**ФУНКЦИЯ ПОИСКА МАКСИМАЛЬНОГО ЭЛЕМЕНТА ПОБОЧНОЙ ДИАГОНАЛИ ЧЁТНОЙ КВАДРАТНОЙ МАТРИЦЫ С ПАРАМЕТРОМ В ФУНКЦИИ**

#include<iostream>

#include<ctime>

#include<limits.h>

using namespace std;

int search(int \*arr[], const int SIZE)

{

int MAX = arr[0][SIZE];

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

if (arr[i][SIZE - 1 - 1] > MAX)

MAX = arr[i][SIZE - 1 - 1];

return MAX;

}

int main()

{

int \*\*arr, SIZE=5;

arr = new int\* [SIZE];

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

arr[i] = new int[SIZE];

srand(time(NULL));

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

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

arr[i][j] = rand() % 100;

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

{

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

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

cout << endl;

}

cout << endl;

cout << search(arr,SIZE) << endl;

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

delete[]arr[i];

delete[]arr;

system("pause");

return 0;

}

//////////////////////////////////////////////////////////

Написать фрагмент программы для вывода на экран матрицы:

1 0 0 0 0 0 0 0 0

0 2 0 0 0 0 0 0 0

0 0 3 0 0 0 0 0 0

..........................

0 0 0 0 0 0 0 0 9

int main()

{

int \*\*arr, SIZE=9;

arr = new int\* [SIZE];

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

arr[i] = new int[SIZE];

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

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

arr[i][j] = 0;

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

arr[i][i] = i+1;

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

{

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

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

cout << endl;

}

cout << endl;

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

delete[]arr[i];

delete[]arr;

system("pause");

return 0;

}

////////////////////////////////////////

ТРЕУГОЛЬНЯ ДИНАМИЧЕСКАЯ МАТРИЦА

1

2 2

3 3 3

N N N N N

int main()

{

int \*\*arr, SIZE;

cin >> SIZE;

arr = new int\* [SIZE];

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

for (int j = i; j>=0; j--)

arr[j] = new int[SIZE];

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

for (int j = i; j >= 0; j--)

arr[i][j] = i+1;

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

{

for (int j = i; j >= 0; j--)

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

cout << endl;

}

cout << endl;

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

for (int j = i; j >= 0; j--)

delete[] arr[j];

delete[]arr;

system("pause");

return 0;

}

Написать программу формирования квадратной матрицы размерности N (N>2) по следующему правилу: если N – нечетное, то в центре матрицы ставится 1, если N – четное, то четыре центральных элемента равны 1. Остальные элементы получаются как сумма номера строки и столбца (не индекса!!!). Память выделять динамически. Вывести полученную матрицу на экран.

int main()

{

int\*\* arr, SIZE = 8;

arr = new int\* [SIZE];

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

arr[i] = new int[SIZE];

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

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

arr[i][j] = i+1+j+1;

if ((SIZE % 2) != 0)

arr[SIZE / 2][SIZE / 2] = 1;

else

for (int i = SIZE / 2 - 1; i <= SIZE / 2; i++)

for (int j = SIZE / 2 - 1; j <= SIZE / 2; j++)

arr[i][j] = 1;

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

{

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

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

cout << endl;

}

cout << endl;

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

delete[]arr[i];

delete[]arr;

system("pause");

return 0;

}

/////////////////////////////////////////////////////////////

НАПИСАТЬ ФРАГМЕНТ ПРОГРАММЫ ПРЕОБРАЗОВНИЯ ОДНОМЕРНОГО МАССИВА ПО СЛЕДУЮЩЕМУ ПРАВИЛУ

ПЕРВЫЙ И ПОСЛЕДНИЙ ЭЛЕМЕНТ ОСТАВИТЬ БЕЗ ИЗМЕНЕНИЯ,А ВСЕ ОСТАЛЬНЫЕ ЗАМЕНИТЬ ЧАСТНЫМ ОТ ДЕЛЕНИЯ ЛЕВОГО НА ПРАВЫЙ

int main()

{

int const SIZE = 10;

double arr[SIZE],carr[SIZE], temp;

srand(time(NULL));

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

arr[i] = rand() % (101 - 200 + 1) + 101;

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

cout << arr[i] << "\t";

cout << endl;

carr[0] = arr[0];

carr[SIZE - 1] = arr[SIZE - 1];

for (int i = 1; i < SIZE - 1; i++)

carr[i] = arr[i - 1] / arr[i + 1];

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

arr[i] = carr[i];

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

cout << arr[i] << "\t";

cout << endl;

return 0;

}

Нарисовать блок-схему слияния двух упорядоченных по возрастанию массивов вещественных чисел.

#include <iostream>

using namespace std;

//функция, сливающая массивы

void Merge(int\* A, int first, int last, int\* mas) {

int middle, part1, part2, j;

// Начало массива А.

middle = (first + last) / 2; //вычисление среднего элемента A

part1 = first; //начало левой части A

part2 = middle; //начало правой части A

for (j = first; j < last; j++) { //выполнять от начала до конца A

if (part1 >= middle) {

mas[j] = A[part2++];

continue;

}

if (part2 >= last) {

mas[j] = A[part1++];

continue;

}

if (A[part1] < A[part2])

mas[j] = A[part1++];

else

mas[j] = A[part2++];

}

//возвращение результата в список

for (j = first; j < last; j++) A[j] = mas[j];

};

//рекурсивная процедура сортировки

void \_MergeSort(int\* A, int first, int last, int\* mas) {

{

if (last - first > 1) {

\_MergeSort(A, first, (first + last) / 2, mas);

\_MergeSort(A, (first + last) / 2, last, mas);

Merge(A, first, last, mas); //слияние двух частей

}

}

};

//нерекурсивная процедура сортировки

void MergeSort(int\* A, int first, int last) {

int\* \_mas = new int[100]; // используем для слияния постоянный буфер

\_MergeSort(A, first, last, \_mas);

delete[] \_mas;

}

//главная функция

int main() {

setlocale(LC\_ALL, "Rus");

int n, m, i, l, q; int p = 0;

int\* A = new int[100], \* B = new int[100], \* C = new int[100];

cout << "Размер массива A > "; cin >> n;

cout << "Размер массива B > "; cin >> m;

cout << "Размер массива С > "; q = n + m; cout << q << endl;

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

cout << i << " элемент A > "; cin >> A[i];

}

for (l = 0; l < m; l++) {

cout << l << " элемент B > "; cin >> B[l];

}

MergeSort(A, 0, n); //вызов сортирующей процедуры

MergeSort(B, 0, m); //вызов сортирующей процедуры

cout << "Упорядоченный массив A: "; //вывод упорядоченного массива A

for (i = 0; i < n; i++) cout << A[i] << " ";

cout << endl;

cout << "Упорядоченный массив B: "; //вывод упорядоченного массива B

for (l = 0; l < m; l++) cout << B[l] << " ";

cout << endl;

cout << "Упорядоченный массив C: "; //вывод упорядоченного массива C

i = l = 0;

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

if (i >= n) { // массив A уже весь обработан

C[j] = B[l++];

continue;

}

if (l >= m) { // массив B уже весь обработан

C[j] = A[i++];

continue;

}

if (A[i] < B[l])

C[j] = A[i++];

else

C[j] = B[l++];

}

for (p = 0; p < q; p++) cout << C[p] << " ";

cout << endl;

delete[]A; delete[]B; delete[]C;

system("pause");

}

.......................................................

int main()

{

int a = 200;

const int \*p1 = &a; //p1 - указатель на int-константу,

//разыменованным указателем значение изменять нельзя.

//\*p1 = 300; //Это будет ошибкой.

int const \*p2 = &a; //p2 - указатель на int-константу,

//разыменованным указателем значение изменять нельзя.

//\*p2 = 300; //Это будет ошибкой.

const int \*p3 = &a; //p3 - указатель на int-константу,

//разыменованным указателем значение изменять нельзя

//\*p3 = 300; //Это будет ошибкой.

int const \* const p4 = &a; //p4 - константный указатель на int-константу.

//Нельзя записать в p4 никакие иные адреса, кроме сейчас записываемого.

//разыменованным указателем значения изменять нельзя.

int b = 100;

// p4 = &b; //Это будет ошибкой. Нельзя изменить адрес внутри указательной переменной.

//\*p4 = 100; //Это будет ошибкой. Нельзя изменить значение разыменованным указателем.

const int \* const p5 = &a; //Попробуйте сами

......................................................................

//Borland C++3.1 Указатели и константы Листинг #1

#include <iostream.h>

int main()

{

int a = 100; //Два обычных объекта типа int

int b = 222;

int \*const P2 = &a; //Константный указатель {P2} указывает на переменную {a}, тип которой int.

\*P2 = 987; //Менять значение разрешено.

//P2 = &b; //Но изменять адрес не разрешается.

const int \*P1 = &a; //Указатель на константу int.

//\*P1 = 110; //Менять значение нельзя.

P1 = &b; //Но менять адрес разрешено.

const int \*const P3 = &a; //Константный указатель на константу int.

//\*P3=155; //Изменять нельзя ни значение,

//P3=&b; //ни адрес, к которому такой указатель привязан.

cin.get();

}

}