Лабораторная работа №6

Часть I

**Задача 1**

Постановка задачи:



Математическая модель: представлена в условии задачи

Список идентификаторов:

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Смысл |
| pointer | Int \*\* | Указатель |

Код программы:

#include <stdio.h>

#include <stdlib.h>

int main() {

    double \*\*pointer = NULL;

    pointer = (double \*\*) malloc(sizeof(double \*));

    pointer[0] = (double \*) malloc(sizeof(double));

    \*pointer[0] = 2;

    printf("%f", \*pointer[0]);

    free(pointer);

    return 0;

}

Результат:



**Задача 2**

Постановка задачи: Напишите программу, которая складывает два числа с использованием указателей на эти числа.

Математическая модель:

Если \*p\_a = &a, \*p\_b = &b

Тогда: a + b = \*p\_a + \*p\_b

Список идентификаторов:

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Смысл |
| a | int | Число 1 |
| b | int | Число 2 |
| p\_a | int \* | Указатель на число 1 |
| p\_b | int \* | Указатель на число 2 |

Код программы:

#include <stdio.h>

#include <stdlib.h>

int main(void) {

    int a, b;

    scanf("%d\n%d", &a, &b);

    int \*p\_a = &a, \*p\_b = &b;

    printf("%d", \*p\_a + \*p\_b);

    return 0;

}

Результат:



**Задача 3**

Постановка задачи:

Напишите программу, которая находить максимальное число из двух чисел, используя указатели на эти числа.

Математическая модель:

Если \*p\_a = &a, \*p\_b = &b

Тогда res = max(\*p\_a, \*p\_b)

Список идентификаторов:

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Смысл |
| a | int | Число 1 |
| b | int | Число 2 |
| p\_a | int \* | Указатель на число 1 |
| p\_b | int \* | Указатель на число 2 |

Код программы:

#include <stdio.h>

#include <stdlib.h>

int main(void) {

    int a, b;

    scanf("%d\n%d", &a, &b);

    int \*p\_a = &a, \*p\_b = &b;

    if (\*p\_a > \*p\_b)

        printf("%d", \*p\_a);

    else

        printf("%d", \*p\_b);

    return 0;

}

Результат:



**Задача 4**

Постановка задачи: Напишите программу, которая создаёт одномерный динамический массив из чисел с плавающей точкой двойной точности, заполняет его значениями с клавиатуры и распечатывает все элементы этого массива, используя арифметику указателей (оператор +), а не обычный оператор доступа к элементу массива — [].

Математическая модель:

A[i] = \*(A + i)

Список идентификаторов:

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Смысл |
| A | float \* | Массив |
| i | int | Параметр цикла |
| n | int | Размер массива |

Код программы:

#include <stdio.h>

#include <stdlib.h>

int main(void) {

    int n;

    double \*A;

    printf("Vvedite kol-vo chisel:\n");

    scanf("%d", &n);

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

    for (int i = 0; i < n; i++) {

        printf("A[%d] = ", i);

        scanf("%lf", (A + i));

    }

    for (int i = 0; i < n; i++) {

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

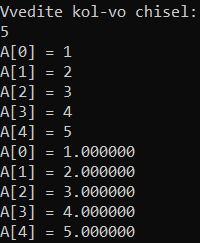
    }

    free(A);

    return 0;

}

Результат:



**Задача 5**

Постановка задачи: Вычислить факториал заданного числа, используя указатель на целое число, а просто не переменную целого типа

Математическая модель:

for (int i = 2; i <= n; i++)

        \*p\_a \*= i

Где \*p\_a = &a

Список идентификаторов:

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Смысл |
| i | int | Параметр цикла |
| n | int | Факториал какого числа вычислить |
| a | int | Результат |
| p\_a | int\* | Результат; указатель на число |

Код:

#include <stdlib.h>

#include <stdio.h>

int main(void) {

    int n, a = 1;

    printf("Vvedite chislo: ");

    scanf("%d", &n);

    int \*p\_a = &a;

    for (int i = 2; i <= n; i++)

        \*p\_a \*= i;

    printf("%d! = %d", n, \*p\_a);

    return 0;

}

Результат:



**Задача 6**

Постановка задачи:

Вывести элементы динамического массива целых чисел в обратном порядке, используя указатель и операцию декремента (--).

