**Лабораторная работа №5. Массивы.**

**Часть I.**

**Задача 1.**

Постановка задачи:

Для некоторого числового вектора X, введённого с клавиатуры, вычислить значения вектора Y = X · X (yi = xi · xi — поэлементно).

Математическая модель: представлена в условии задачи

Список идентификаторов:

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Смысл |
| n | const | Константа |
| x | int array | Набор векторов x[i] |
| y | int array | Вычисляемые векторы y[i] |

Код программы:

#include <stdio.h>

#include <math.h>

#define n 10

int main(void) {

int x[n], y[n]={0};

printf("Введите 10 чисел \n");

for(int i = 0; i < n; i++) {

printf("x[%d] = ", i);

scanf("%d", &x[i]);

}

for(int i = 0; i < n; i++) {

y[i] = x[i] \* x[i];

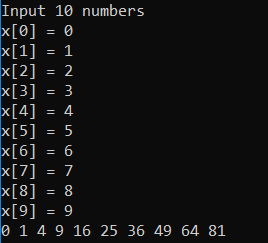
printf("%d ", y[i]);

}

return 0;

}

Результат:



**Задача 2.**

Постановка задачи:

Для некоторого числового массива X, введённого с клавиатуры, изменить порядок элементов на обратный без привлечения вспомогательного массива и со вспомогательным массивом.

Математическая модель:

С использованием вспомогат. массива: B[4] = X[0], B[3] = X[1] … B[0] = X[4]

Без использования вспомогат. массива: X[i] = X[4 - i]

Список идентификаторов:

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Смысл |
| N | const | Константа (размер массива) |
| X | int array | Заданный массив |
| B | int array | Вспомогательный массив |
| i | int | Параметр цикла |
| k | int | Вспомогательная переменная |
| t | int | Вспомогательная переменная |

Код программы:

#include <stdio.h>

#define N 5

int main(void) {

int X[N];

int B[N];

for (int i = 0; i <= N - 1; i++) { // Ввод массива

printf("A[%d] = ", i);

scanf("%d", &X[i]);

}

int k = N - 1; // С использованием вспомогательного массива

for (int i = 0; i <= N - 1; i++) {

B[k] = X[i];

k--;

}

int t; // Без использованием вспомогательного массива

for (int i = 0; i <= (N - 1) / 2; i++) {

t = X[i];

X[i] = X[4-i];

X[4-i] = t;

}

printf("--------------\n"); // Вывод отсортированного массива

for (int i = 0; i <= N - 1; i++) { // без использования вспомогательного

printf("A[%d] = %d\n", i, X[i]);

}

printf("--------------\n"); // Вывод отсортированного массива

for (int i = 0; i <= N - 1; i++) { // с использованием вспомогательного

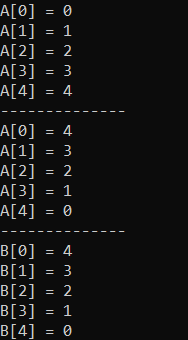
printf("B[%d] = %d\n", i, B[i]);

}

return 0;

}

Результат:



**Задача 3.**

Постановка задачи:

Реализовать различные варианты алгоритма сортировки пузырьком, организовав проходы алгоритма с начала, и с конца массива, а также с двумя противоположными условиями сравнения. В качестве элементов сортировки использовать произвольные массивы чисел. Каждый найденный возможный вариант алгоритма должен приводить к некоторому осмысленному результату сортировки.

Математическая модель:

if a[j] > a[j-1] than swap(a[j-1], a[j])

Список идентификаторов:

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Смысл |
| N | const | Константа (размер массива) |
| A | int array | Массив |
| i | int | Параметр цикла |
| j | int | Параметр цикла |

Код программы:

#include <stdio.h>

#define N 5

void swap(int \*p\_i, int \*p\_j) { // процедура для

int t = \*p\_i; // обмена значениями

\*p\_i = \*p\_j; // двух значений

\*p\_j = t; // массива

}

void outputArray(int array\_in[N]) { // процедура вывода массива

for (int i = 0; i <= N - 1; i++) {

printf("A[%d] = %d\n", i, array\_in[i]);

}

}

void sortLeftIncrease(int array[N]) { // процедура сортировки

for (int i = N - 1; i >= 0; i--) // по возврастанию

for (int j = 0; j < i; j++) { // с левой стороны

if (array[j] > array[j+1])

swap(&array[j], &array[j+1]);

}

outputArray(array);

}

void sortLeftDecrease(int array[N]) { // процедура сортировки

for (int i = N - 1; i >= 0; i--) // по убыванию

for (int j = 0; j < i; j++) { // с левой стороны

if (array[j] < array[j+1])

swap(&array[j], &array[j+1]);

}

outputArray(array);

}

void sortRightIncrease(int array[N]) { // процедура сортировки

for (int i = N - 1; i >= 1; i--) // по возврастанию

for (int j = N - 1; j >= N - i; j--) { // с правой стороны

if (array[j-1] > array[j])

swap(&array[j-1], &array[j]);

}

outputArray(array);

}

void sortRightDecrease(int array[N]) { // процедура сортировки

for (int i = N - 1; i >= 1; i--) // по возврастанию

for (int j = N - 1; j >= N - i; j--) { // с правой стороны

if (array[j-1] < array[j])

swap(&array[j-1], &array[j]);

}

outputArray(array);

}

int main(void) {

int A[N] = {4, 5, 2, 1, 3};

printf("Source array:\n");

outputArray(A);

printf("Sorted array by increase left:\n");

sortLeftIncrease(A);

printf("Sorted array by decrease left:\n");

sortLeftDecrease(A);

printf("Sorted array by increase right:\n");

sortRightIncrease(A);

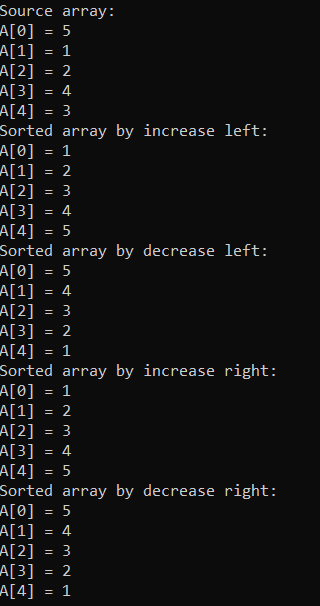
printf("Sorted array by decrease right:\n");

sortRightDecrease(A);

return 0;

}

Результат работы:



**Задача 4.**

Постановка задачи:

Реализовать самостоятельно алгоритм сортировки вставками и сравнить его реализацию с полученными реализациями «алгоритма пузырька» в задаче 5.3.

Математическая модель:

i = j - 1;

while ((i >= 0) && (A[i] > A[i + 1]))

swap(A[i], A[i+1])

i--;

Список идентификаторов:

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Смысл |
| N | const | Константа (размер массива) |
| A | int array | Массив |
| i | int | Параметр цикла |
| j | int | Параметр цикла |

Код программы:

#include <stdio.h>

#define N 5

int main(void) {

int A[N], j, i;

for (i = 0; i <= N - 1; i++) { // Ввод массива

printf("A[%d] = ", i);

scanf("%d", &A[i]);

}

for (j = 1; j <= N - 1; j++) { // Сортировка массива

i = j - 1;

while ((i >= 0) && (A[i] > A[i + 1])) {

int t = A[i];

A[i] = A[i + 1];

A[i + 1] = t;

i--;

}

}

printf("--------------\n");

for (int i = 0; i <= N - 1; i++) { // Вывод отсортированного массива

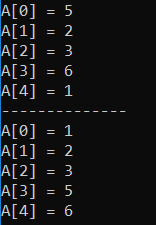
printf("A[%d] = %d\n", i, A[i]);

}

return 0;

}

Результат программы:



**Задача 5**

Постановка задачи:

Организовать ввод массива (матрицы) по столбцам.

Математическая модель:

for (int j = 0; j <= N - 1; j++)

for (int i = 0; i <= N - 1; i++)

read(A[i, j])

Список идентификаторов:

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Смысл |
| N | const | Константа (размер массива) |
| A | int array | Массив |
| i | int | Параметр цикла |
| j | int | Параметр цикла |

Код программы:

#include <stdio.h>

#define N 3

int main(void) {

int A[N][N];

for (int j = 0; j <= N - 1; j++) // Ввод массива

for (int i = 0; i <= N - 1; i++) {

printf("A[%d, %d] = ", i, j);

scanf("%d", &A[i][j]);

}

printf("--------------\n");

for (int i = 0; i <= N - 1; i++)

for (int j = 0; j <= N - 1; j++) {

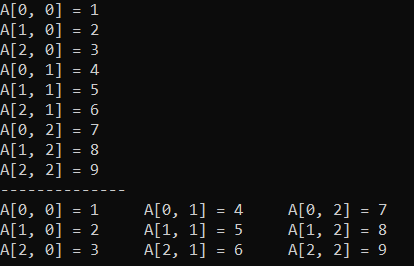
printf("A[%d, %d] = %d\n", i, j, A[i][j]);

}

return 0;

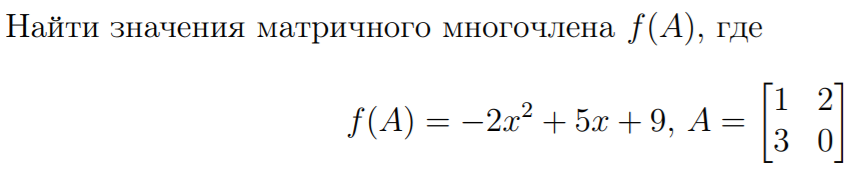
}

Результат:



**Задача 6**

Постановка задачи:



Математическая модель: представлена в условии задачи

Список идентификаторов:

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Смысл |
| N | const | Размер массива |
| A | int array | Массив |
| f | int array | Результат |
| i | int | Параметр цикла |
| j | int | Параметр цикла |
| r | int | Параметр цикла |
| t | int | Вспомогательная переменная |

Код программы:

#include <stdio.h>

#define N 2

int main(void) {

int A[N][N] = {1, 2, 3, 0};

int f[N][N];

for (int i = 0; i <= N - 1; i++) // Вычисление матрицы А в квадрате

for (int j = 0; j <= N - 1; j++) { // результат записывается в новый массив F

int t = 0;

for (int r = 0; r <= 1; r++) {

t += A[i][r] \* A[r][j];

}

f[i][j] = t;

}

for (int i = 0; i <= N - 1; i++) // Умножение матрицы F на -2

for (int j = 0; j <= N - 1; j++) { // и прибавление 5 \* A и 9

f[i][j] = - 2 \* f[i][j] + 5 \* A[i][j]; // где А - изначальная матрица

if (i == j)

f[i][j] += 9;

}

printf("--------------\n");

for (int i = 0; i <= N - 1; i++) {

for (int j = 0; j <= N - 1; j ++)

printf("F[%d, %d] = %d\t", i, j, f[i][j]);

printf("\n");

}

return 0;

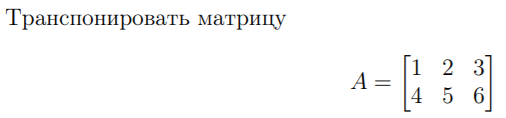
}

Результат:

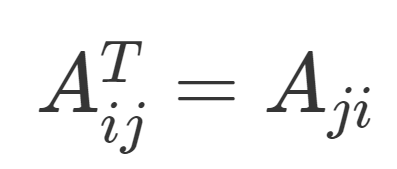


**Задача 7**

Постановка задачи:



Математическая модель:



Список идентификаторов:

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Смысл |
| A | int array | Исходная матрица |
| At | int array | Транспонированная матрица |
| i | int | Параметр цикла |
| j | int | Параметр цикла |

Код программы:

#include <stdio.h>

int main(void) {

int A[2][3] = { 1, 2, 3, 4, 5, 6 };

int At[3][2];

for (int i = 0; i <= 2; i++)

for (int j = 0; j <= 1; j++)

At[i][j] = A[j][i];

for (int i = 0; i <= 2; i++) {

for (int j = 0; j <= 1; j++)

printf("At[%d][%d] = %d\t", i, j, At[i][j]);

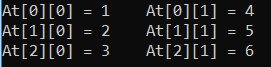
printf("\n");

}

return 0;

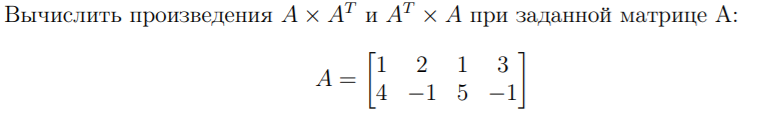
}

Результат:

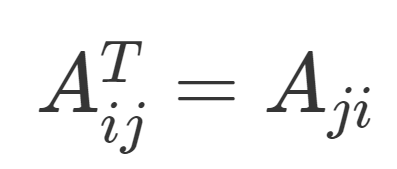


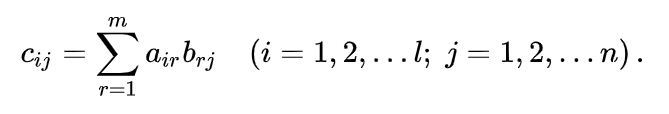
**Задача 8**

Постановка задачи:



Математическая модель:





Список идентификаторов:

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Смысл |
| A | int array | Матрица А |
| At | int array | Транспонированная матрица AT |
| A\_At | int array | A\*AT |
| At\_A | int array | AT\*A |
| i | int | Параметр цикла |
| j | int | Параметр цикла |
| r | int | Параметр цикла |

Код программы:

#include <stdio.h>

int main(void) {

int A\_At[2][2] = {0};

int At\_A[4][4] = {0};

int A[2][4] = { 1, 2, 1, 3, 4, -1, 5, -1 };

int At[4][2];

for (int i = 0; i <= 3; i++) // Транспонирование матрицs

for (int j = 0; j <= 1; j++)

At[i][j] = A[j][i];

for (int i = 0; i <= 1; i++)

for (int j = 0; j <= 1; j++)

for (int r = 0; r <= 3; r++)

A\_At[i][j] += (A[i][r] \* At[r][j]);

for (int i = 0; i <= 1; i++) {

for (int j = 0; j <= 1; j++)

printf("A \* At[%d][%d] = %d\t", i, j, A\_At[i][j]);

printf("\n");

}

for (int i = 0; i <= 3; i++)

for (int j = 0; j <= 3; j++)

for (int r = 0; r <= 1; r++)

At\_A[i][j] += (At[i][r] \* A[r][j]);

printf("-----------------------------------------------------\n");

for (int i = 0; i <= 3; i++) {

for (int j = 0; j <= 3; j++)

printf("At \* A[%d][%d] = %d\t", i, j, At\_A[i][j]);

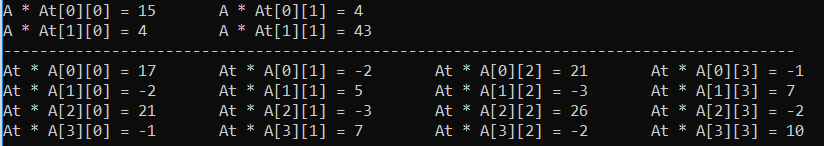
printf("\n");

}

return 0;

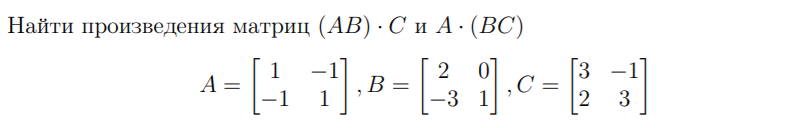
}

Результат:

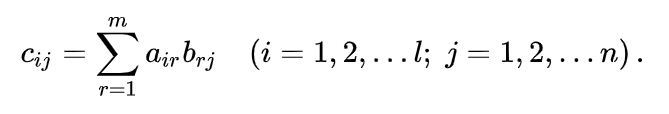


**Задача 9**

Постановка задачи:



Математическая модель:



Список идентификаторов:

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Смысл |
| A | int array | Матрица А |
| B | int array | Матрица В |
| C | int array | Матрица С |
| AB | int array | A \* B |
| BC | int array | B \* C |
| res | int | Вспомогательная переменная |
| i | int | Параметр цикла |
| j | int | Параметр цикла |
| r | int | Параметр цикла |
| n | const | Размер массива |

Код программы:

#include <stdio.h>

#define n 2

int main(void) {

int A[n][n] = {1, -1, -1, 1};

int B[n][n] = {2, 0, -3, 1};

int C[n][n] = {3, -1, 2, 3};

int AB[n][n] = {0}; // В эту матрицу будет записываться произведение А \* В

int BC[n][n] = {0}; // А в эту В \* С

int res = 0; // Результативная матрица

for (int i = 0; i <= n-1; i++) // В этом цикле

for (int j = 0; j <= n-1; j++) // вычисляются

for (int r = 0; r <= n-1; r++) { // произведения

AB[i][j] += A[i][r] \* B[r][j]; // А \* В

BC[i][j] += B[i][r] \* C[r][j]; // и В \* С

}

printf("(A \* B) \* C:\n");

for (int i = 0; i <= n-1; i++) { // В этом цикле

for (int j = 0; j <= n-1; j++) { // вычисляется

res = 0;

for (int r = 0; r <= n-1; r++) // произведение

res += AB[i][r] \* C[r][j]; // (A\*B) \* C

printf("R[%d][%d] = %d\t", i, j, res);

}

printf("\n");

}

printf("----------------------\n");

printf("A \* (B \* C):\n");

for (int i = 0; i <= n-1; i++) { // В этом цикле

for (int j = 0; j <= n-1; j++) { // вычисляется

res = 0;

for (int r = 0; r <= n-1; r++) // произведение

res += A[i][r] \* BC[r][j]; // A \* (B\*C)

printf("R[%d][%d] = %d\t", i, j, res);

}

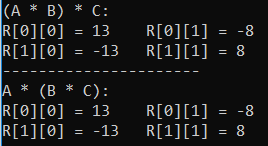
printf("\n");

}

return 0;

}

Результат:



**Задача 10**

Постановка задачи:

Преобразовать исходную матрицу так, чтобы первый элемент каждой строки был заменён средним арифметическим элементов этой строки.

Математическая модель:

Список идентификаторов:

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Смысл |
| n | const | Кол-во строк |
| m | const | Кол-во столбцов |
| s | int | Сумма элементов в строке |
| matrix | int array | Матрица |
| i | int | Параметр цикла |
| j | int | Параметр цикла |

Код программы:

#include <stdio.h>

#include <stdlib.h>

#define n 3

#define m 4

int main(void) {

int s = 0;

float matrix[n][m];

for (int i = 0; i < n; i++)

for (int j = 0; j < m; j ++) {

printf("M[%d][%d] = ", i, j);

scanf("%f", &matrix[i][j]);

}

for (int i = 0; i < n; i++) {

for (int j = 0; j < m; j ++)

s += matrix[i][j];

matrix[i][0] = 1. \* s / m;

s = 0;

}

for (int i = 0; i < n; i++) {

for (int j = 0; j < m; j ++)

printf("M[%d][%d] = %.1f\t", i, j, matrix[i][j]);

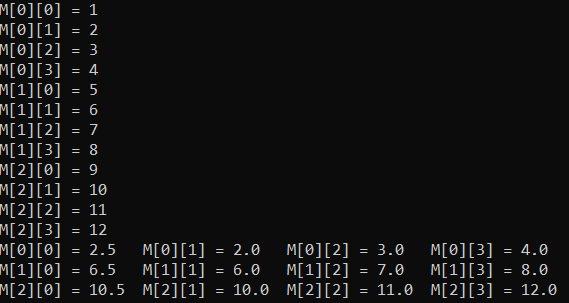
printf("\n");

}

return 0;

}

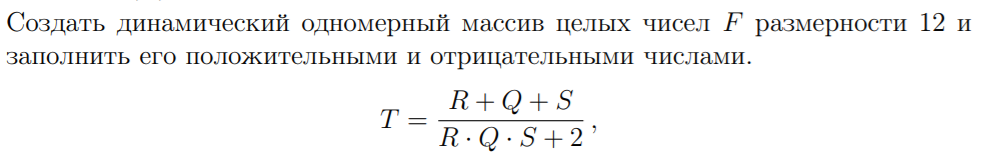
Результат:



**Часть II**

**Задача 1**

Постановка задачи:

****

Математическая модель: представлена в условии задачи

Список идентификаторов:

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Смысл |
| n | const | Размер массива |
| R | int | Сумма отриц. элементов |
| Q | int | Отрицательный элемент |
| S | int | Произведение полож. элементов |
| T | float | Результат |
| F | int array | Массив |
| i | int | Параметр цикла |

Код программы:

#include <stdio.h>

#include <stdlib.h>

#define n 12

int main(void) {

int R = 0, Q = 0, S = 1;

float T;

int \*F = (int \*)malloc(n \* sizeof(int));

for (int i = 0; i < n; i++) {

printf("F[%d] = ", i+1);

scanf("%d", &F[i]);

if (F[i] < 0)

R += F[i];

if (F[i] > 0)

S \*= F[i];

}

for (int i = 0; i < n; i++) {

if (F[i] < 0) {

Q = F[i];

T = 1. \* (R + Q + S) / (R \* Q \* S + 2);

printf("Q = %d, T = %f\n", Q, T);

}

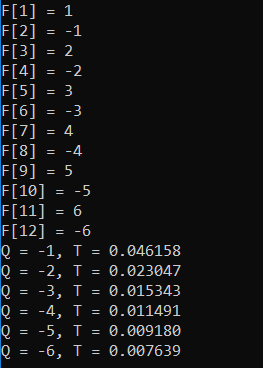
}

free(F);

return 0;

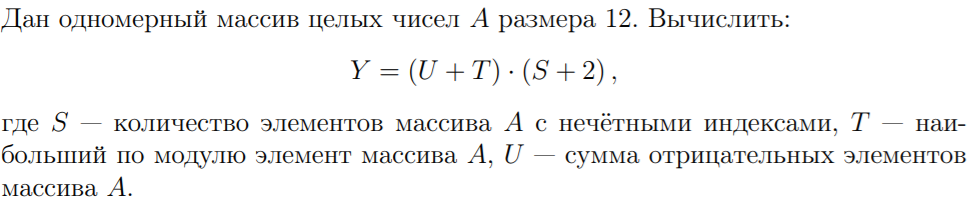
}

Результат:



**Задача 2**

Постановка задачи:

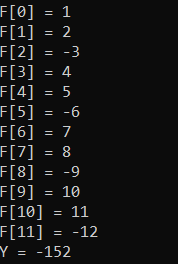


Математическая модель: представлена в условии задачи

Список идентификаторов:

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Смысл |
| n | const | Размер массива |
| S | int | Кол-во эл-ов с нечёт. индексами |
| U | int | Сумма отриц. элементов |
| T | int | Наиб. по модулю элемент массива |
| A | int array | Массив |
| i | int | Параметр цикла |
| Y | int | Результат |

Результат:



**Задача 3**

Постановка задачи:

Напишите программу для вычисления пересечения двух конечных множеств (наборов) A и B целых чисел одинакового размера с использованием динамических массивов. В качестве множества A можно взять первые 12 чисел ряда Фибоначчи: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144. В качестве множества B можно взять первые 12 чисел последовательности Падована: 1, 1, 1, 2, 2, 3, 4, 5, 7, 9, 12, 16. Дублирующиеся значения можно исключать.

Математическая модель:

Если fibb(a) == padovan(b) тогда вывод a

Список идентификаторов:

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Смысл |
| A | int array | Множество A |
| B | int array | Множество B |
| temp | int | Вспомогательная переменная |
| i | int | Параметр цикла |
| j | int | Параметр цикла |
| n | const | Размер множеств |

Код программы:

#include <stdio.h>

#include <stdlib.h>

#define n 12

int fibb(int a) { // Для заполнения

if (a == 1 || a == 2) // массива А

a = 1; // числами ряда Фибоначчи

else

a = fibb(a - 1) + fibb(a - 2);

return a;

}

int padovan(int b) { // Для заполнения

if (b == 1 || b == 2 || b == 3) // массива В

b = 1; // числами ряда Падована

else

b = padovan(b - 2) + padovan(b - 3);

return b;

}

int main(void) {

int \*A;

int \*B;

int temp = 0;

A = (int\*) malloc(n \* sizeof(int));

B = (int\*) malloc(n \* sizeof(int));

for (int i = 0; i < n; i++) {

A[i] = fibb(i+1);

printf("%d\t", A[i]);

}

printf("\n");

for (int i = 0; i < n; i++) {

B[i] = padovan(i+1);

printf("%d\t", B[i]);

}

printf("\n");

for (int i = 0; i < n; i++) // Если среди элементов A[i], i = [0; n)

for (int j = 0; j < n; j++) // Находится элемент B[j], j = [0; n)

if (A[i] == B[j] && A[i] != temp) { // И он не равен temp

temp = A[i]; // где temp предыдущее число

printf("%d\t", temp); // temp присваивается это самое число

}

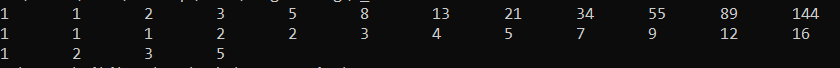
free(A);

free(B);

return 0;

}

Результат:



**Задача 4**

Постановка задачи:

Выделить динамически память под некоторую матрицу A размерности M × N и заполнить её произвольными числами. Сократить правильно размер этой матрицы, удалив из неё одну выбранную строку, освободив от неё также и память, используя указатели. После удаления строки в матрице AM×N должна быть возможность обхода всех элементов «новой» матрицы AM−1×N таким же способом, что и изначальной матрицы. Распечатать (используя циклы) матрицу до удаления строки и после удаления.

Математическая модель:

---

Список идентификаторов:

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Смысл |
| A | int array | Исходный массив |
| B | int array | Результативный массив |
| m | int | Кол-во столбцов в массиве А |
| n | int | Кол-во строк в массиве А |
| d | int | Номер строки которую нужно удалить |
| i | int | Параметр цикла |
| j | int | Параметр цикла |

Код программы:

#include <stdio.h>

#include <stdlib.h>

int main(void) {

int \*\*A = NULL;

int m, n, d;

printf("n = ");

scanf("%d", &n);

printf("m = ");

scanf("%d", &m);

A = (int\*\*) malloc(n \* sizeof(int \*));

// Формируем и заполняем массив А //

for (int i = 0; i < n; i ++)

A[i] = (int\*)malloc(m \* sizeof(int));

for (int i = 0; i < n; i++) {

for (int j = 0; j < m; j++) {

printf("A[%d][%d] = ", i, j);

scanf("%d", &A[i][j]);

}

printf("\n");

}

// Принимаем номер строки, которую нужно удалить //

printf("What a row are you want to delete?\nThe account starts from 0\n");

scanf("%d", &d);

// Формируем новый массив В и заполняем его //

int \*\*B = NULL;

B = (int\*\*) malloc((n-1) \* sizeof(int \*));

for (int i = 0; i < n-1; i ++)

B[i] = (int\*)malloc(m \* sizeof(int));

for (int i = 0; i < n-1; i++) { // Пока мы не дошли до строки

for (int j = 0; j < m; j++) { // которую нужно удалить

if (i < d) // присваиваем элементу B[i][j] элемент A[i][j]

B[i][j] = A[i][j]; // а когда мы дойдём до нужной строки

else // элементу B[i][j] будем присваивать элемент

B[i][j] = A[i+1][j]; // A[i+1][j], т.е. строку которую нужно "удалить"

printf("B[%d][%d] = %d\t", i, j, B[i][j]); // мы будем пропускать

} // таким образом в новом массиве будет на одну строку меньше

printf("\n"); // при этом будет пропущена строка которую нужно было "удалить"

}

// Освобождение памяти

for (int i = 0; i < n; i++)

free(A[i]);

for (int i = 0; i < n-1; i++)

free(B[i]);

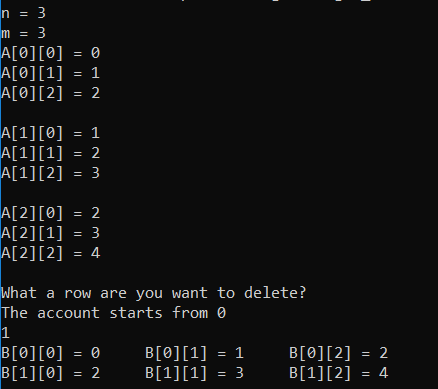
free(A);

free(B);

return 0;

}

Результат:



**Задача 5**

Постановка задачи:

Написать программу, которая вычисляет некоторый вектор b, как результат умножения некоторой матрицы M на вектор a:

b = M × a .

Математическая модель:

b[i] =

Список идентификаторов:

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Смысл |
| i | int | Параметр цикла |
| j | int | Параметр цикла |
| n | int | Кол-во строк в матрице |
| m | int | Кол-во столбцов в матрице |
| b | int | Искомый вектор-столбец |
| a | int | Заданный вектор-столбец |
| matrix | int | Матрица |
| m\_columns | int | Кол-во |

Код:

#include <stdio.h>

#include <stdlib.h>

int main(void) {

int n, m, \*b, \*a, \*\*matrix, n\_rows;

// Читаем размер вектора а, выделяем память и считываем значения

printf("Size of vector a = ");

scanf("%d", &n\_rows);

a = (int \*) malloc(n\_rows \* sizeof(int));

for (int j = 0; j < n\_rows; j++) {

printf("a[0][%d] = ", j);

scanf("%d", &a[j]);

}

//Узнаём размеры матрицы

printf("Size of matrix: ");

printf("n = ");

scanf("%d", &n);

printf("m = ");

scanf("%d", &m);

/\* Если кол-во столбцов в матрице не совпадает с кол-вом

строк в вектор-столбце, тогда умножение невозможно \*/

if (m != n\_rows) {

printf("Umnozjenie nevozmozjno");

return 1;

}

//Выделение памяти под матрицу и её заполнение

matrix = (int \*\*) malloc(n \* sizeof(int \*));

for (int i = 0; i < n; i++)

matrix[i] = (int \*) malloc(m \* sizeof(int));

for (int i = 0; i < n; i++)

for (int j = 0; j < m; j++) {

printf("M[%d][%d] = ", i, j);

scanf("%d", &matrix[i][j]);

}

//Выделение памяти под вектор b и дальнейшее его вычисление

b = (int \*) calloc(n, sizeof(int));

for (int i = 0; i < n; i++) {

for (int j = 0; j < m; j++)

b[i] += a[j] \* matrix[i][j];

printf("b[%d] = %d\n", i, b[i]);

}

// Освобождение памяти

free(b);

free(a);

for (int i = 0; i < n; i++)

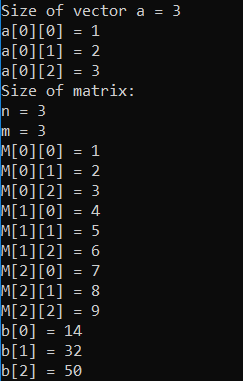
free(matrix[i]);

free(matrix);

return 0;

}

Результат работы:



**Задача 6**

Постановка задачи:

Напишите программу, в которой создаётся квадратная матрица, заполненная нулями и единицами. Единичные значения у тех элементов, для которых сумма индексов является нечётным числом. Нулевые значения у тех элементов, для которых сумма индексов является чётным числом.

Математическая модель:

Если i+j mod 2 = 0 тогда A[i][j] = 0

Иначе A[i][j] = 1

Список идентификаторов:

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Смысл |
| n | const | Размер матрицы |
| A | int array | Массив |
| i | int | Параметр цикла |
| j | int | Параметр цикла |

Код программы:

#include <stdio.h>

#include <stdlib.h>

int main(void) {

int \*\*A = NULL;

int n;

printf("n = ");

scanf("%d", &n);

// Выделение памяти под матрицу А

A = (int\*\*) malloc(n \* sizeof(int \*));

for (int i = 0; i < n; i ++)

A[i] = (int\*)malloc(n \* sizeof(int));

// Заполнение матрицы

for (int i = 0; i < n; i++) {

for (int j = 0; j < n; j++) {

if((i + j) % 2 == 0) // Если сумма индексов чётна

A[i][j] = 0; // тогда элемент равен 0

else

A[i][j] = 1; // иначе 1

printf("A[%d][%d] = %d\t", i, j, A[i][j]);

}

printf("\n");

}

// Освобождение памяти

for (int i = 0; i < n; i ++)

free(A[i]);

free(A);

return 0;

}

Результат:

