Лабораторная работа 5.

«Массивы»

Часть 1.

Задание 1.

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

|  |  |  |
| --- | --- | --- |
| Имя | Смысл | Тип |
| n | Количество координат векторов | int |
| y[n] | Числовой вектор Y | float |
| x[n] | Числовой вектор X | float |
| i | Параметр цикла | int |



#include <stdio.h>

#include <math.h>

#include <locale.h>

int main()

{

char \*locale=setlocale(LC\_ALL, "Russian");

int n, i;

printf("Введите количество координат для векторов: ");

scanf("%d", &n);

float x[n-1], y[n-1];

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

{

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

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

}

printf("Полученный вектор Y: \n");

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

{

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

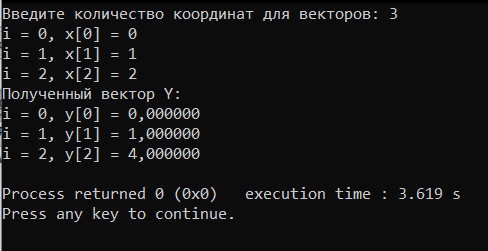
printf("i = %d, y[%d] = %f\n", i, i, y[i]);

}

return 0;

}





Задание 2.

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

вспомогательным массивом.

|  |  |  |
| --- | --- | --- |
| Имя | Смысл | Тип |
| n | Количество элементов массива | int |
| x[n] | Числовой массив, ведённый с клавиатуры | int |
| buf | Буфер | int |
| i | Параметр цикла | int |
| y[n] | Вспомогательный массив | int |



#include <stdio.h>

#include <math.h>

#include <locale.h>

int main()

{

char \*locale=setlocale(LC\_ALL, "Russian");

int n, i;

printf("Введите количество элементов массива: ");

scanf("%d", &n);

int x[n - 1], y[n - 1];

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

{

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

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

}

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

y[i] = x[n - 1 - i];

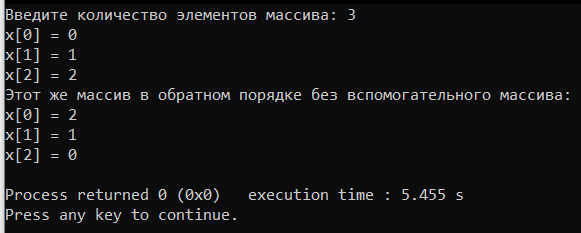
printf("Этот же массив в обратном порядке без вспомогательного массива: \n");

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

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

return 0;

}



Задание 3.

1. 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.

|  |  |  |
| --- | --- | --- |
| Имя | Смысл | Тип |
| a[10] | Массив случайных чисел | int |
| i | Параметр внешнего цикла | int |
| j | Параметр внутреннего цикла | int |
| buf | Буфер | int |