Математическая модель:

for (int i = n-1; i >= 0; i--)

        printf("%d\t", \*(A+i));

A - массив

Список идентификаторов:

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Смысл |
| i | int | Параметр цикла |
| n | int | Размер массива |
| A | int\* | Массив |

Код:

#include <stdio.h>

#include <stdlib.h>

int main(void) {

    int \*A, n;

    printf("Vvedite razmer massiva: ");

    scanf("%d", &n);

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

    for (int i = 0; i < n; i++) {

        printf("A[%d] = ", i);

        scanf("%d", (A + i));

    }

    for (int i = n-1; i >= 0; i--)

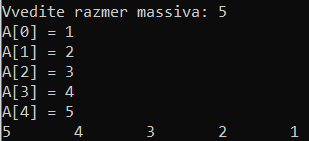
        printf("%d\t", \*(A+i));

    free(A);

    return 0;

}

Результат:



**Задача 7**

Постановка задачи:

Отсортируйте заданный массив целых чисел, используя указатели, а не доступ по индексу ([]).

Математическая модель:

for (int i = 0; i < n - 1; i++)

        for (int j = n - 1; j > i; j--)

            if (\*(array+j-1) > \*(array+j))

                int temp = \*(array+j)

                \*(array+j) = \*(array+j-1)

                \*(array+j-1) = temp

Список идентификаторов:

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Смысл |
| i | int | Параметр цикла |
| j | int | Параметр цикла |
| array | int \* | Массив |
| n | int | Размер массива |

Код программы:

#include <stdio.h>

#include <stdlib.h>

int main(void) {

    int n, \*array;

    printf("Size of array = ");

    scanf("%d", &n);

    // Выделение памяти под массив и его заполнение

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

    for (int i = 0; i < n; i++) {

        printf("array[%d] = ", i);

        scanf("%d", (array + i));

}

    // Сортировка

    for (int i = 0; i < n - 1; i++) {

        for (int j = n - 1; j > i; j--) {

            if (\*(array+j-1) > \*(array+j)) {

                int temp = \*(array+j);

                \*(array+j) = \*(array+j-1);

                \*(array+j-1) = temp;

            }

        }

    }

    printf("Sorted array:\n");

    for (int i = 0; i < n; i++)

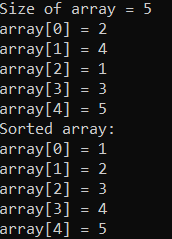
        printf("array[%d] = %d\n", i, \*(array + i) );

    free(array);

    return 0;

}

Результат:



**Задача 8**

Постановка задачи:

Определите переменную целого типа int a = 10; и выведите побайтово её содержимое на экран, используя указатель char \*.

Математическая модель:

a = 10

char \*c = (char\*)&a

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

printf(“%d”, \*(c+i))

Список идентификаторов:

Код:

#include <stdio.h>

#include <stdlib.h>

int main(void) {

    int a = 1234;

    char \*pc = (char\*) &a;

    for (int i = 0; i < 4; i++)

        printf("%d\n", \*(pc + i));

    return 0;

}

Результат:



**Задача 9**

Постановка задачи:

Выделите память под двумерный динамический массив — матрицу — таким образом, чтобы данные все строки этой матрицы гарантированно располагались в оперативной памяти друг за другом (C 2D array contiguous memory allocation).

