**Лабораторная работа № 14. Многомерные массивы**

Многомерными называются массивы, имеющие два и более индексов, которые заключены в квадратные скобки.

**Имя массива без индекса** является **указателем-константой** на массив из указателей, каждый из которых содержит адреса начальных элементов строк. Для доступа к элементу **А[1][2]** двумерного массива через указатели можно использовать следующую форму записи: **\*(\*(А + 1) + 2)** или **\*(А[1] + 2)** или **(\*(А + 1))[2]**.

|  |  |
| --- | --- |
| **Задание** | **Краткие теоретические сведения** |
| 1. Выполнить программу, записанную в правой части. Внести изменения в программу с тем, чтобы инициализировался другой массив. Осуществить вывод этого массива в виде матрицы. | Пример программы, которая инициализирует массив и выводит его элементы на экран.  **#include <iostream>**  **void main()**  **{ const int ci = 3, const int cj = 2;**  **int a[ci][cj] = { { 1, 2 }, { 3, 4 }, { 5, 6 } };**  **for (int i = 0; i < ci; i++)**  **for (int j = 0; j < cj; j++)**  **std::cout << "\n a[" << i << "," << j << "] =" << a[i][j];**  **}**    #include <iostream>  int main()  {  const int ci = 3; const int cj = 2;  int a[ci][cj] = { {2, 6}, {1,5}, {3, 4} };  for (int i = 0; i < ci; ++i) {  for (int j = 0; j < cj; ++j) {  std::cout << a[i][j] << " ";  }  std::cout << std::endl;  }  } | |
| 2. Выполнить программы, записанные в правой части для одной и той же задачи, условие которой надо определить. Одна из программ использует указатели. | |  |  | | --- | --- | | **#include <iostream>**  **void main()**  **{ const int n = 3, m = 2, h = 0;**  **int A[n][m] = { 0, 1, 2, 3, 4, 5 };**  **for(int i = 0; i < n; i++)**  **for(int j = 0; j < m; j++)**  **h += A[i][j];**  **std::cout << h;**  **}** | **#include <iostream>**  **void main()**  **{ const int n = 3, m = 2, h = 0;**  **int C[n][m] = { 0, 1, 2, 3, 4, 5 };**  **for(int i = 0; i < n; i++)**  **for(int j = 0; j < m; j++)**  **h += \*(\*(C + i) + j);**  **std::cout << h;**  **}** |   Программы для вычисления суммы всех элементов двумерного массива. | |
| 3. Изучить способы работы с двумерными массивами, выполнив программу в правой части и записав ее условие.  Внести изменения в программу с тем, чтобы определялись минимальные элементы каждой строки. Для доступа к элементам матрицы использовать указатели. | **#include <iostream>**  **void main()**  **{ setlocale(LC\_CTYPE, "Russian");**  **using namespace std;**  **const int n = 2, m = 4;**  **int B[n][m];**  **int i, j, row = 0, column = 0;**  **cout << "Введите элементы массива" << endl;**  **for (i = 0; i < n; i++)**  **for (j = 0; j < m; j++)**  **cin >> B[i][j];**  **int min = B[0][0];**  **for (int i = 0; i < n; i++)**  **for (int j = 0; j < m; j++)**  **if (min > B[i][j])**  **{ min = B[i][j];**  **column = i;**  **row = j;**  **}**  **cout << " Исходный массив:" << endl;**  **for (int i = 0; i < n; i++)**  **{ cout << "\n";**  **for (int j = 0; j < m; j++)**  **cout << "B[" << i << "," << j << "] =" << B[i][j] << "\t";**  **}**  **cout << endl;**  **cout << "Минимальный элемент B[" << column << "," << row << "] ="<<min<< endl;}**  **Программа ищет минимальный элемент в массиве** |
|  | #include <iostream>  using namespace std;  int main() {  setlocale(LC\_CTYPE, "Russian");  const int n = 2, m = 4;  int B[n][m];  cout << "Введите элементы массива:" << endl;    for (int i = 0; i < n; i++) {  for (int j = 0; j < m; j++) {  cin >> \*(\*(B + i) + j);  }  }    cout << "Исходный массив:" << endl;  for (int i = 0; i < n; i++) {  for (int j = 0; j < m; j++) {  cout << "B[" << i << "," << j << "] = " << \*(\*(B + i) + j) << "\t";  }  cout << endl;  }    cout << "Минимальные элементы каждой строки:" << endl;  for (int i = 0; i <= n; i++) {  int min = \*(\*(B + i));  for (int j = 1; j < m; j++) {  if (\*(\*(B + i) + j) < min) {  min = \*(\*(B + i) + j);  }  }  cout << "Минимальный элемент строки " << i+1 << " = " << min << endl;  }  return 0;  } |