|  |  |
| --- | --- |
| Сортировка с начала по возрастанию | Сортировка с конца по возрастанию |
| #include <stdio.h>  #include <math.h>  #include <locale.h>  int main()  {  char \*locale=setlocale(LC\_ALL, "Russian");  int a[10], i, j, buf;  srand(time(0));  for (int i = 0; i < 10; i++)  {  a[i] = rand() % 101;  printf("%d ", a[i]);  }  for (i = 0; i < 10 - 1; i++)  {  for (j = (10 - 1); j > i; j--)  {  if (a[j - 1] > a[j])  {  buf = a[j - 1];  a[j - 1] = a[j];  a[j] = buf;  }  }  }  printf("\n");  for (i = 0; i < 10; i++)  printf("%d ", a[i]);  return 0;  } | #include <stdio.h>  #include <math.h>  #include <locale.h>  int main()  {  char \*locale=setlocale(LC\_ALL, "Russian");  int a[10], i, j, buf;  srand(time(0));  for (int i = 0; i < 10; i++)  {  a[i] = rand() % 101;  printf("%d ", a[i]);  }  for (i = 10; i >= 0; i--)  {  for (j = 1; j < i; j++)  {  if (a[j - 1] > a[j])  {  buf = a[j - 1];  a[j - 1] = a[j];  a[j] = buf;  }  }  }  printf("\n");  for (i = 0; i < 10; i++)  printf("%d ", a[i]);  return 0;  } |
|  |  |
| Сортировка с начала по убыванию | Сортировка с конца по убыванию |
| #include <stdio.h>  #include <math.h>  #include <locale.h>  int main()  {  char \*locale=setlocale(LC\_ALL, "Russian");  int a[10], i, j, buf;  srand(time(0));  for (int i = 0; i < 10; i++)  {  a[i] = rand() % 101;  printf("%d ", a[i]);  }  for (i = 0; i < 10 - 1; i++)  {  for (j = (10 - 1); j > i; j--)  {  if (a[j - 1] < a[j])  {  buf = a[j - 1];  a[j - 1] = a[j];  a[j] = buf;  }  }  }  printf("\n");  for (i = 0; i < 10; i++)  printf("%d ", a[i]);  return 0;  } | #include <stdio.h>  #include <math.h>  #include <locale.h>  int main()  {  char \*locale=setlocale(LC\_ALL, "Russian");  int a[10], i, j, buf;  srand(time(0));  for (int i = 0; i < 10; i++)  {  a[i] = rand() % 101;  printf("%d ", a[i]);  }  for (i = 10; i >= 0; i--)  {  for (j = 1; j < i; j++)  {  if (a[j - 1] < a[j])  {  buf = a[j - 1];  a[j - 1] = a[j];  a[j] = buf;  }  }  }  printf("\n");  for (i = 0; i < 10; i++)  printf("%d ", a[i]);  return 0;  } |
|  |  |

Задание 4.

1. Реализовать самостоятельно алгоритм сортировки вставками и сравнить его реализацию с полученными реализациями «алгоритма пузырька» в задании 3.

|  |  |  |
| --- | --- | --- |
| Имя | Смысл | Тип |
| n | Количество элементов массива | int |
| i | Параметр цикла | int |
| a[n-1] | Массив случайных чисел | int |
| newElement | Значение текущего элемента | int |
| location | Индекс текущего элемента | int |



#include <stdio.h>

#include <math.h>

#include <locale.h>

int main()

{

char \*locale=setlocale(LC\_ALL, "Russian");

int n, i, newElement, location;

printf("Введите количество элементов массива: ");

scanf("%d", &n);

int a[n - 1];

srand(time(0));

for (int i = 0; i < 10; i++)

{

a[i] = rand() % 101;

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

}

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

{

newElement = a[i];

location = i - 1;

while(location >= 0 && a[location] > newElement)

{

a[location + 1] = a[location];

location = location - 1;

}

a[location + 1] = newElement;

}

printf("\n");

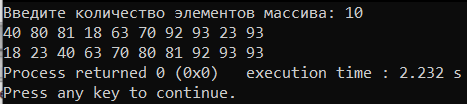
for (i = 0; i < 10; i++)

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

return 0;

}





Задание 5.

1. Организовать ввод массива (матрицы) по столбцам.

|  |  |  |
| --- | --- | --- |
| Имя | Смысл | Тип |
| i | Параметр внешнего цикла, текущий столбец матрицы | int |
| j | Параметр внутреннего цикла, текущая строка | int |
| a[5][5] | Вводимая матрица | int |



#include <stdio.h>

#include <math.h>

#include <locale.h>

int main()

{

char \*locale=setlocale(LC\_ALL, "Russian");

int i, j;

int a[5][5];

for (i = 0; i < 5; i++)

{

printf("Введите значение 5-ти элементов для %d - го столбца: ", i+1);

scanf("%d %d %d %d %d", &a[0][i], &a[1][i], &a[2][i], &a[3][i], &a[4][i]);

}

printf("Полученная матрица: \n");

for (i = 0; i < 5; i++)

{

for (j = 0; j < 5; j++)

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

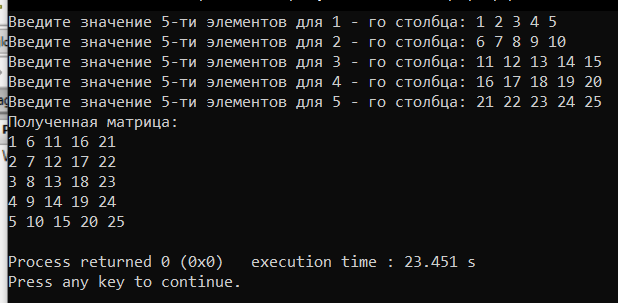
printf("\n");

}

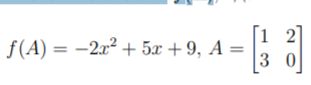
return 0;

}





Задание 6.

1. Найти значения матричного многочлена f(A)
2. 

|  |  |  |
| --- | --- | --- |
| Имя | Смысл | Тип |
| i | Параметр внешнего цикла, текущая строка матриц | int |
| j | Параметр внутреннего цикла, текущий столбик матриц | int |
| a[2][2] | Исходная матрица | int |
| res[2][2] | Результативная матрица | int |



#include <stdio.h>

#include <math.h>

#include <locale.h>

int main()

{

char \*locale=setlocale(LC\_ALL, "Russian");

int i, j;

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

int res[2][2];

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

{

for (j = 0; j < 2; j++)

res[i][j] = (-2) \* a[i][j] \* a[i][j] + 5 \* a[i][j] + 9;

}

printf("Полученная матрица: \n");

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

{

for (j = 0; j < 2; j++)

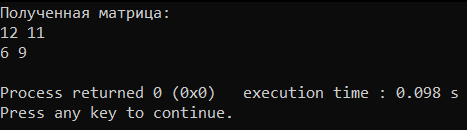
printf("%d ", res[i][j]);

printf("\n");

}

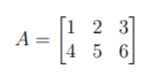
return 0;

}



Задание 7.

1. Транспонировать матрицу





|  |  |  |
| --- | --- | --- |
| Имя | Смысл | Тип |
| i | Параметр внешнего цикла, текущая строка матриц | int |
| j | Параметр внутреннего цикла, текущий столбик матриц | int |
| a[2][3] | Исходная матрица | int |
| res[3][2] | Результативная матрица после транспонирования матрицы a[2][3] | int |



#include <stdio.h>

#include <math.h>

#include <locale.h>

int main()

{

char \*locale=setlocale(LC\_ALL, "Russian");

int i, j;

int a[2][3] = {1, 2, 3,

4, 5, 6};

int res[3][2];

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

{

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

res[j][i] = a[i][j];

}

printf("Полученная матрица: \n");

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

{

for (j = 0; j < 2; j++)

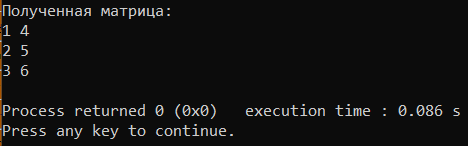
printf("%d ", res[i][j]);

printf("\n");

}

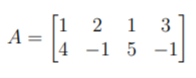
return 0;

}



Задание 8.

1. Вычислить произведения A × AT и AT × A при заданной матрице A:





|  |  |  |
| --- | --- | --- |
| Имя | Смысл | Тип |
| i | Параметр внешнего цикла, текущая строка матриц | int |
| j | Параметр внутреннего цикла, текущий столбик матриц | int |
| n | Параметр цикла | int |
| a[2][4] | Исходная матрица | int |
| aT[4][2] | Результативная матрица после транспонирования матрицы a[2][4] | int |
| res1[2][2] | Матрица результата умножения матрицы a[2][4] на транспонированную матрицу aT][4][2] | int |
| res[4][4] | Матрица результата умножения транспонированной матрицы aT][4][2] на матрицу a[2][4] | int |



#include <stdio.h>

#include <math.h>

#include <locale.h>

int main()

{

char \*locale=setlocale(LC\_ALL, "Russian");

int i, j;

int a[2][4] = {1, 2, 1, 3,

4, -1, 5, -1};

printf("Исходная матрица: \n");

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

{

for (j = 0; j < 4; j++)

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

printf("\n");

}

int aT[4][2];

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

{

for (j = 0; j < 4; j++)

aT[j][i] = a[i][j];

}

printf("Транспонируемая матрица: \n");

for (i = 0; i < 4; i++)

{

for (j = 0; j < 2; j++)

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

printf("\n");

}

int res1[2][2] = {0, 0, 0, 0};

printf("A x A^T: \n");

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

for (j = 0; j < 2; j++)

for (int n = 0; n < 4; n++)

res1[i][j] += a[i][n] \* aT[n][j];

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

{

for (j = 0; j < 2; j++)

printf("%d ", res1[i][j]);

printf("\n");

}

int res2[4][4] = {0, 0, 0, 0,

0, 0, 0, 0,

0, 0, 0, 0,

0, 0, 0, 0};

printf("A^T x A: \n");

for (i = 0; i < 4; i++)

for (j = 0; j < 4; j++)

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

res2[i][j] += aT[i][n] \* a[n][j];

for (i = 0; i < 4; i++)

{

for (j = 0; j < 4; j++)

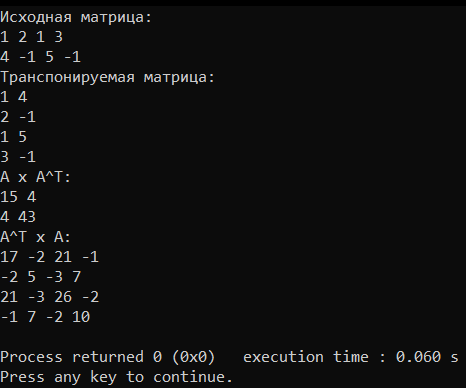
printf("%d ", res2[i][j]);

printf("\n");

}

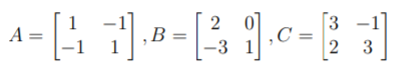
return 0;

}



Задание 9.

1. Найти произведения матриц (AB) · C и A · (BC)





|  |  |  |
| --- | --- | --- |
| Имя | Смысл | Тип |
| a[2][2] | Исходная матрица А | int |
| a[2][2] | Исходная матрица В | int |
| a[2][2] | Исходная матрица С | int |
| res1[2][2] | Результирующая матрица первого произведения | int |
| res2[2][2] | Результирующая матрица второго произведения | int |
| i | Параметр внешнего цикла, текущая строка матриц | int |
| j | Параметр внутреннего цикла, текущий столбик матриц | int |
| n | Параметр цикла | int |

#include <stdio.h>

#include <math.h>

#include <locale.h>

int main()

{

char \*locale=setlocale(LC\_ALL, "Russian");

int i, j;

int a[2][2] = {1, -1,

-1, 1};

int b[2][2] = {2, 0,

-3, 1};

int c[2][2] = {3, -1,

2, 3};

int res1[2][2] = {0, 0, 0, 0};

int res2[2][2] = {0, 0, 0, 0};

printf("(A x B) X C: \n");

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

for (j = 0; j < 2; j++)

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

res1[i][j] += a[i][n] \* b[n][j];

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

for (j = 0; j < 2; j++)

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

res2[i][j] += res1[i][n] \* c[n][j];

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

{

for (j = 0; j < 2; j++)

{

printf("%d ", res2[i][j]);

res1[i][j] = 0;

res2[i][j] = 0;

}

printf("\n");

}

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

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

for (j = 0; j < 2; j++)

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

res1[i][j] += b[i][n] \* c[n][j];

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

for (j = 0; j < 2; j++)

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

res2[i][j] += a[i][n] \* res1[n][j];

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

{

for (j = 0; j < 2; j++)

printf("%d ", res2[i][j]);

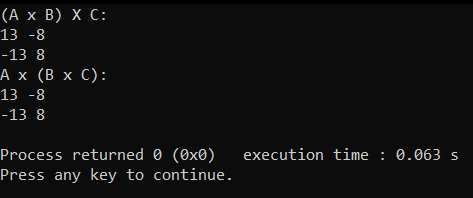
printf("\n");

}

return 0;

}





Задание 10.

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

|  |  |  |
| --- | --- | --- |
| Имя | Смысл | Тип |
| i | Параметр внешнего цикла, текущая строчка матрицы | int |
| j | Параметр внутреннего цикла, текущий столбец матрицы | int |
| a[4][4] | Исходная матрица, заполняется случайными значениями | int |



#include <stdio.h>

#include <math.h>

#include <locale.h>

int main()

{

char \*locale=setlocale(LC\_ALL, "Russian");

int i, j;

int a[4][4];

srand(time(0));

printf("Исходная матрица: \n");

for (i = 0; i < 4; i++)

{

for (j = 0; j < 4; j++)

{

a[i][j] = rand() % 101;

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

}

printf("\n");

}

for (i = 0; i < 4; i++)

{

for (j = 1; j < 4; j++)

a[i][0] += a[i][j];

a[i][0] /= 4;

}

printf("Результат: \n");

for (i = 0; i < 4; i++)

{

for (j = 0; j < 4; j++)

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

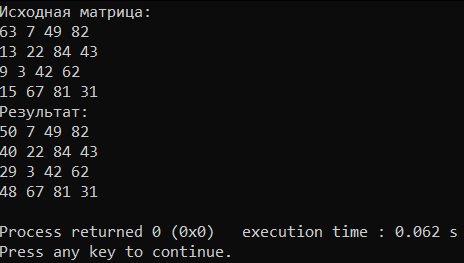
printf("\n");

}

return 0;

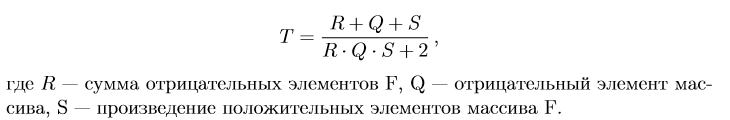
}





Часть 2.

Задание 1.

1. Создать динамический одномерный массив целых чисел F размерности 12 и заполнить его положительными и отрицательными числами.
2. 

|  |  |  |
| --- | --- | --- |
| Имя | Смысл | Тип |
| F | Динамический массив | int |
| R | Сумма отрицательных элементов F | int |
| Q | Отрицательный элемент F | int |
| S | Произведение положительных элементов массива F | int |
| T | Искомая переменная | float |
| i | Параметр цикла | int |
| n | Размерность массива | int |



#include <stdio.h>

#include <stdlib.h>

int main() {

int R = 0, Q = 0, S = 1, n = 12;

float T;

int \*F = (int \*)malloc(12 \* sizeof(int));

srand(time(NULL));

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

F[i] = rand() % 101 - 50;;

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);

}

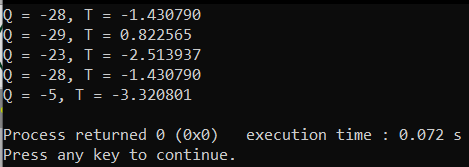
}

free(F);

return 0;

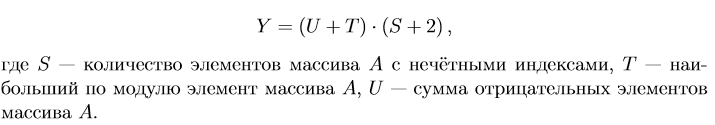
}





Задание 2.

1. Дан одномерный массив целых чисел A размера 12. Вычислить:





|  |  |  |
| --- | --- | --- |
| Имя | Смысл | Тип |
| A | Динамический массив | int |
| S | Количество элементов массива А с нечётными индексами | int |
| T | Наибольший по модулю элемент массива А | int |
| U | Сумма отрицательных элементов массива А | int |
| Y | Искомая переменная | int |
| n | Размерность массива | int |
| i | Параметр цикла | int |



#include <stdio.h>

#include <stdlib.h>

int main() {

int S = 0, T = 0, U = 0, Y, n = 12;

int \*A = (int \*)malloc(12 \* sizeof(int));

srand(time(NULL));

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

A[i] = rand() % 101 - 50;

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

if (i % 2 == 1)

S++;

if (abs(A[i]) > T)

T = abs(A[i]);

if (A[i] < 0)

U += A[i];

}

Y = (U + T) \* (S + 2);

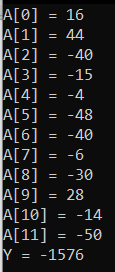
printf("Y = %d\n", Y);

free(A);

return 0;

}





Задание 3.

1. 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.
2. Если a[i] = b[i], то вывести a[i]

|  |  |  |
| --- | --- | --- |
| Имя | Смысл | Тип |
| fib(a) | Функция подсчёта следующего числа из ряда чисел Фибоначчи | int |
| pad(b) | Функция подсчёта следующего числа из ряда чисел Падована | int |
| a, b | Локальные переменные функций fib(a) и pad(b) соответственно | int |
| A | Динамический массив, состоящий из первых 12 чисел Фибоначчи | int |
| B | Динамический массив, состоящий из первых 12 чисел Падована | int |
| buf | Буфер | int |
| i | Параметр цикла | int |



#include <stdio.h>

#include <stdlib.h>

#define n 12

int fib(int a)

{

if (a == 1 || a == 2)

a = 1;

else

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

return a;

}

int pad(int b)

{

if (b == 1 || b == 2 || b == 3)

b = 1;

else

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

return b;

}

int main() {

int \*A;

int \*B;

int buf = 0;

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

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

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

{

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

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

}

printf("\n");

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

{

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

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

}

printf("\n");

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

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

if (A[i] == B[j] && A[i] != buf)

{

buf = A[i];

printf("%d\t", buf);

}

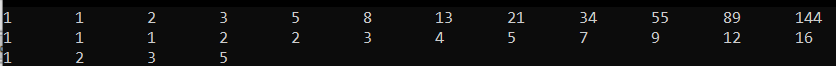
free(A);

free(B);

return 0;

}





Задание 4.

1. 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 A M×N должна быть возможность обхода всех элементов «новой» матрицы A M−1×N таким же способом, что и изначальной матрицы. Распечатать (используя циклы) матрицу до удаления строки и после удаления.

|  |  |  |
| --- | --- | --- |
| Имя | Смысл | Тип |
| m | Количество строк в матрице | int |
| n | Количество столбцов в матрице | int |
| d | Номер удаляемой строки | int |
| A | Исходная матрица | int |
| B | «Новая» матрица | int |
| i | Параметр цикла | int |
| j | Параметр цикла | int |



#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

int main() {

int m, n, d;

int \*\*A = NULL;

srand(time(NULL));

char \*locale=setlocale(LC\_ALL, "Russian");

printf("n = ");

scanf("%d", &n);

printf("m = ");

scanf("%d", &m);

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

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++) {

A[i][j] = rand()%101 - 50;

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

}

printf("\n");

}

printf("Какую строку нужно удалить?\n");

scanf("%d", &d);

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++) {

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

if (i < d)

B[i][j] = A[i][j];

else

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

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

}

printf("\n");

}

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

free(A[i]);

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

free(B[i]);

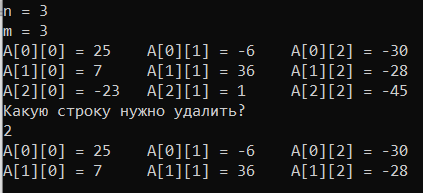
free(A);

free(B);

return 0;

}





Задание 5.

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

|  |  |  |
| --- | --- | --- |
| Имя | Смысл | Тип |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |



#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

int main() {

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

srand(time(NULL));

char \*locale=setlocale(LC\_ALL, "Russian");

printf("Размер вектора a = ");

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

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

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

a[j] = rand() % 101 - 50;

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

}

printf("\nРазмерность матрицы:\n");

printf("m = ");

scanf("%d", &m);

printf("n = ");

scanf("%d", &n);

if (m != n\_a) {

printf("Умножение невозможно.");

return 0;

}

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++) {

matrix[i][j] = rand() % 101 - 50;

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

}

printf("\n");

}

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

printf("\nРезультат:\n");

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]);

}

free(b);

free(a);

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

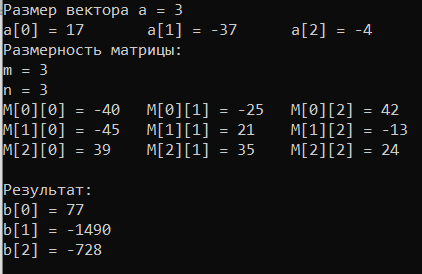
free(matrix[i]);

free(matrix);

return 0;

}





Задание 6.

1. 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.