Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Рязанский государственный радиотехнический университет
имени В. Ф. Уткина»

Кафедра «Вычислительная и прикладная математика»

Отчет
по лабораторной работе № 5

по дисциплине

«Низкоуровневое программирование»

на тему

«Динамическая память»

Выполнил:
студент гр. 143
Вербицкая И. С.

Проверил:
Щенева Ю.Б.

Рязань 2022

Задание (вариант №5):

Задание

В каждом варианте задания требуется создать **целочисленный** динамический двумерный массив, который может становиться «неправильной формы», т.е. каждая строка которого может быть своей длины, причём индексация строк начинается с 1, а в нулевом элементе хранится общее количество элементов данной строки. В начале массив генерируется заданного пользователем размера A x B с элементами заданными случайным образом из заданного пользователем диапазона, а все строки которого изначально будут одинаковой длины B. В последствии необходимо к каждой строке применить одну из 4-х функций по модификации одномерного динамического массива (по классу задач: удаление, добавление, перестановка, поиск), заданных ниже по-вариантно, и в результате удаления\вставок элементов каждая строка может изменять свой размер. Функции к строкам применяются последовательно: т.е. функция «удаление» применяется к 1 строке двумерного массива, функция «добавление» применяется ко 2 строке, функция «перестановка» применяется к 3 строке, функция «поиск» применяется к 4 строке, функция «удаление» применяется к 5 строке, функция «добавление» применяется к 6 строке и т.д. При добавлении элементов соответственно строки динамического массива должны расширяться, а при удалении элементов – уменьшаться. Весь текстовый ввод\вывод должен осуществляться в консоль исключительно из основной функции **main**.

Важно:

1) Функции работают с указателями на динамические **одномерные** массивы, т.е. строки двумерного массива мы передаём построчно для обработки в функции как **одномерные** массивы через указатели.
2) Функции работают с **целочисленными** динамическими одномерными массивами где индексы начинаются с 1, а в 0-вом элементе записан текущий размер динамического массива. При удалении\добавлении элементов функции должны записать новый размер массива в 0-вой элемент.
3) В функциях будет более удобно использовать арифметику указателей и относительные смещения.
4) При добавлении случайных элементов по заданию нужно использовать тот же диапазон генерации, который задал пользователь для генерации исходного двумерного массива.
5) Циклический сдвиг вправо или влево – это когда элементы из конца или начала массива переходят соответственно в его начало или конец.

Вариант	Функции для работы с одномерным динамическим массивом			
	Удаление	Добавление	Перестановка	Поиск
5	N элементов, перед элементом с номером K	N случайных элементов, перед элементом с номером K	Сдвинуть циклически на M элементов вправо	Занулить все элементы с заданным значением

Анализ задания:

ОДЗ: A>0, B>0, mx>mn, k>0, n>0, n<k (для функции удаления), k<=C[i][0] (не больше длины строки i), m>0, m< C[i][0].

Входные данные: A – количество строк матрицы, B – количество столбцов, mn – нижняя граница диапазона случайных чисел, mx – верхняя, n, k, m, s в зависимости от вышеуказанного задания.

Выходные данные: элементы матрицы C.

Этапы решения задачи:

1. Составить блок-схему;
2. Составить программу;
3. Провести проверку работы программы.

Блок-схема:

Основная программа (рисунок 1):