Математическая модель: -

Список идентификаторов:

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Смысл |
| i | int | Параметр цикла |
| j | int | Параметр цикла |
| N | const | Кол-во строк матрицы |
| M | const | Кол-во столбцов матрицы |
| A | int \* | Матрица |
| start | int \* | Промежуточная переменная |

Код программы:

#include <stdio.h>

#include <stdlib.h>

#define N 2

#define M 3

int main(void) {

    int \*\*A = (int\*\*)malloc(N \* sizeof(int\*) + N \* M \* sizeof(int));

    int \*start = (int \*)((char \*)A + N \* sizeof(int \*));

    for (int i = 0; i < N; i++)

        A[i] = start + i \* M;

    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");

    for (int i = 0; i < N; i++) {

        for (int j = 0; j < M; j++)

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

        printf("\n");

    }

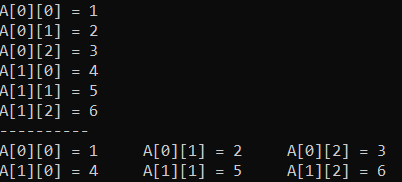
    free(A);

    return 0;

}

// Алгоритм на основе объяснений Тимофея Хирьянова  
// https://www.youtube.com/watch?v=MQBV3PMx5sg&feature=emb\_title

Результат:



**Часть II**

**Задача 1**

Постановка задачи:

Создайте две функции, которые вычисляют факториал числа, используя цикл:

• функцию, которая вычисляет факториал, используя цикл;

• функцию, которая вычисляет факториал, используя рекурсивный вызов самой себя

Математическая модель:

С использованием цикла:

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

r = r \* i;

Рекурсивный метод:

if (n == 1 || n == 0)

n = 1

else n = n \* f(n-1)

return n

Список идентификаторов:

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Смысл |
| fact\_1 | int function | Функция вычисления с рекурсивным вызовом |
| fact\_2 | int function | Функция вычисления с использованием цикла |
| r | int | Переменная результата в функции fact\_2 |
| i | int | Параметр цикла |
| f | int | Результат в главной функции |
| n | int | Аргумент функции |

Код:

#include <stdio.h>

int fact\_1(int n) {

    if (n == 1 || n == 0)

        n = 1;

    else

        n = n \* fact\_1(n - 1);

    return n;

}

int fact\_2(int n) {

    int r = 1;

    for (int i = 1; i <= n; i ++)

        r \*= i;

    return r;

}

int main(void) {

    int f = fact\_1(10);

    printf("%d\n", f);

    f = fact\_2(10);

    printf("%d", f);

    return 0;

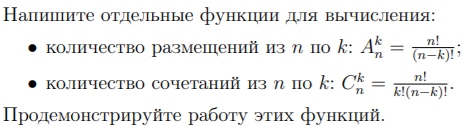
}

Результат:



**Задача 2**

Постановка задачи:



Математичекая модель: формулы представлены в усовии задачи

Список идентификаторов:

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Смысл |
| m | int | Аргумент функции |
| n | int | Аргумент функции |
| k | int | Аргумент функции |
| Ank | int fucntion | Функция вычисления А из n по k |
| Cnk | int function | Функция вычисления C из n по k |
| fact | int function | Функция вычисления факториала |
| res | int | Результат |

Код:

#include <stdio.h>

int fact(int m) {

    if (m == 1 || m == 0)

        m = 1;

    else

        m = m \* fact(m - 1);

    return m;

}

int Ank(int n, int k) {

    return fact(n) / fact(n - k);

}

int Cnk(int n, int k) {

    return fact(n) / (fact(k) \* fact(n - k));

}

int main(void) {

    int res = Ank(4, 2);

    printf("%d\n", res);

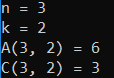
    res = Cnk(4, 2);

    printf("%d", res);

    return 0;

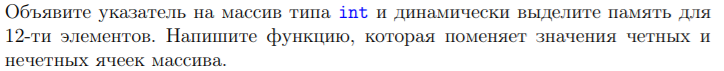
}

Результат:



**Задача 3**

Постановка задачи:



Матемаическая модель:

Пока i < n-1:

Swap(a[i], a[i+1])

i += 2

Список идентификаторов:

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Смысл |
| n | const | Размер массива |
| i | int | Параметр цикла |
| array | int\* | Массив |
| swapElements | void function | Функция обмена |
| arr | int\* | Аргумент функции |

Код:

#include <stdio.h>

#include <stdlib.h>

#define n 12

void swapElements(int\* arr) {

    int i = 0;

    while (i < n - 1) {

        int t = arr[i+1];

        arr[i+1] = arr[i];

        arr[i] = t;

        i += 2;

    }

}

int main(void) {

    int \*array;

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

    for (int i = 0; i < n; i++) {

        printf("a[%d] = ", i);

        scanf("%d", &array[i]);

    }

    swapElements(array);

    printf("---------\n");

    for (int i = 0; i < n; i++)

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

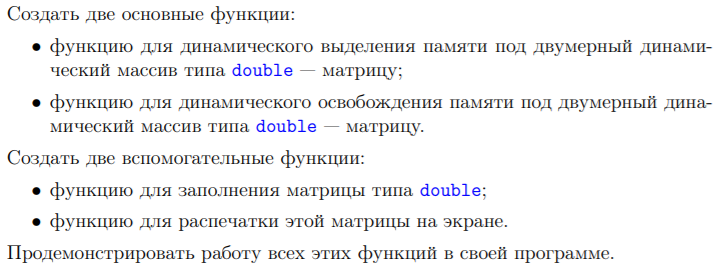
    free(array);

    return 0;

}

**Задача 4**

Постановка здачи:



Математическая модель:

Список идентификаторов:

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Смысл |
| n\_rows | int | Аргумент функций (кол-во строк) |
| m\_columns | int | Аргумент функций (кол-во столбцов) |
| \_A | int\*\* | Аргумент функций (матрица) |
| i | int | Параметр циклов |
| j | int | Параметр циклов |
| n | int | Входной размер матрицы (кол-во строк) |
| m | int | Входной размер матрицы (кол-во столбцов) |
| createMatrix | int\*\* | Функция создания матрицы |
| freeMatrix | void | Функция освобождения памяти |
| fillMatrix | void | Функция заполнения матрицы |
| printMatrix | void | Функция вывода матрицы |
| A | int\*\* | Матрица |

Код:

#include <stdio.h>

#include <stdlib.h>

int\*\* createMatrix(int n\_rows, int m\_columns) {

    int \*\*\_A;

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

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

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

    return \_A;

}

void freeMatrix(int n\_rows, int \*\*\_A) {

    for (int i = 0; i < n\_rows; i++)

        free(\_A[i]);

    free(\_A);

}

void fillMatrix(int n\_rows, int m\_columns, int \*\*\_A) {

    for (int i = 0; i < n\_rows; i++)

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

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

            scanf("%d", &\_A[i][j]);

        }

}

void printMatrix(int n\_rows, int m\_columns, int \*\*\_A) {

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

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

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

        printf("\n");

    }

}

int main(void) {

    printf("Vvedite razmer matritci:\n");

    int n, m;

    printf("n = ");

    scanf("%d", &n);

    printf("m = ");

    scanf("%d", &m);

    int \*\*A = createMatrix(n, m);

    fillMatrix(n, m, A);

    printMatrix(n, m, A);

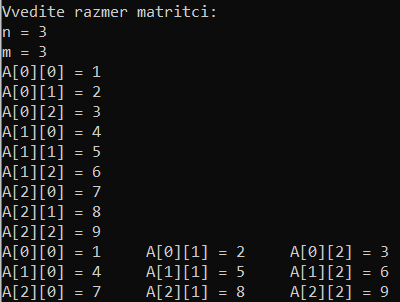
    freeMatrix(n, A);

    system("pause");

    return 0;

}

Результат:



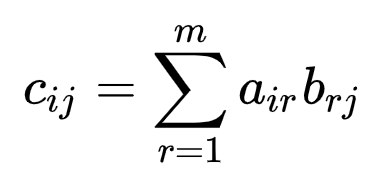
**Задача 5**

Постановка задачи: Создать две функции для динамических массивов (для указателей):

• функцию умножения матрицы типа double на вектор-столбец типа double;

• функцию умножения одной матрицы типа double на другую матрицу типа double.

Математическая модель:



Список идентификаторов:

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Смысл |
| i | int | Параметр цикла |
| j | int | Параметр цикла |
| r | int | Параметр цикла |
| matrix1 | double\*\* | Матрица 1 |
| n\_rows1 | int | Кол-во строк матрицы 1 |
| m\_columns1 | int | Кол-во столбцов матрицы 1 |
| choice | int | Переменная, служащая для выбора действий пользователя |
| vector | double\* | Вектор-столбец |
| n | int | Размер вектор-столбца |
| matrix2 | double\*\* | Матрица 2 |
| n\_rows2 | int | Кол-во строк матрицы 2 |
| m\_columns2 | int | Кол-во столбцов матрицы 2 |
| fillMatrix | void | Функция заполнения матрицы |
| freeMatrix | void | Функция освобождения памяти |
| createMatrix | double\*\* | Функция создания (выделения памяти) матрицы |
| multVector | void | Функция умножения матрицы на вектор-столбец |
| multMatrix | void | Функция умножения двух матриц |
| n\_rows | int | Аргумент функции (кол-во строк передаваемой матрицы) |
| m\_columns | int | Аргумент функции (кол-во столбцов передаваемой матрицы) |
| \_A | double\*\* | Аргумент функции (передаваемая матрица) |
| result | double\* | Локальная переменная функций multVector и multMatrix (служит как результат умножения) |
| l | int | Аргумент функции (размер передаваемой матрицы) |
| q | int | Аргумент функции (размер передаваемой матрицы) |
| k | int | Аргумент функции (размер передаваемой матрицы) |
| \_A1 | double\*\* | Аргумент функции (передаваемая матрица 1) |
| \_A2 | double\*\* | Аргумент функции (передаваемая матрица 2) |

Код:

#include <stdio.h>

#include <stdlib.h>

void fillMatrix(int n\_rows, int m\_columns, double \*\*\_A) { // Функция заполнения матрицы

    for (int i = 0; i < n\_rows; i++)

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

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

            scanf("%lf", &\_A[i][j]);

        }

}

void freeMatrix(int n\_rows, double \*\*\_A) { // Функция освобождения памяти

    for (int i = 0; i < n\_rows; i++)

        free(\_A[i]);

    free(\_A);

}

double\*\* createMatrix(int n\_rows, int m\_columns) { // Функция создания матрицы

    double \*\*\_A;

    \_A = (double \*\*)malloc(n\_rows \* sizeof(double \*));

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

        \_A[i] = (double \*)malloc(m\_columns \* sizeof(double));

    return \_A;

}

void multVector(int n\_rows, int m\_columns, double \*\*\_A, int n, double \*\_vector) {

    double \*result; // Функция умножения вектор-столбца на матрицу

    result = (double \*) calloc(n, sizeof(double));

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

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

            result[i] += \_vector[j] \* \_A[i][j];

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

    }

    free(result);

}

void multMatrix(int l, int q, int k, double \*\*\_A1, double \*\*\_A2) {

    double \*\*result; // Функция умножения двух матриц

    result = createMatrix(l, q);

    for (int i = 0; i < l; i++)

        for (int j = 0; j < q; j++) {

            result[i][j] = 0;

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

                result[i][j] += \_A1[i][r] \* \_A2[r][j];

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

        }

    freeMatrix(l, result);

}

int main(void) {

    double\*\* matrix1;

    int n\_rows1, m\_columns1;

    int choice;

    printf("Enter size of 1st matrix:\nn = ");

    scanf("%d", &n\_rows1);

    printf("m = ");

    scanf("%d", &m\_columns1);

    matrix1 = createMatrix(n\_rows1, m\_columns1);

    fillMatrix(n\_rows1, m\_columns1, matrix1);

    printf("What are you want to do?\n"

    "1 - multiply matrix by column-vector\n"

    "2 - multiply matrix by matrix\n"

    "Enter 1 or 2\n");

