому номеру (индексу). Это дает возможность компактно записать множество операций с использованием циклов.

Принцип объявления указателей такой же, как и принцип объявления переменных. Отличие заключается только в том, что перед именем ставится символ звёздочки *. Визуально указатели отличаются от переменных только одним символом. При объявлении указателей компилятор выделяет несколько байт памяти, в зависимости от типа данных отводимых для хранения некоторой информации в памяти. Чтобы получить значение, записанное в некоторой

области, на которое ссылается указатель нужно воспользоваться операцией разыменования указателя *. Необходимо поставить звёздочку перед именем и получим доступ к значению указателя.

Формально указатели представляют собой обычные целые 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.

Указатели поддерживают ряд операций: присваивание, получение адреса указателя, получение значения по указателю, некоторые арифметические операции и операции сравнения.

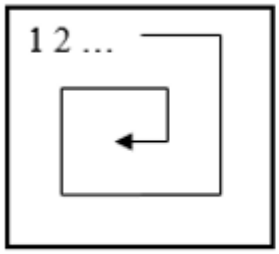
К указателям можно применять некоторые арифметические операции. К таким операциям относятся: +, -, ++, --. Результаты выполнения этих операций по отношению к указателям существенно отличаются от результатов соответствующих арифметических операций, выполняющихся с обычными числовыми данными.

Указатели – это очень мощное, полезное, но и очень опасное средство. Ошибки, которые возникают при неправильном использовании указателей, кроме того, что могут приводить к серьезным и непредсказуемым ошибкам в работе программы, еще и очень трудно диагностировать (обнаруживать).

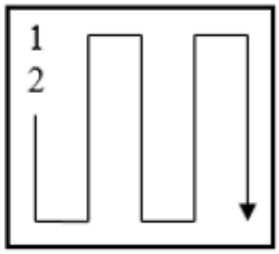
Постановка задачи

Необходимо написать программу, которая:

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.

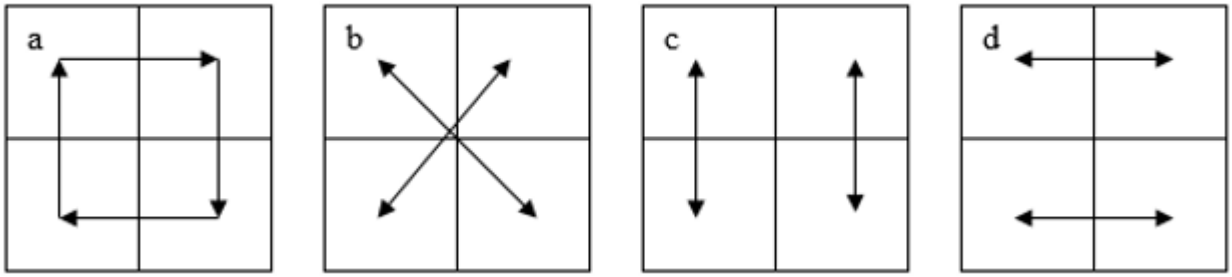


а



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

Для решения поставленных была создана программа на языке программирования C++. Итоговый код программы представлен в приложении А, а результат работы в приложении В.

В первой задаче были созданы две функции, которые через цикл for заполняют матрицу (6*6) в соответствии со схемами через указатели.

Во второй задаче для перестановки блоков было создана функция, которая через указатель на первый элемент матрицы переставляла блоки в соответствии со схемами через цикл for.

В третьей задаче была создана функция пузырьковой сортировки, работающая через указатели.

В четвертой задаче также была создана функция, которая в соответствии с указаниями пользователя производится уменьшение, увеличение, умножение или деление всех элементов матрицы на введенное пользователем число. Функция также производит работу через указатели.

Вывод

При разработке программы мною были изучены указатели и арифметика указателей. А также проведена работа с указателями на двумерный массив.

ПРИЛОЖЕНИЕ А
ПОЛНЫЙ КОД ПРОГРАММЫ

```
#include <iostream>
#include <thread>
#include <chrono>

using namespace std;

void animation(int *p, int s) {
    system("CLS");
    for (int i = 0; i < s; ++i) {
        for (int j = 0; j < s; ++j) {
            cout << *(p + s * i + j) << " ";
        }
        cout << endl;
    }
    this_thread::sleep_for(chrono::milliseconds(40));
}

void withoutAnimation(int* p, int s) {
    for (int i = 0; i < s; ++i) {
        for (int j = 0; j < s; ++j) {
            cout << *(p + s * i + j) << " ";
        }
        cout << endl;
    }
    cout << endl;
}

void fillSnake(int* p, int s) {

    int counter = 1;

    for (int i = 0; i < s; ++i) {
        for (int j = 0; j < s; ++j) {
            *(p + s * j + i) = 0;
        }
    }
    for (int i = 0; i < s; i++) {
        if ((i % 2) != 0) {
            for (int j = (s-1); j >= 0; j--) {
                *(p + s * j + i) = rand() % (s * s) + 1;
                //*(p + s * j + i) = counter; для проверки корректной
работы
                animation(p, s);
                counter++;
            }
```

```
        }
        else {
            for (int j = 0; j < s; j++) {
                *(p + s * j + i) = rand() % (s * s) + 1;
                //*(p + s * j + i) = counter; для проверки корректной
работы
                animation(p, s);
                counter++;
            }
        }
    }
}

void fillSpiral(int* p, int s) {

    for (int i = 0; i < s; ++i) {
        for (int j = 0; j < s; ++j) {
            *(p + s * j + i) = 0;
        }
    }
    int c = 1, j, k = 0, d = 1;

    while (c < (s * s + 1))
    {
        k++;
        for (j = k - 1; j < s - k + 1; j++)
        {
            *(p + (k - 1) * s + j) = rand() % (s * s) + 1;
            //*(p + (k - 1) * s + j) = c;
            animation(p, s);
            c++;
        }   //верх

        for (j = k; j < s - k + 1; j++)
        {
            *(p + j * s + (s - k)) = rand() % (s * s) + 1;
            //*(p + j * s + (s-k)) = c;
            animation(p, s);
            c++;
        }   //право

        for (j = s - k - 1; j >= k - 1; j--)
        {
            *(p + (s - k) * s + j) = rand() % (s * s) + 1;
            //*(p + (s - k) * s + j) = c;
            animation(p, s);
            c++;
```

```
        }   //низ

        for (j = s - k - 1; j >= k; j--)
        {
            *(p + j * s + (k - 1)) = rand() % (s * s) + 1;
            //*(p + j * s + (k-1)) = c;
            animation(p, s);
            c++;
        }   //лево
    }
}

void maths(int* p, int s) {

    int answer3 = 0;
    int n;

    while (answer3 != 5) {

        cout << "\n What exectly do you want?\n";
        cout << "\nTo reduce, enter 1\n";
        cout << "\nTo increase, enter 2\n";
        cout << "\nTo multiplie, enter 3\n";
        cout << "\nTo divide, enter 4\n";
        cout << "\nTo exit, enter 5 \n";
        cin >> answer3;

        switch (answer3) {

        case 1:
            cout << "And enter a number:";
            cin >> n;

            cout << "\nBefore:\n";
            withoutAnimation(p, s);
            for (int i = 0; i < s; ++i) {
                for (int j = 0; j < s; ++j) {
                    *(p + s * j + i) = *(p + s * j + i) - n;
                }
            }
            cout << "\nAfter:\n";
            withoutAnimation(p, s);
            break;

        case 2:
            cout << "And enter a number:";
            cin >> n;
```

```
    cout << "\nBefore:\n";
    withoutAnimation(p, s);
    for (int i = 0; i < s; ++i) {
        for (int j = 0; j < s; ++j) {
            *(p + s * j + i) = *(p + s * j + i) + n;
        }
    }
    cout << "\nAfter:\n";
    withoutAnimation(p, s);
    break;

case 3:
    cout << "And enter a number:";
    cin >> n;

    cout << "\nBefore:\n";
    withoutAnimation(p, s);
    for (int i = 0; i < s; ++i) {
        for (int j = 0; j < s; ++j) {
            *(p + s * j + i) = *(p + s * j + i) * n;
        }
    }
    cout << "\nAfter:\n";
    withoutAnimation(p, s);
    break;

case 4:
    cout << "And enter a number:";
    cin >> n;

    cout << "\nBefore:\n";
    withoutAnimation(p, s);
    for (int i = 0; i < s; ++i) {
        for (int j = 0; j < s; ++j) {
            *(p + s * j + i) = *(p + s * j + i) / n;
        }
    }
    cout << "\nAfter:\n";
    withoutAnimation(p, s);
    break;

case 5:
    break;

default:
```

```
            cout << "Oh, no! This task doesn't exist. Let's try
again.\n";
            break;
        }
    }
    system("CLS");
}

void bubbleSort(int* p, int s) {

    bool sw = 1;

    while (sw != 0) {
        sw = 0;
        for (int i = 0; i < (s * s - 1); i++) {
            if ((*(p + i)) > (*(p + i + 1))) {
                swap((*(p + i)), (*(p + i + 1)));
                sw = 1;
            }
        }
    }
}

void blocks(int* p, int s) {

    int answer1 = 0;
    int half = s / 2;
    int t;

    while (answer1 != 5) {

        cout << "\n What exectly do you want?\n";
        cout << "\na)To rearrange with a snake, enter 1\n";
        cout << "\nb)To rearrange the diagonal, enter 2\n";
        cout << "\nc)To rearrange the vertical, enter 3\n";
        cout << "\nd)To rearrange the horizontal, enter 4\n";
        cout << "\nTo exit, enter 5 \n";
        cin >> answer1;

        switch (answer1) {

        case 1:
            cout << "\nBefore:\n";
            withoutAnimation(p, s);
            for (int i = 0; i < half; ++i) {
                for (int j = 0; j < half; ++j) {
                    t = *(p + s * j + i);
```

```
                    *(p + s * j + i) = *(p + s * (j + half) + i);
                    *(p + s * (j + half) + i) = *(p + s * (j + half) +
(i+half));
                    *(p + s * (j + half) + (i + half)) = *(p + s * j + (i
+ half));
                    *(p + s * j + (i + half)) = t;
                }
            }
            cout << "\nAfter:\n";
            withoutAnimation(p, s);
            break;

        case 2:
            cout << "\nBefore:\n";
            withoutAnimation(p, s);
            for (int i = 0; i < half; ++i) {
                for (int j = 0; j < half; ++j) {
                    t = *(p + s * j + i);
                    *(p + s * j + i) = *(p + s * (j + half) + (i +
half));
                    *(p + s * (j + half) + (i + half)) = t;
                }
            }
            for (int i = 0; i < half; ++i) {
                for (int j = 0; j < half; ++j) {
                    t = *(p + s * (j + half) + i);
                    *(p + s * (j + half) + i) = *(p + s * j  + (i +
half));
                    *(p + s * j + (i + half)) = t;
                }
            }
            cout << "\nAfter:\n";
            withoutAnimation(p, s);
            break;

        case 3:
            cout << "\nBefore:\n";
            withoutAnimation(p, s);
            for (int i = 0; i < s; ++i) {
                for (int j = 0; j < half; ++j) {
                    t = *(p + s * j + i);
                    *(p + s * j + i) = *(p + s * (j + half) + i);
                    *(p + s * (j + half) + i) = t;
                }
            }
            cout << "\nAfter:\n";
            withoutAnimation(p, s);
```

```
            break;

        case 4:
            cout << "\nBefore:\n";
            withoutAnimation(p, s);
            for (int i = 0; i < half; ++i) {
                for (int j = 0; j < s; ++j) {
                    t = *(p + s * j + i);
                    *(p + s * j + i) = *(p + s * j + (i + half));
                    *(p + s * j + (i + half)) = t;
                }
            }
            cout << "\nAfter:\n";
            withoutAnimation(p, s);
            break;

        case 5:
            break;

        default:
            cout << "Oh, no! This task doesn't exist. Let's try
again.\n";
            break;
        }
    }
    system("CLS");
}

int main()
{
    srand(time(0));

    const int size = 6;
    int A[size][size];
    int* p = &A[0][0];

    int end = size*size - 1;
    int begin = 0;

    cout << "\nLet's make some matrix (6*6)\n";
    int answer;
    answer = 0;

    while (answer != 3) {
        cout << "\nTo make new matrix with Snake animation, enter 1\n";
        cout << "\nTo make new matrix with Spiral animation, enter 2\n";
        cout << "\nTo exit, enter 3 \n";
```

```
        cin >> answer;
        system("CLS");

        if (answer == 1 || answer == 2) {
            if (answer == 1)
                fillSnake(p, size);

            if (answer == 2)
                fillSpiral(p, size);

            while (answer != 4) {
                cout << "\nTo make new matrix, rearranging its blocks in
accordance with the schemes(a, b, c, d), enter 1\n";
                cout << "\nTo sort elements by bubble sort, using pointer
arithmetic, enter 2\n";
                cout << "\nTo reduce, increase, multiplie, or divide all
matrix elements by the number entered by the user, enter 3\n";
                cout << "\nTo make new matrix, enter 4 \n";
                cin >> answer;
                system("CLS");

                if (answer == 1 || answer == 2 || answer == 3) {
                    if (answer == 1)
                        blocks(p, size);

                    if (answer == 2) {
                        cout << "\nUnsorted array:\n";
                        withoutAnimation(p, size);
                        bubbleSort(p, size);
                        cout << "\nSorted array:\n";
                        withoutAnimation(p, size);
                    }

                    if (answer == 3)
                        maths(p, size);
                }

                else if (answer == 4) {
                    cout << "Let's make some matrix (6*6)\n";
                }

                else {
                    cout << "Oh, no! This task doesn't exist. Let's try
again.\n";
                }
            }
        }
```

```
        else if (answer == 3) {
            cout << "Have a nice day!\n";
        }

        else {
            cout << "Oh, no! This task doesn't exist. Let's try
again.\n";
        }
    }
    return 0;
}
```

ПРИЛОЖЕНИЕ В
РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ

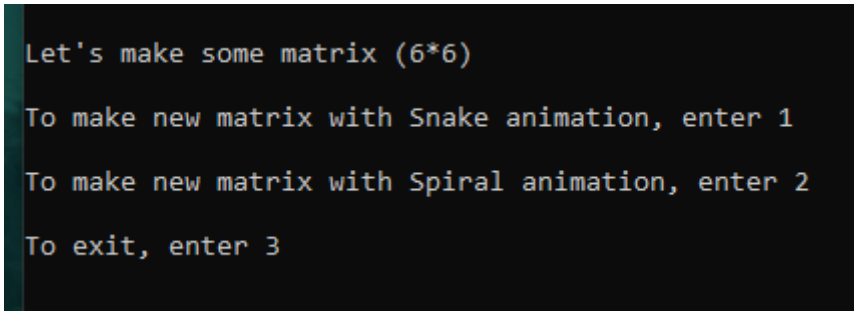


Рисунок 1 - Создание матрицы

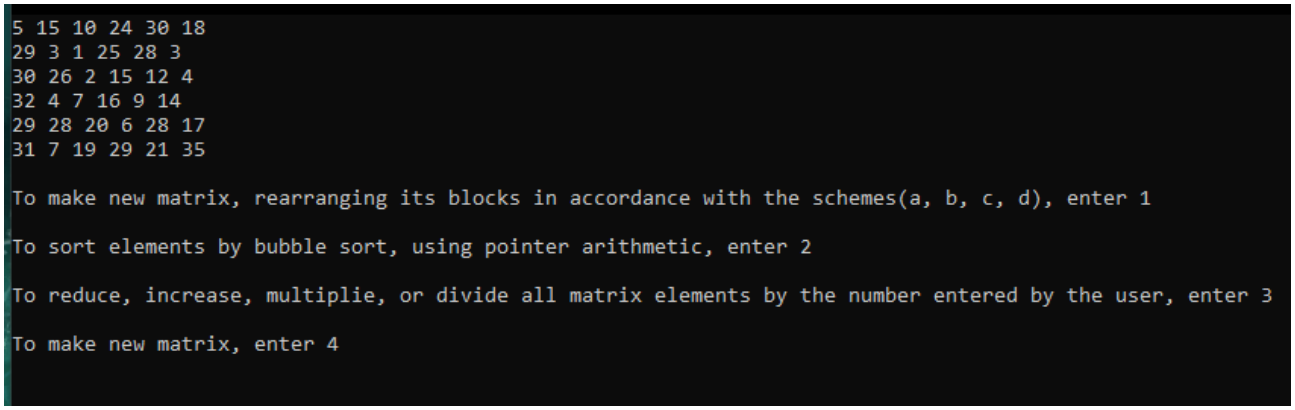


Рисунок 2 - Реализация первого задания, а также меню

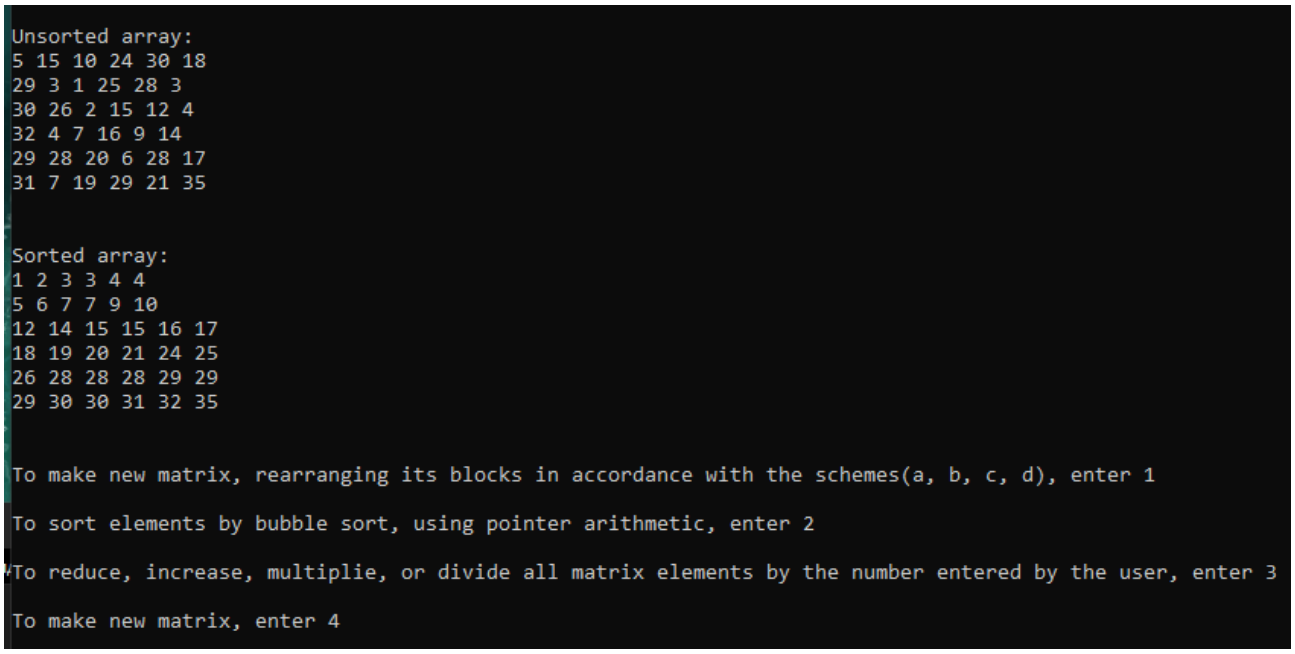


Рисунок 3 - Реализация 3 задания (сортировка выполнена в первую очередь для более наглядного представления других заданий)

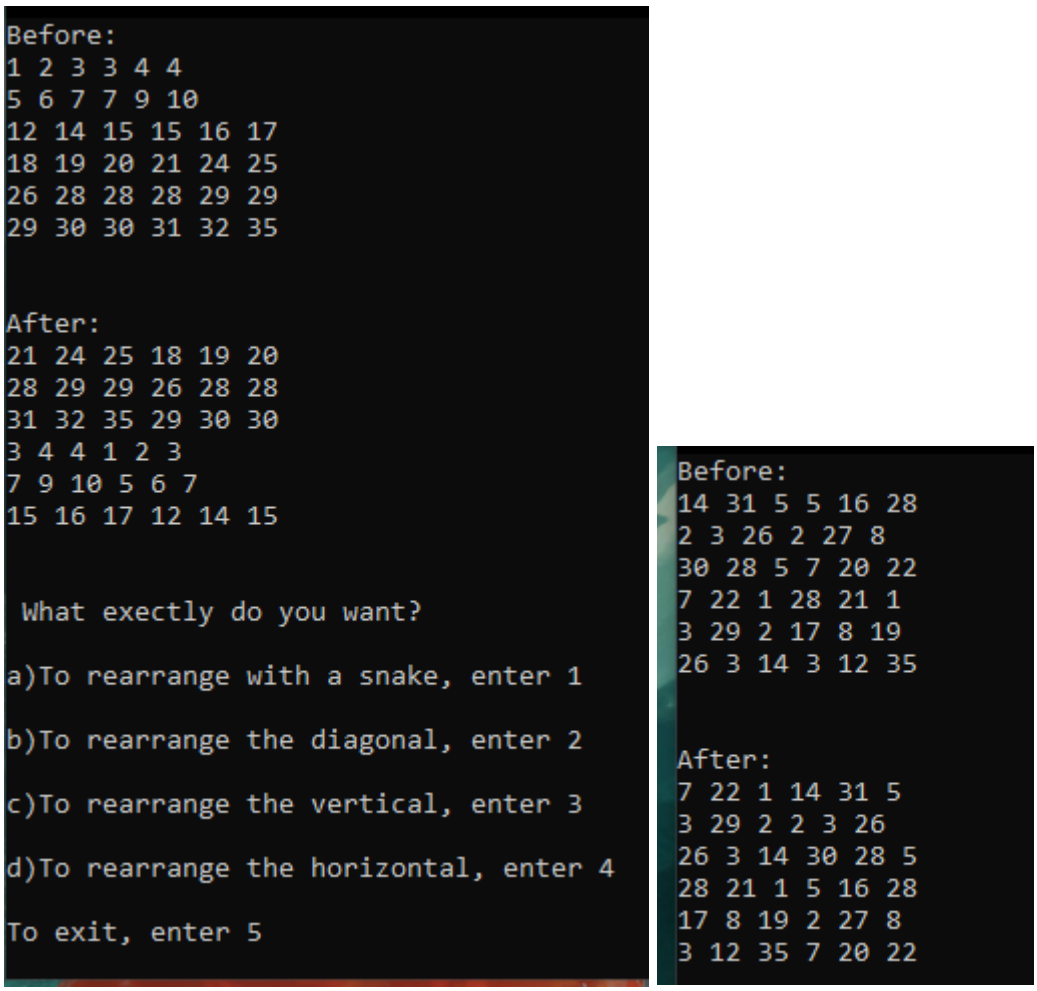


Рисунок 4 – Реализация 2 задания (пункт B и A(взята другая матрица), также работают C и D)

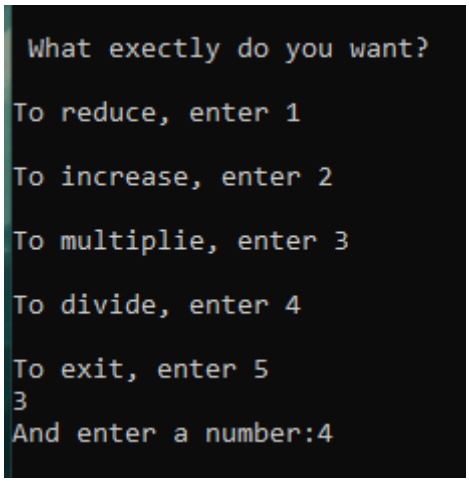


Рисунок 5 – Меню выполнения 4 задания (было выбрано умножение всех элементов на 4)

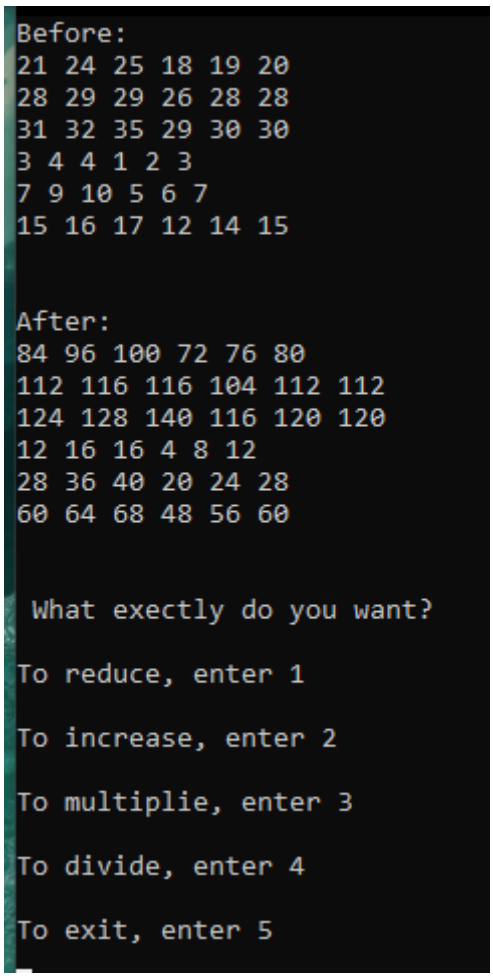


Рисунок 6 – Результат выполнения 4 задания (было выбрано умножение всех элементов на 4)

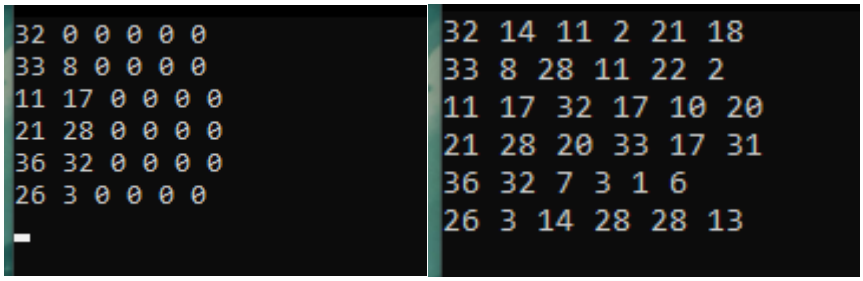


Рисунок 7 – Процесс реализации 1 задания в соответствии со схемой (б)

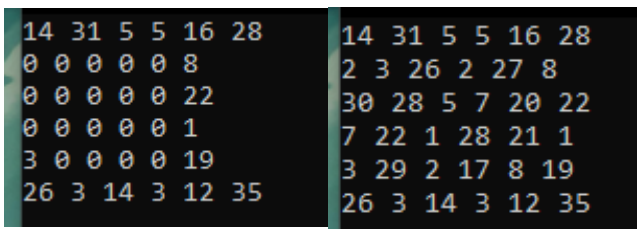


Рисунок 8 – Процесс реализации 1 задания в соответствии со схемой (а)