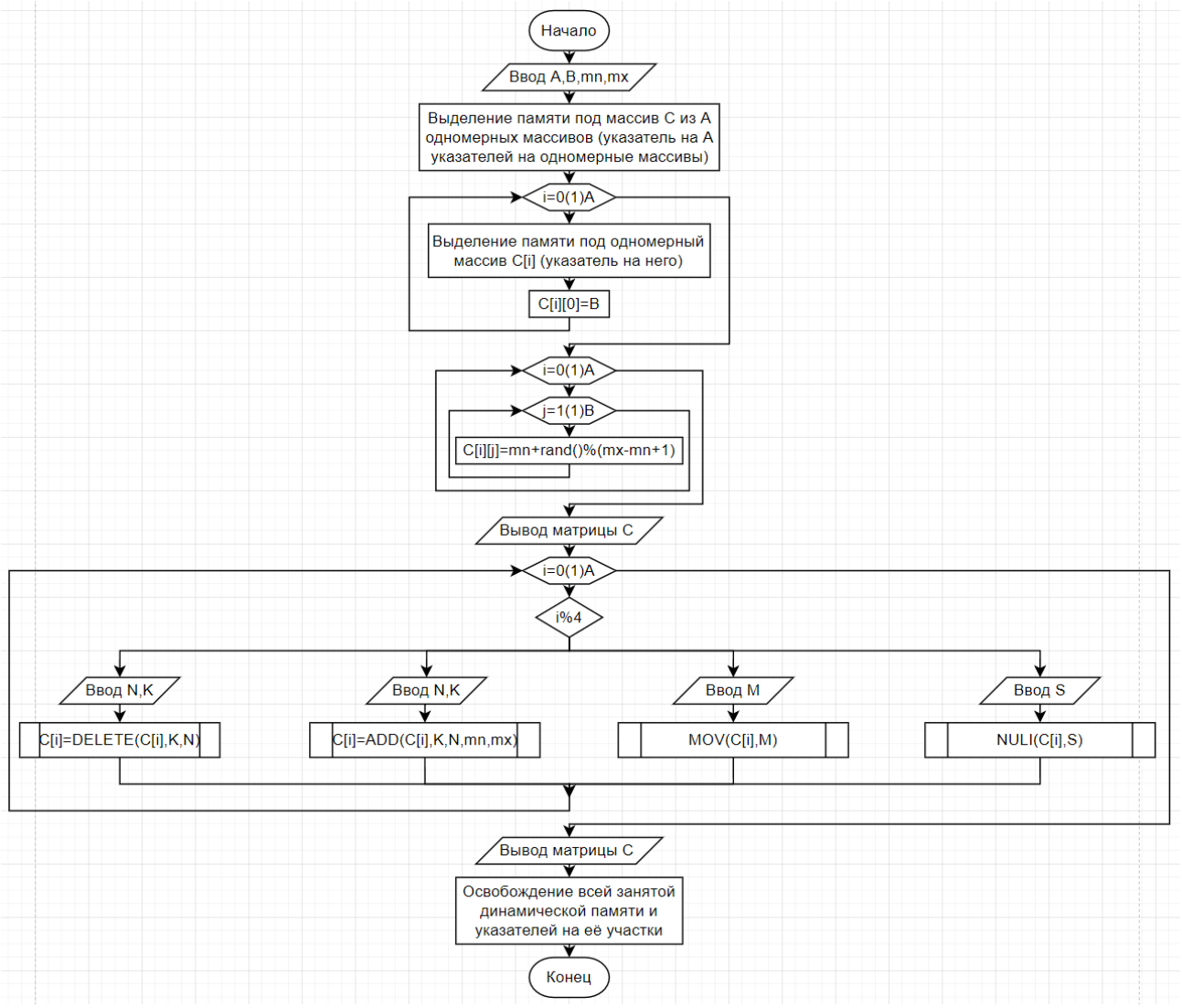

Рисунок 1

Подпрограмма удаления N элементов перед элементом с номером K (рисунок 2):

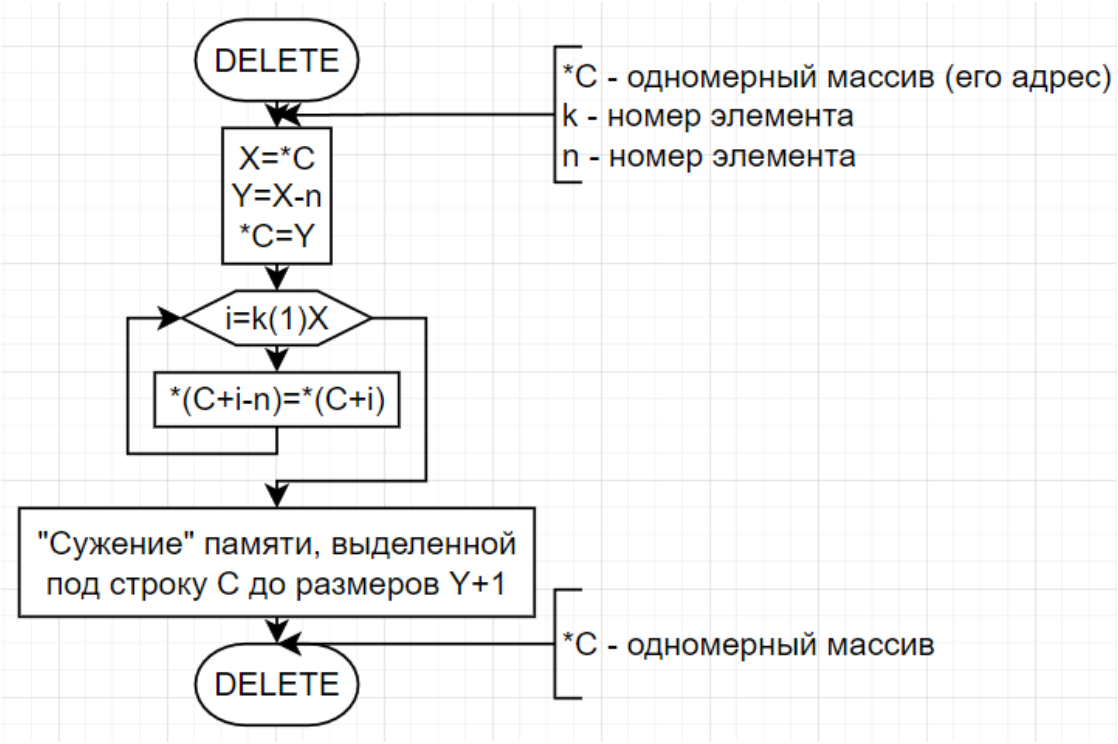

Рисунок 2

Подпрограмма добавления N случайных элементов перед элементом с номером K (рисунок 3):

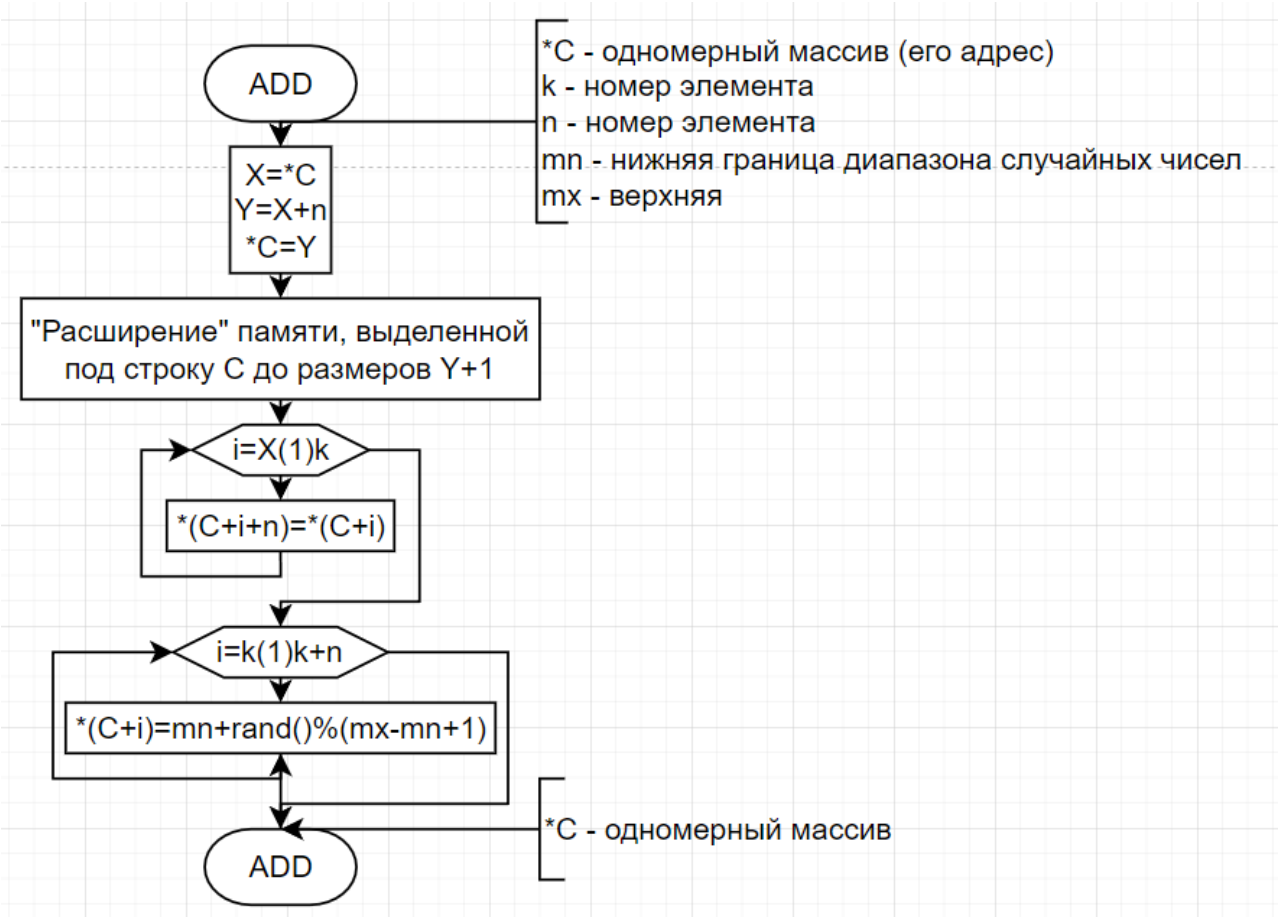

Рисунок 3

Подпрограмма циклического сдвига на M элементов вправо (рисунок 4):

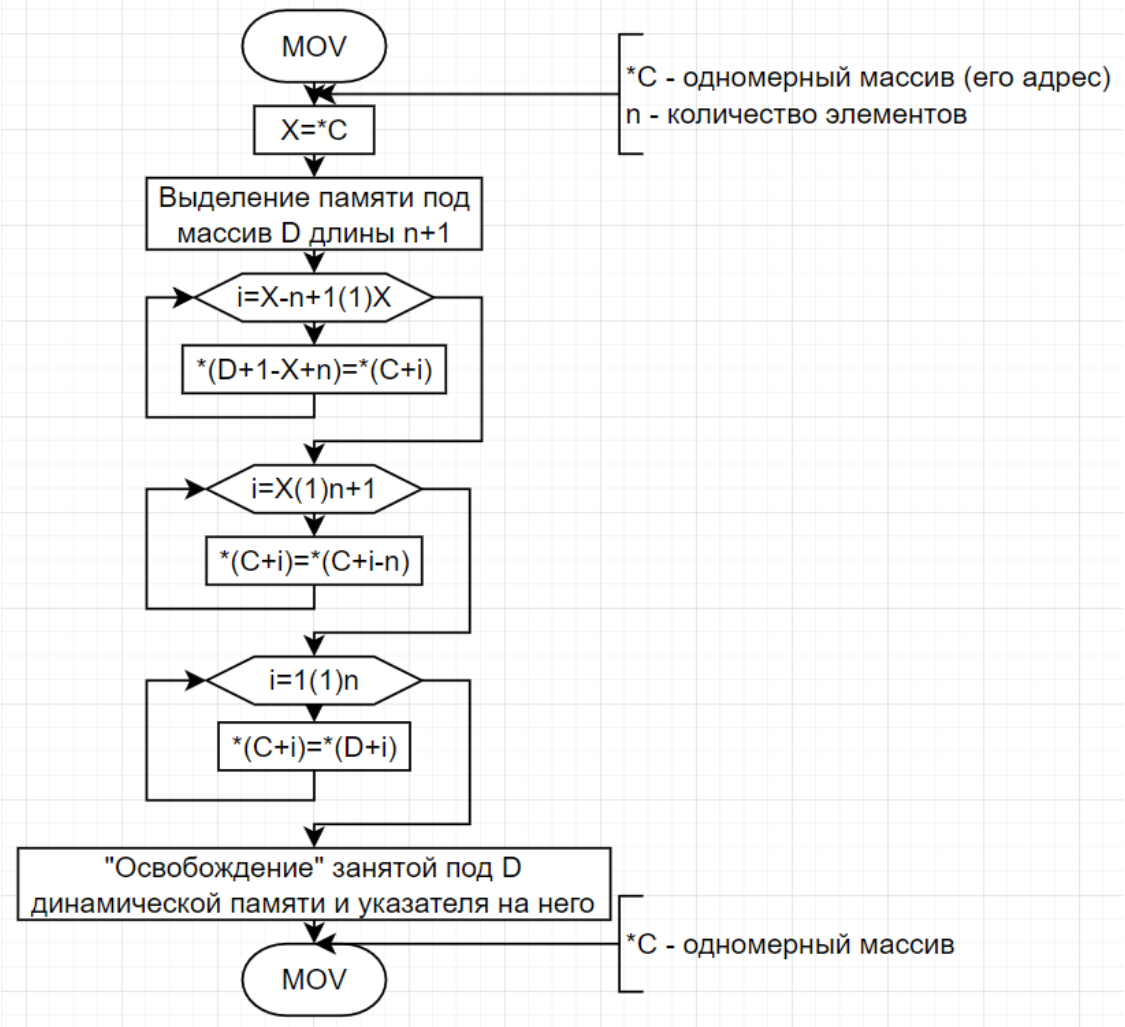

Рисунок 4

Подпрограмма зануления всех элементов с заданным значением S (рисунок 5):

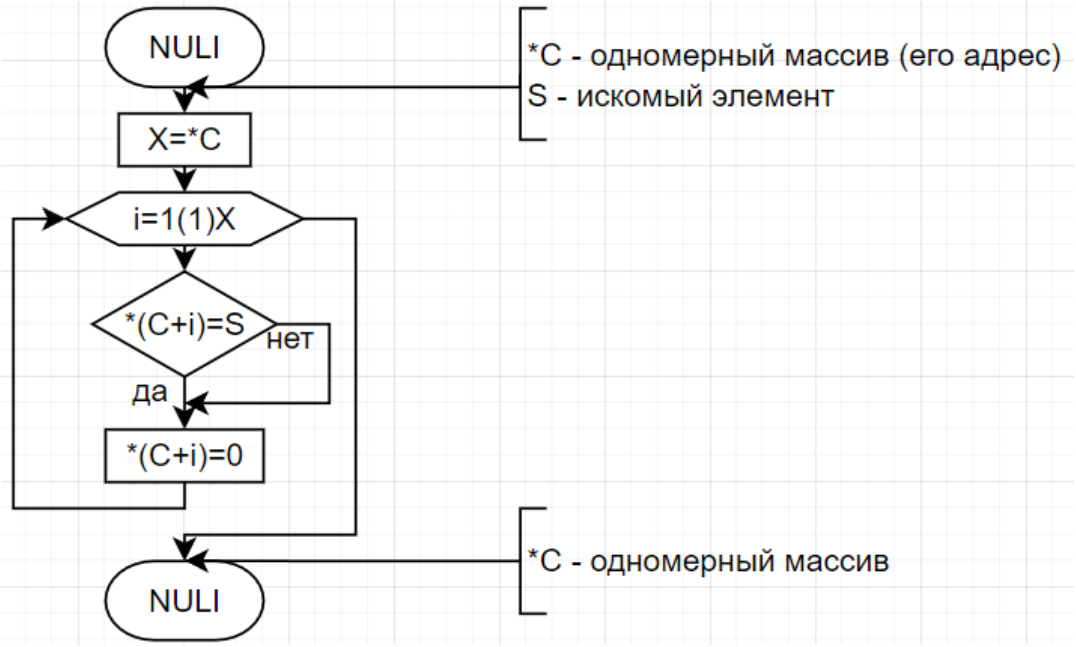

Рисунок 5

Листинг программы (рисунок 6, рисунок 7):

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
#include <mem.h>
int* DELETE(int* C, int k, int n) //УДАЛЕНИЕ N ЭЛЕМЕНТОВ ПЕРЕД ЭЛЕМЕНТОМ С НОМЕРОМ K
{
    int i;
    int X=*C;  //старый размер
    int Y=X-n; //новый размер
    *C=Y; //записываем новый размер в нулевую ячейку
    for (i=k;i<=X;i++) //"сдвигаем" поэлементно влево так, чтобы перекрыть весь удаляемый кусок
        *(C+i-n)=*(C+i);
    C = (int*)realloc(C,(Y+1) * sizeof(int)); //"отсекаем" ненужную правую часть массива
    return C;
}
int* ADD(int* C, int k, int n, int mn, int mx) //ДОБАВЛЕНИЕ N СЛУЧАЙНЫХ ЭЛЕМЕНТОВ ПЕРЕД ЭЛ.С НОМЕРОМ K
{
    int i;
    int X=*C;  //старый размер
    int Y=X+n; //новый размер
    *C=Y; //записываем новый размер в нулевую ячейку
    C = (int*)realloc(C,(Y+1) * sizeof(int)); //"растягиваем" на n элементов вправо
    for (i=X;i>=k;--i)  //сдвигаем поэлементно вправо так, чтобы освободить местечко для случайных элементов
        *(C+i+n)=*(C+i);
    for (i=k;i<k+n;i++) //в месте для случайных эл создаем случайные
        *(C+i)=mn+rand()%(mx-mn+1);
    return C;
}
int* MOV(int* C, int n) //ЦИКЛИЧЕСКИЙ СДВИГ НА N ЭЛЕМЕНТОВ ВПРАВО
{
    int i;
    int X=*C;
    int *D = (int*) malloc((n+1)*sizeof(int)); //выделяем массив для N ячеек у правого края
    if(!D) printf("Ошибка: недостаточно памяти"), exit(1);
    for (i=X-n+1;i<X;i++) //и сохраняем их
        *(D+i-X+n)=*(C+i);
    for (i=X;i>=n+1;i--)  //сдвигаем вправо до конца
        *(C+i)=*(C+i-n);
    for (i=1;i<=n;i++)    //сохранённый "хвостик" записываем в начало массива
        *(C+i)=*(D+i);
    free(D);     //не забываем убрать массив
    D=NULL;      //и указатель
}
int* NULI(int* C, int S) //ПОИСК И ЗАНУЛЕНИЕ ЭЛЕМЕНТОВ РАВНЫХ S
{
    int i;
    int X=*C;
    for (i=1;i<=X;i++)
        if (*(C+i)==S) *(C+i)=0;
}
int main() //ОСНОВНАЯ ПРОГРАММА
{
//ВВОД ИСХОДНЫХ ДАННЫХ: A,B,mn,mx
    system("chcp 1251");
    srand(time(NULL));
    int A,B,mn,mx;
    printf("Введите количество строк матрицы, A=");
    do {
        scanf("%d",&A);
        if (A<=0)
            printf("Ошибка: A должно быть больше нуля, A=");
        } while (A<=0);
    printf("Введите количество столбцов матрицы, B=");
    do {
        scanf("%d",&B);
        if (B<=0)
            printf("Ошибка: B должно быть больше нуля, B=");
        } while (B<=0);
    printf("Введите нижнюю границу диапазона случайных чисел mn, mn=");
    scanf("%d",&mn);
    printf("Введите верхнюю границу диапазона случайных чисел mx, mx=");
    do {
        scanf("%d",&mx);
        if (mx<=mn)
            printf("Ошибка: mx должно быть больше mn, mx=");
        } while (mx<=mn);
//СОЗДАНИЕ И ИНИЦИАЛИЗАЦИЯ ДИНАМИЧЕСКОГО МАССИВА
    int i,j;
    int **C = (int**) malloc(A * sizeof(int*));
    if(!C) printf("Ошибка: недостаточно памяти"), exit(1);
    for (i=0; i<A; i++)
    {
        C[i] = (int*) malloc((B+1) * sizeof(int));
        C[i][0]=B;
        if(!C[i]) printf("Ошибка: недостаточно памяти"), exit(1);
    }
    for (i=0;i<A;i++)
        for (j=1;j<=B;j++)
            C[i][j]=mn+rand()%(mx-mn+1);
    printf("\nИСХОДНАЯ МАТРИЦА:\n");
    for (i=0;i<A;i++)
    {
        for (j=1;j<=B;j++)
            printf("%*d\t", 4, C[i][j]);
        printf("\n");
    }
//ПОСЛЕДОВАТЕЛЬНОЕ ПРИМЕНЕНИЕ ФУНКЦИЙ К СТРОКАМ И ВЫВОД МАССИВА
```

Рисунок 6

```c
//ПОСЛЕДОВАТЕЛЬНОЕ ПРИМЕНЕНИЕ ФУНКЦИЙ К СТРОКАМ И ВЫВОД МАССИВА
    int N,M,K,S;
    for (i=0;i<A;i++)
    switch (i%4) {
        case 0: {
            //ввод N и K
            printf("\nУдаление N элементов перед элементом с номером K в строке №%d\n",i+1);
            printf("Введите номер элемента K, K=");
            do {
                scanf("%d",&K);
                if (K<=1)
                    printf("Ошибка: K должно быть больше единицы, K=");
                if (K>C[i][0])
                    printf("Ошибка: K не может быть больше количества элементов в строке, K=");
                } while ((K<=1)||(K>C[i][0]));
            printf("Введите количество элементов N, N=");
            do {
                scanf("%d",&N);
                if (N<=0)
                    printf("Ошибка: N должно быть больше нуля, N=");
                if (N>=K)
                    printf("Ошибка: N должно быть меньше K, N=");
            } while ((N<=0) || (N>=K));
             C[i]=DELETE(C[i],K,N); //функция удаления
            break;};
        case 1: {
            //ввод N и K
            printf("\nДобавление N случайных элементов перед элементом с номером K в строке №%d\n",i+1);
            printf("Введите номер элемента K, K=");
            do {
                scanf("%d",&K);
                if (K<=0)
                    printf("Ошибка: K должно быть больше нуля, K=");
                if (K>C[i][0])
                    printf("Ошибка: K не может быть больше количества элементов в строке, K=");
                } while ((K<=0)||(K>C[i][0]));
            printf("Введите количество элементов N, N=");
            do {
                scanf("%d",&N);
                if (N<=0)
                    printf("Ошибка: N должно быть больше нуля, N=");
            } while (N<=0);
            C[i]=ADD(C[i],K,N,mn,mx); //функция добавления
            break;}
        case 2: {
            //ввод M
            printf("\nЦиклический сдвиг на M элементов вправо в строке №%d\n",i+1);
            printf("Введите количество элементов M, M=");
            do {
                scanf("%d",&M);
                if (M<=0)
                    printf("Ошибка: M должно быть больше нуля, M=");
                if (M>=C[i][0])
                    printf("Ошибка: M должно быть меньше длины строки, M=");
                } while ((M<=0)||(M>=C[i][0]));
            MOV(C[i],M); //функция перестановки
            break;}
        case 3: {
            //ввод S
            printf("\nЗануление всех элементов с заданным значением в строке №%d\n",i+1);
            printf("Введите значение, которое будет обнуляться, S=");
            scanf("%d",&S);
            NULI(C[i],S); //функция поиска
            break;}
    }
    printf("\nПОЛУЧЕННАЯ МАТРИЦА:\n");
    for (i=0;i<A;i++)
    {
        for (j=1;j<=*C[i];j++)
            printf("%*d\t", 6, C[i][j]);
        printf("\n");
    }
    for (i = 0; i<A; ++i)
        free(C[i]);
    free(C);
    C = NULL;
    return 0;
}
```

Рисунок 7

Результаты работы программы и проверка (рисунок 8):

Программа была запущена для изображенных на рисунке входных значений. Все функции (по удалению, добавлению, сдвигу и поиску) работают корректно, программа свои функции выполняет и справляется с поставленной задачей.

```
Текущая кодовая страница: 1251
Введите количество строк матрицы, A=4
Введите количество столбцов матрицы, B=10
Введите нижнюю границу диапазона случайных чисел mn, mn=-5
Введите верхнюю границу диапазона случайных чисел mx, mx=5

ИСХОДНАЯ МАТРИЦА:
  -5       3      -1      -1      -3      -1      -2      -1      -1      -5
   1      -5       4       0       2      -5      -5      -4       3      -3
   0       4       4       4      -3       4       4       1      -1       0
  -2       1       5      -2      -3       4      -2       1      -5       1

Удаление N элементов перед элементом с номером K в строке №1
Введите номер элемента K, K=10
Введите количество элементов N, N=9

Добавление N случайных элементов перед элементом с номером K в строке №2
Введите номер элемента K, K=1
Введите количество элементов N, N=4

Циклический сдвиг на M элементов вправо в строке №3
Введите количество элементов M, M=3

Зануление всех элементов с заданным значением в строке №4
Введите значение, которое будет обнуляться, S=1

ПОЛУЧЕННАЯ МАТРИЦА:
    -5
    -2       5       4       3       1      -5       4       0       2      -5      -5      -4       3      -3
     1      -1       0       0       4       4       4      -3       4       4
    -2       0       5      -2      -3       4      -2       0      -5       0

--------------------------------
Process exited after 25.21 seconds with return value 0
Для продолжения нажмите любую клавишу . . .
```

Рисунок 8