    // Спрашиваем у пользователя какое действие он хочет выполнить

    scanf("%d", &choice);

    if (choice == 1) {

        double \*vector;

        int n;

        printf("Enter size of column-vector:\nn = ");

        scanf("%d", &n);

        if (n\_rows1 != n) { // Кол-во столбцов должно быть == кол-ву строк второй

            printf("Error!"); // Иначе выход из программы

            return 1;

        }

        vector = (double \*) malloc(n \* sizeof(double));

        for (int i = 0; i < n; i++) {

            printf("Vector[%d][0] = ", i);

            scanf("%lf", &vector[i]);

        }

        multVector(n\_rows1, m\_columns1, matrix1, n, vector);

        free(vector);

    }

    else if (choice == 2) {

        double \*\*matrix2;

        int n\_rows2, m\_columns2;

        printf("Enter size of 1st matrix:\nn = ");

        scanf("%d", &n\_rows2);

        printf("m = ");

        scanf("%d", &m\_columns2);

        if (n\_rows1 != m\_columns2) {

            printf("Error!");

            return 1;

        }

        matrix2 = createMatrix(n\_rows2, m\_columns2);

        fillMatrix(n\_rows2, m\_columns2, matrix2);

        multMatrix(n\_rows1, m\_columns2, n\_rows2, matrix1, matrix2);

        freeMatrix(n\_rows2, matrix2);

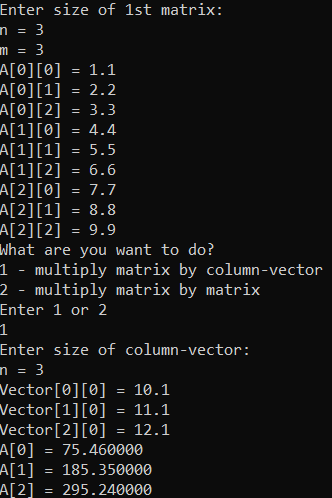
    }

    freeMatrix(n\_rows1, matrix1);

    return 0;

}

Результат:



**Задача 6**

Постановка задачи: Создать функцию, которая вычисляет векторное произведение двух векторов в декартовых координатах, используя указатели на соответствующие массивы.

Математическая модель:



Список идентификаторов:

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Смысл |
| A | int\* | Задаваемый вектор 1 |
| B | int\* | Задаваемые вектор 2 |
| result | int\* | Результативный вектор в главной функции (main) |
| i | int | Параметр цикла |
| n | const | Размер векторов |
| fillVector | int\* | Функция создания и заполнения вектора |
| vector | int\* | Возвращаемое значение полученного вектора в функции fillVector |
| multVectors | int\* | Функция умножения двух векторов |
| vector1 | int\* | Передаваемый вектор 1 в функцию multVectors |
| vector2 | int\* | Передаваемый вектор 2 в функцию multVectors |
| res | int\* | Результат умножения векторов в функции multVectors |

Код:

#include <stdio.h>

#include <stdlib.h>

#define n 3

int\* fillVector() {

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

    for (int i = 0; i < n; i++) {

        printf("[%d] = ", i);

        scanf("%d", &vector[i]);

    }

    return vector;

}

int\* multVectors(int \*vector1, int \*vector2) {

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

    \*(res + 0) = \*(vector1 + 1) \* \*(vector2 + 2) - \*(vector1 + 2) \* \*(vector2 + 1);

    \*(res + 1) = \*(vector1 + 2) \* \*(vector2 + 0) - \*(vector1 + 0) \* \*(vector2 + 2);

    \*(res + 2) = \*(vector1 + 0) \* \*(vector2 + 1) - \*(vector1 + 1) \* \*(vector2 + 0);

    return res;

}

int main(void) {

    int \*A;

    int \*B;

    printf("1st vector A:\n");

    A = fillVector();

    printf("2nd vector B:\n");

    B = fillVector();

    int \*result = multVectors(A, B);

    for (int i = 0; i < n; i++)

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

    free(A);

    free(B);

    free(result);

    return 0;

}

Результат:

