using namespace std;

enum MenuCodes

{

PRINT\_ARRAY,

TASK\_1,

TASK\_2,

TASK\_3,

EXIT

};

const int MINIMUM\_RANDOM\_NUMBER = -12;

const int MAXIMUM\_RANDOM\_NUMBER = 12;

const int MINIMUM\_ARRAY\_BORDER = 3;

const int MAXIMUM\_ARRAY\_BORDER = 17;

void Menu();

void FillArray(int\*\* array, int arrayBorder, int mode);

void PrintArray(int\*\* array, int arrayBorder);

int SumOfDigits(int number);

int FindSideDiagonalSameRowNumber(int\*\* array, int lengthOfRow, int numberOfRow);

void IncreaseArrayElements(int\*\* array, int arrayBorder);

void CheckSignAlternation(int\*\* array, int arrayBorder);

void FindZeroElement(int\*\* array, int arrayBorder);

int main()

{

setlocale(LC\_ALL, "Russian");

srand(time(0));

Menu();

return 0;

}

void Menu()

{

cout << "Введите размер квадратной матрицы (" << MINIMUM\_ARRAY\_BORDER << '-' << MAXIMUM\_ARRAY\_BORDER << "): ";

int arrayBorder{};

cin >> arrayBorder;

while (arrayBorder < MINIMUM\_ARRAY\_BORDER || arrayBorder > MAXIMUM\_ARRAY\_BORDER)

{

cout << "Неподходящее значение, попробуйте еще раз: ";

cin >> arrayBorder;

}

cout << "\nВыберете способ задания матрицы\n";

cout << "0. Вручную\n";

cout << "1. Случайно\n";

cout << "Выберете команду : ";

int mode{};

cin >> mode;

while (mode != 0 && mode != 1)

{

cout << "Неверная команда, попробуйте еще раз: ";

cin >> mode;

}

cout << '\n';

int\*\* array = new int\* [arrayBorder] {};

FillArray(array, arrayBorder, mode);

int code = -1;

while (code != EXIT)

{

cout << PRINT\_ARRAY << ". Вывести матрицу\n";

cout << TASK\_1 << ". Сложить элементы с суммой цифр элемента побочной диагонали той же строки\n";

cout << TASK\_2 << ". Проверка строки на чередование знаков\n";

cout << TASK\_3 << ". Поиск первого нулевого элемента\n";

cout << EXIT << ". Выход\n";

cout << "Выберете команду: ";

cin >> code;

while (code < PRINT\_ARRAY || code > EXIT)

{

cout << "Неверная команда, попробуйте еще раз: ";

cin >> code;

}

cout << "\n";

switch (code)

{

case PRINT\_ARRAY:

PrintArray(array, arrayBorder);

break;

case TASK\_1:

IncreaseArrayElements(array, arrayBorder);

break;

case TASK\_2:

CheckSignAlternation(array, arrayBorder);

break;

case TASK\_3:

FindZeroElement(array, arrayBorder);

break;

}

}

}

void FillArray(int\*\* array, int arrayBorder, int mode)

{

if (mode == 0)

{

cout << "Задайте матрицу\n";

}

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

{

array[i] = new int[arrayBorder] {};

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

{

if (mode == 0)

{

cin >> array[i][j];

}

else

{

array[i][j] = rand() % (MAXIMUM\_RANDOM\_NUMBER - MINIMUM\_RANDOM\_NUMBER + 1) + MINIMUM\_RANDOM\_NUMBER;

}

}

}

if (mode == 0)

{

cout << "\n";

}

}

void PrintArray(int\*\* array, int arrayBorder)

{

cout << " ";

for (int columnNumber = 0; columnNumber < arrayBorder; columnNumber++)

{

cout << setw(4) << columnNumber;

}

cout << "\n\n";

for (int row = 0; row < arrayBorder; row++)

{

cout << setw(2) << row << " ";

for (int column = 0; column < arrayBorder; column++)

{

cout << setw(4) << array[row][column];

}

cout << '\n';

}

cout << "\n";

}

int SumOfDigits(int number)

{

if (abs(number) < 10)

{

return abs(number);

}

else

{

return abs(number) % 10 + abs(number / 10);

}

}

int FindSideDiagonalSameRowNumber(int\*\* array, int lengthOfRow, int numberOfRow)

{

return array[numberOfRow][lengthOfRow - numberOfRow - 1];

}

void IncreaseArrayElements(int\*\* array, int arrayBorder)

{

for (int row = 0; row < arrayBorder; row++)

{

for (int column = 0; column < arrayBorder; column++)

{

if (column != arrayBorder - row - 1)

{

array[row][column] += SumOfDigits(FindSideDiagonalSameRowNumber(array, arrayBorder, row));

}

}

array[row][arrayBorder - row - 1] += SumOfDigits(array[row][arrayBorder - row - 1]);

}

}

void CheckSignAlternation(int\*\* array, int arrayBorder)

{

int row{};

cout << "Введите номер строки: ";

cin >> row;

if (row < 0 || row >= arrayBorder)

{

cout << "Ввод несуществующей строки\n";

return;

}

bool plusIsFound = false;

bool minusIsFound = false;

for (int column = 0; column < arrayBorder; column++)

{

if (array[row][column] > 0)

{

plusIsFound = true;

}

else if(array[row][column] < 0)

{

minusIsFound = true;

}

}

if (plusIsFound && minusIsFound)

{

cout << "Произошло чередование знаков\n";

}

else

{

cout << "Чередование знаков не произошло\n";

}

cout << "\n";

}

void FindZeroElement(int\*\* array, int arrayBorder)

{

int foundRow = -1;

int foundColumn = -1;

int firstRowOfSearch = arrayBorder / 3;

cout << "Первая строчка поиска: " << firstRowOfSearch << '\n';

for (int row = firstRowOfSearch; row < arrayBorder; row++)

{

for (int column = 0; column < arrayBorder; column++)

{

if (array[row][column] == 0)

{

foundRow = row;

foundColumn = column;

break;

}

}

if (foundRow != -1)

{

break;

}

}

if (foundRow != -1)

{

cout << "Первый нулевой элемент находится в " << foundRow << " строке " << foundColumn << " столбце\n";

}

else

{

cout << "Нулевых элементов в области поиска нет\n";

}

cout << "\n";

}