4. Выполнить задания из таблицы ниже, используя ***индексы*** для доступа к элементам массивов в первой программе и ***указатели*** − во второй программе.

|  |  |
| --- | --- |
| **№ варианта** | **Условие задачи** |
| **11** | 1. Найти наибольший элемент главной диагонали матрицы **A(n, n)** и вывести на печать всю строку, в которой он находится.   #include <iostream>  using namespace std;  int main() {  setlocale(LC\_ALL, "RU");  int n;  cout << "Введите размер матрицы: ";  cin >> n;  int A[100][100];  cout << "Введите элементы марицы: " << endl;  for (int i = 0; i < n; i++) {  for (int j= 0; j < n; j++) {  cin >> A[i][j];  }  }  int maxdiag = A[0][0];  int maxel = 0;  for(int i = 0; i < n; i++) {  if (A[i][i] > maxdiag) {  maxdiag = A[i][i];  maxel = i;  }  }  cout << "Наибольший элемент главной диагонали: " << maxdiag << endl;  cout << "Строка, содержащая этот элемент:" << endl;  for (int j = 0; j < n; ++j) {  cout << A[maxel][j] << " ";  }  cout << endl;  return 0;  }     1. Определить в матрице первый столбец, все элементы которого отрицательны, и среднее арифметическое этих элементов. Вычесть полученное значение из всех элементов матрицы.   #include <iostream>  using namespace std;  int main() {  setlocale(LC\_ALL, "RU");  int n, m;  cout << "Введите размеры матрицы (n x m): ";  cin >> n >> m;  int A[100][100];  cout << "Введите элементы матрицы: ";  for (int i = 0; i < n; i++) {  for (int j = 0; j < m; j++) {  cin >> \*(\*(A + i) + j);  }  }  int targetColumn = -1;  double sum = 0;  int count = 0;  for (int j = 0; j < m; ++j) {  bool allNegative = true;  for (int i = 0; i < n; ++i) {  if (\*(\*(A + i) + j) >= 0) {  allNegative = false;  break;  }  }  if (allNegative) {  targetColumn = j;  for (int i = 0; i < n; ++i) {  sum += \*(\*(A + i) + j);  }  count = n;  break;  }  }  if (targetColumn == -1) {  cout << "В матрице нет столбцов, все элементы которых отрицательны." << endl;  }  else {  double average = sum / count;  cout << "Первый столбец с отрицательными элементами: " << targetColumn + 1 << endl;  cout << "Среднее арифметическое элементов этого столбца: " << average << endl;    for (int i = 0; i < n; ++i) {  for (int j = 0; j < m; ++j) {  \*(\*(A + i) + j) -= average;  }  }  cout << "Изменённая матрица:" << endl;  for (int i = 0; i < n; ++i) {  for (int j = 0; j < m; ++j) {  cout << \*(\*(A + i) + j) << " ";  }  cout << endl;  }  }  return 0;  } |

6. Дополнительные задания.

1. Дана квадратная матрица порядка **2n**, элементы которой формируются случайным образом и находятся в пределах от −10 до 10. Получить новую матрицу, переставляя ее блоки размера **n×n** в соответствии со схемой.

#include <iostream>

#include <cstdlib>

#include <ctime>

using namespace std;

void fillMatrix(int\*\* matrix, int size, int minVal, int maxVal) {

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

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

matrix[i][j] = rand() % (maxVal - minVal + 1) + minVal;

}

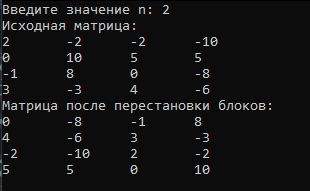
}

}

void printMatrix(int\*\* matrix, int size) {

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

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

 cout << matrix[i][j] << "\t";

}

cout << endl;

}

}

void rearrangeBlocks(int\*\* matrix, int n) {

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

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

swap(matrix[i][j], matrix[i + n][j + n]);

swap(matrix[i][j + n], matrix[i + n][j]);

}

}

}

int main() {

setlocale(LC\_ALL, "RU");

srand(time(0));

int n;

cout << "Введите значение n: ";

cin >> n;

int size = 2 \* n;

int\*\* matrix = new int\* [size];

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

matrix[i] = new int[size];

}

fillMatrix(matrix, size, -10, 10);

cout << "Исходная матрица:" << endl;

printMatrix(matrix, size);

rearrangeBlocks(matrix, n);

cout << "Матрица после перестановки блоков:" << endl;

printMatr ix(matrix, size);

return 0;

}

1. Латинским квадратом порядка **n** называется квадратная таблица размером **nхn**, каждая строка и каждый столбец которой содержат все числа от 1 до **n**. Для заданного **n** в матрице **L(n, n**) построить латинский квадрат порядка **n**.

#include <iostream>

using namespace std;

void generateLatinSquare(int\*\* L, int n) {

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

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

L[i][j] = (i + j) % n + 1;

}

}

}

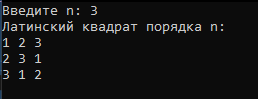
void printSquare(int\*\* L, int n) {

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

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

cout << L[i][j] << " ";

}

 cout << endl;

}

}

int main() {

setlocale(LC\_ALL, "RU");

int n;

cout << "Введите n: "; cin >> n;

int\*\* L = new int\* [n];

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

L[i] = new int[n];

}

generateLatinSquare(L, n);

cout << "Латинский квадрат порядка n: " << endl;

printSquare(L, n);

}

1. Путем перестановки элементов квадратной вещественной матрицы добиться того, чтобы ее максимальный элемент находился в левом верхнем углу, следующий по величине − в позиции (2, 2), следующий − в позиции (3, 3) и т. д., заполнив таким образом всю главную диагональ.

#include <iostream>

#include <algorithm>

using namespace std;

const int MAX\_SIZE = 100;

void printMatrix(double matrix[MAX\_SIZE][MAX\_SIZE], int n) {

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

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

cout << matrix[i][j] << "\t";

}

cout << endl;

}

}

void rearrangeMatrix(double matrix[MAX\_SIZE][MAX\_SIZE], int n) {

double allElements[MAX\_SIZE \* MAX\_SIZE];

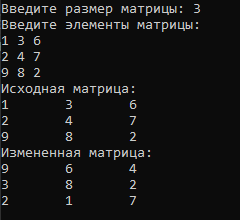
int index = 0;

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

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

allElements[index++] = matrix[i][j];

}

 }

sort(allElements, allElements + n \* n, greater<double>());

index = 0;

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

matrix[i][i] = allElements[index++];

}

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

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

if (i != j) {

matrix[i][j] = allElements[index++];

}

}

}

}

int main() {

setlocale(LC\_ALL, "RU");

int n;

double matrix[MAX\_SIZE][MAX\_SIZE];

cout << "Введите размер матрицы: ";

cin >> n;

if (n > MAX\_SIZE) {

cout << "Размер матрицы превышает максимальный допустимый размер." << endl;

return 1;

}

cout << "Введите элементы матрицы:" << endl;

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

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

cin >> matrix[i][j];

}

}

cout << "Исходная матрица:" << endl;

printMatrix(matrix, n);

rearrangeMatrix(matrix, n);

cout << "Измененная матрица:" << endl;

printMatrix(matrix, n);

return 0;

}