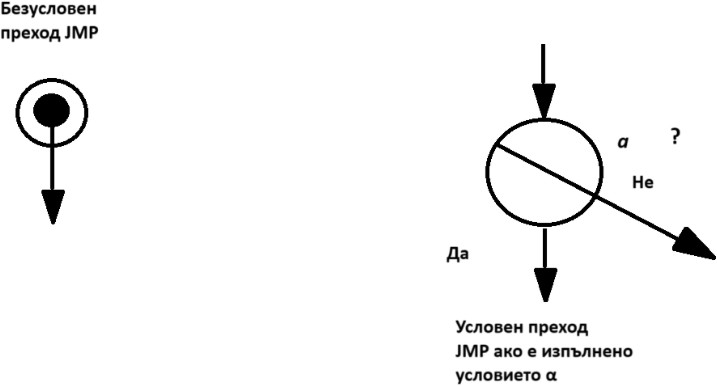
# Христо Минчев F113042

# Домашна работа 1

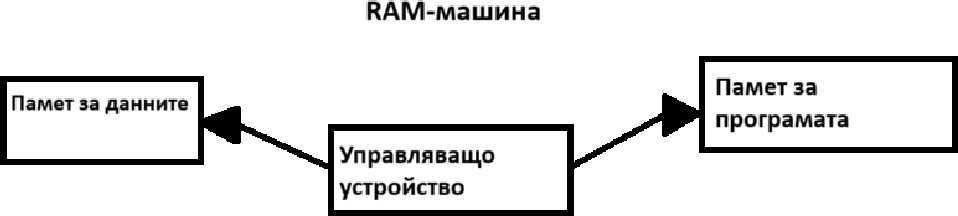
1.

Алгоритъмът е такова крайно, дискретно (постъпково), детерминирано преобразование, което, приложено над произволен допустим набор от стойности на входното множество, довежда до получаването на единствен набор от допустими стойности на изходното множество.

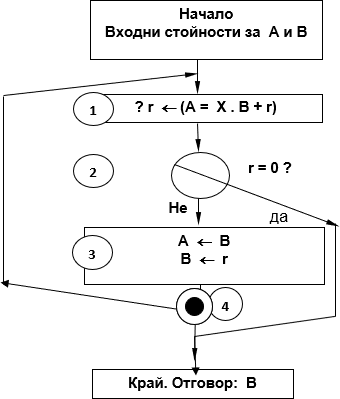
2.



3.



4.



# Домашна работа

2

Задача 1:

#include <iostream> using namespace std;

int main() {

int a,b,r;

cout << "Enter the value for a and b:"; cin >> a >> b;

start\_label: r = a%b;

if (r == 0)

{

goto end\_label;

}

else{

a = b;

b = r;

goto start\_label;

}

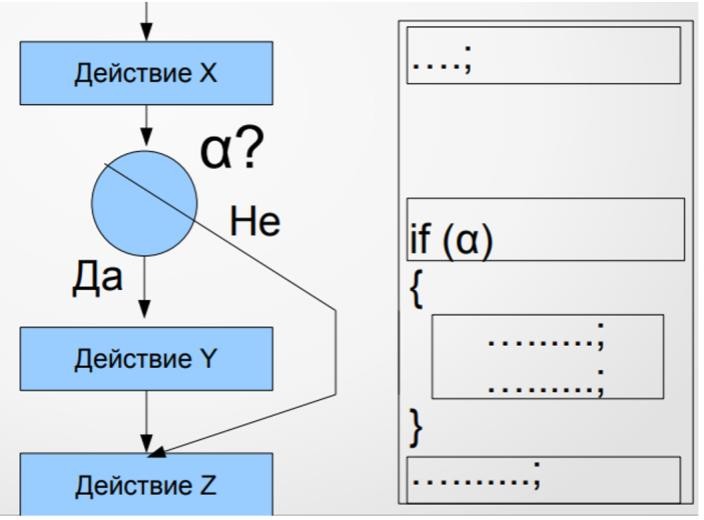
end\_label:

cout << b << " se sudurja cql broi puti v " << a << " i v " << b;

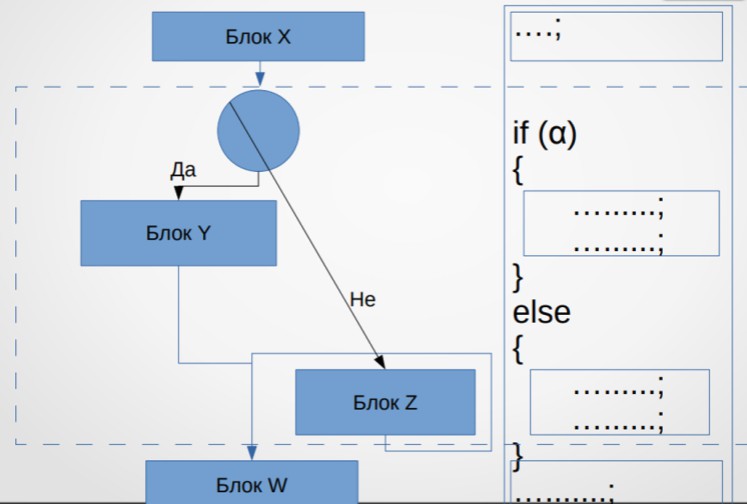
return 0;

}

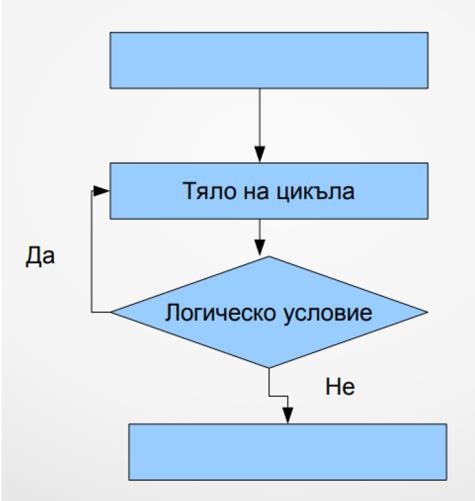
Задача 2: Байпас



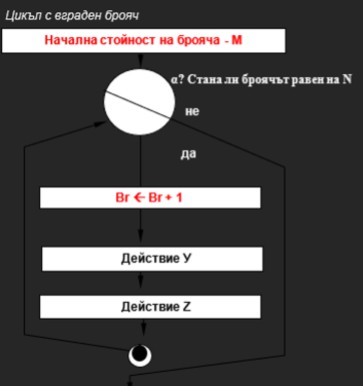
Двуклон



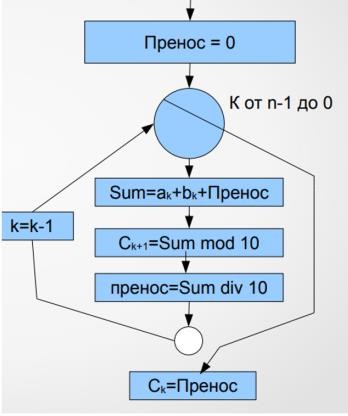
Цикъл с предусловие

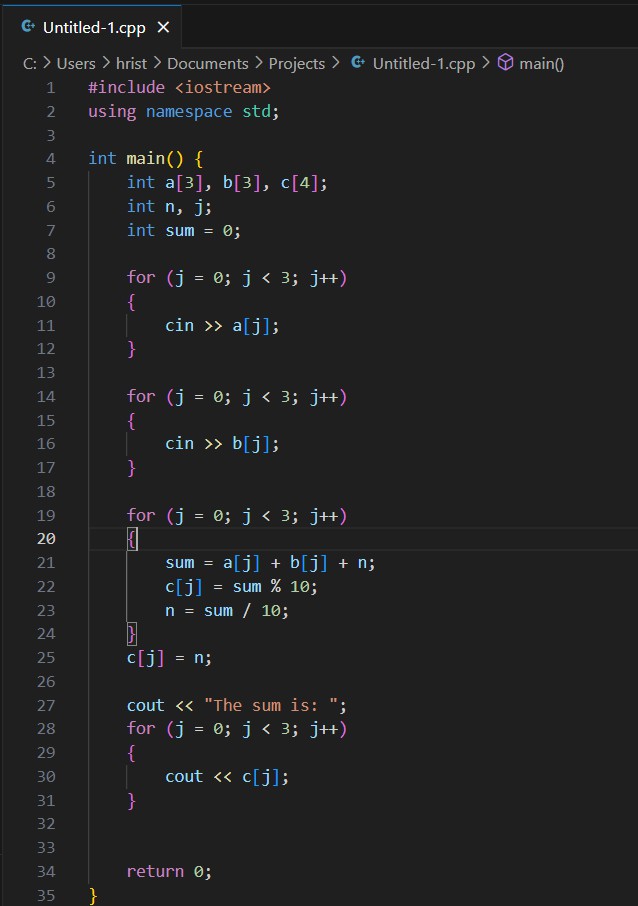


Цикъл с вграден брояч

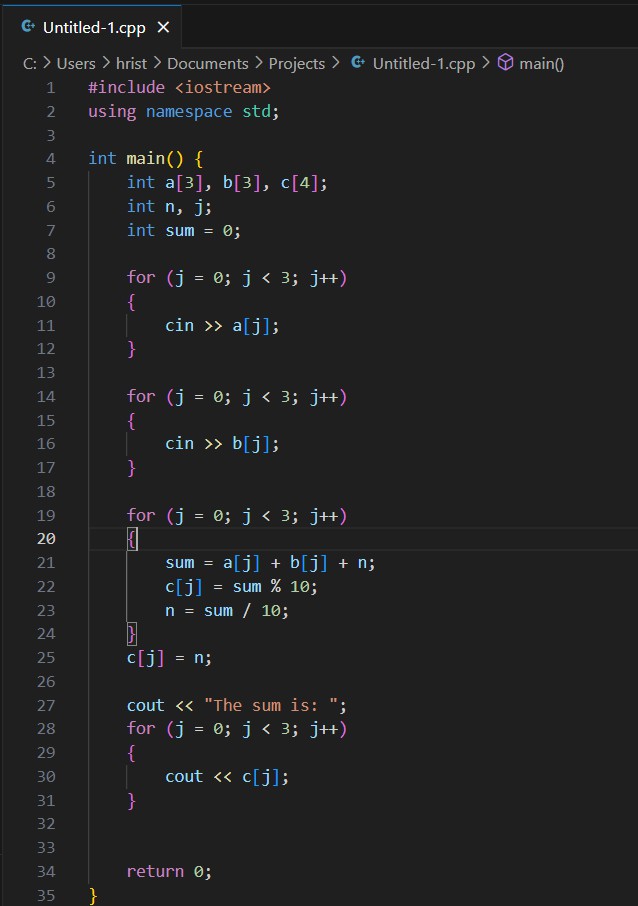


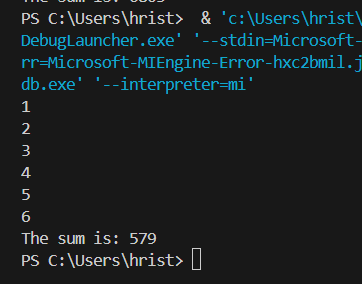
Задача 3:





Задача 4:





# Домашна работа

3

Задача 1:

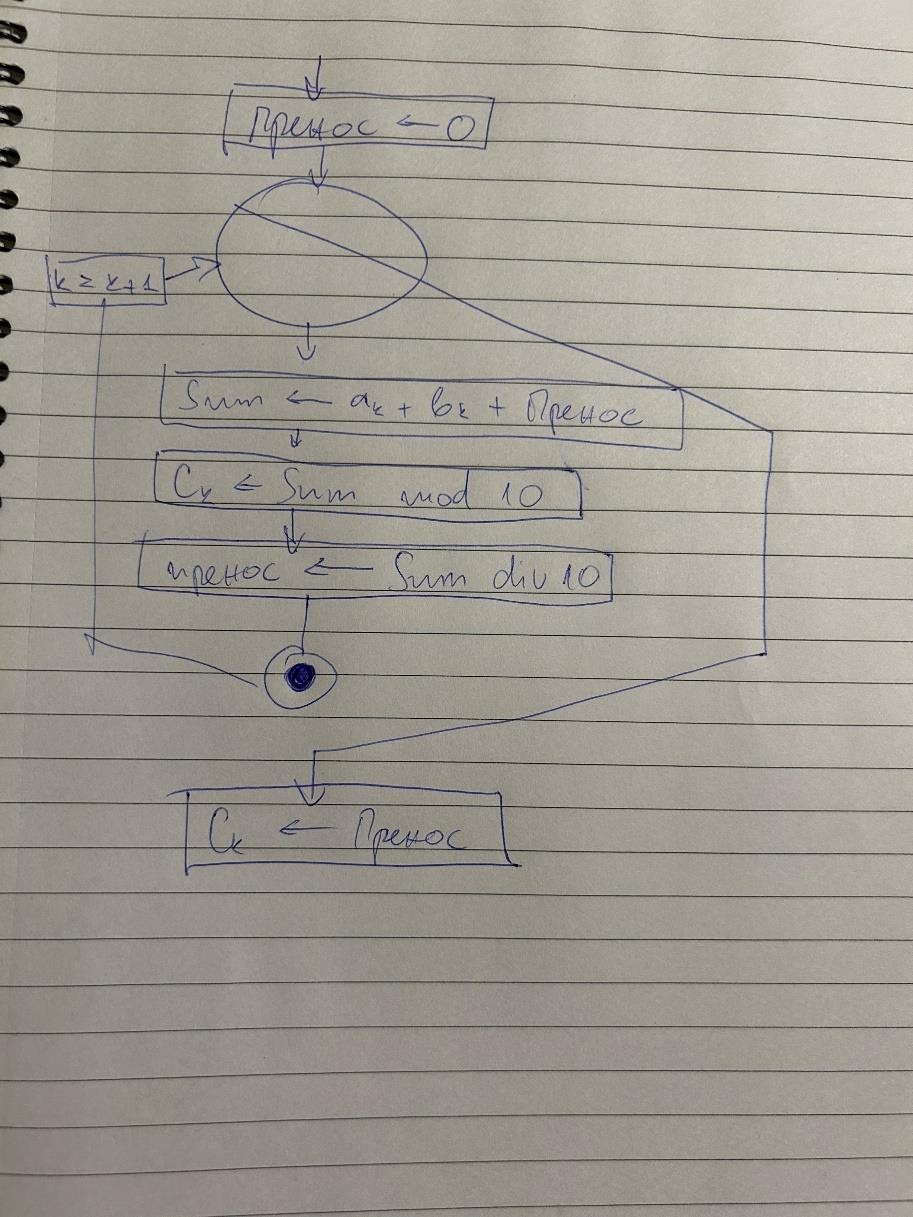
- остатъците са 7 (от 0 до 6)

- 2

- 7

- 9

Задача 2:



#include <iostream> using namespace std;

int main() {

int a[3], b[3], c[4]; int n = 0;

int j;

int sum = 0;

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

{

cin >> a[j];

}

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

{

cin >> b[j];

}

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

{

sum = a[j] + b[j] + n; c[j] = sum % 10;

n = sum / 10;

}

c[j] = n;

cout << "The sum is: "; for (j = 0; j < 3; j++)

{

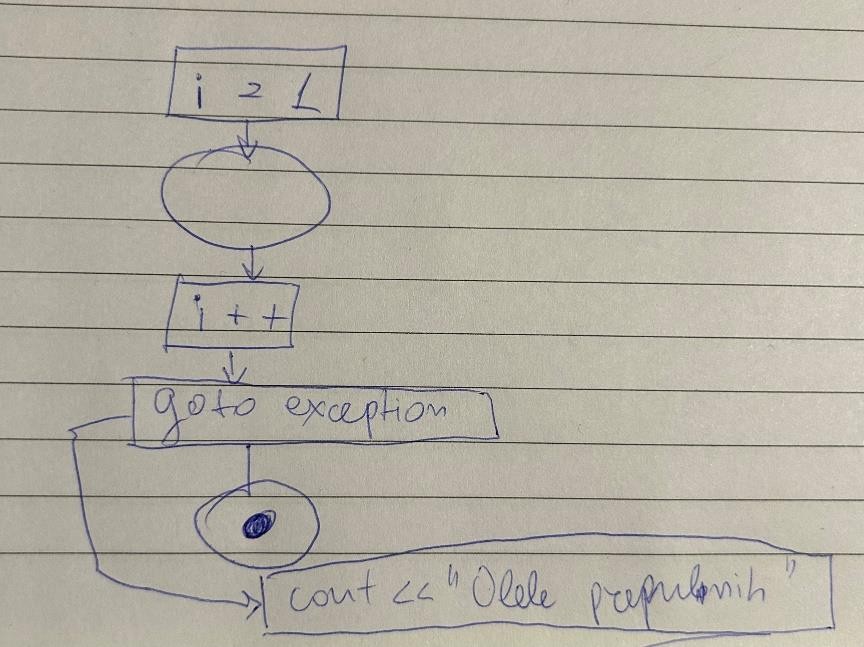
cout << c[j];

}

return 0;

}

Задача 3:



#include <iostream> using namespace std;

int main()

{

int i = 1;

for ( ; ; i++)

{

if (i < 0)

{

goto exception;

}

}

exception:

cout << "Olele prepulnih: " << i << endl;

return 0;

}

# Домашна работа

4

Задача 1:

𝑛

∑ 𝑖 =

𝑖=1

(𝑛 + 1). 𝑛 2

F113042 – Факултетния ми номер

(2 + 1). 2

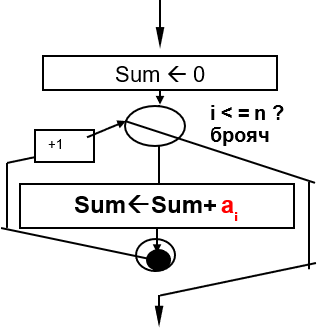
= 3

2

Задача 2:

 *i* 2

*n*

*i*1

#include <iostream> using namespace std;

int main() {

int n, i; int sum = 0; cin >> n;

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

{

sum = sum + i\*i;

}

cout << sum << endl; return 0;

}

*i* 2

*n*

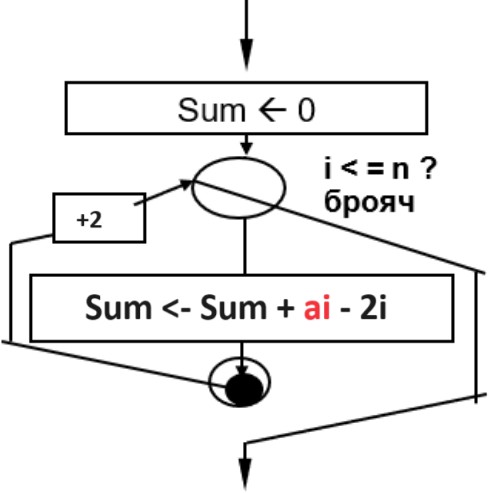
 2*i*

*i*2

#include <iostream> using namespace std;

int main() {

int n, i;

int sum = 0; cin >> n;

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

{

sum += i\*i - 2\*i;

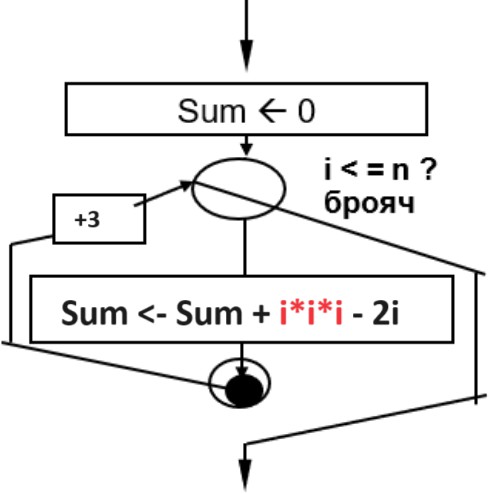
}

cout << sum << endl; return 0;

}

*i*3  2*i*

*n*

*i*3

#include <iostream> using namespace std;

int main() {

int n, i; int sum = 0; cin >> n;

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

{

sum += i\*i\*i - 2\*i;

}

cout << sum << endl; return 0;

}

Задача 3:

using namespace std;

int main() {

int n;

long factorial = 1;

cout << "Enter an integer: "; cin >> n;

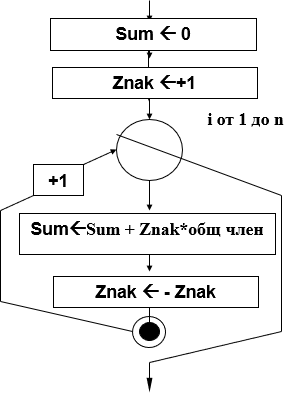
for (int i = 1; i <= n; ++i) { factorial \*= i;

}

cout << "Factorial of " << n << " is: " << factorial <<endl; return 0;

}

Задача 4:



#include <iostream> #include <stdlib.h> using namespace std;

int main() {

double count = 1; double result = 0;

double result\_before = 1; double precision = 0.0000001; int sign = 1;

while (abs(result - result\_before) > precision)

{

result\_before = result;

result += sign \* (1 / (2 \* count - 1)); sign \*= -1;

count++;

}

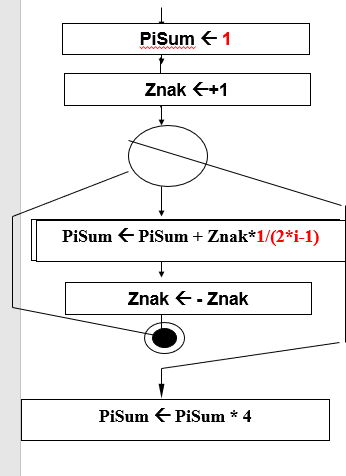
cout << result \* 4 << endl; return 0;

}

# Домашна работа

5

Задача 1:



#include <iostream> #include <cmath> using namespace std;

int main() {

double result = 0; double result\_before = 1; int sign = 1;

for (double count = 1; count <= 10000; count++) { result\_before = result;

result += sign \* (1 / (2 \* count - 1)); sign \*= -1;

}

cout << result \* 4 << endl; return 0;

}

#include <iostream> #include <cmath> using namespace std;

int main() {

double result = 0; double result\_before = 1;

double precision = 0.0001; int sign = 1;

int count = 1;

while (abs(result - result\_before) >= precision) { result\_before = result;

result += sign \* (1.0 / (2 \* count - 1)); sign \*= -1;

count++;

}

cout << result \* 4 << endl; return 0;

}

#include <iostream> #include <cmath> using namespace std;

int main() {

double result = 0; double result\_before = 1;

double precision = 0.0001; int sign = 1;

int count = 1;

start:

if (abs(result - result\_before) >= precision) { result\_before = result;

result += sign \* (1.0 / (2 \* count - 1)); sign \*= -1;

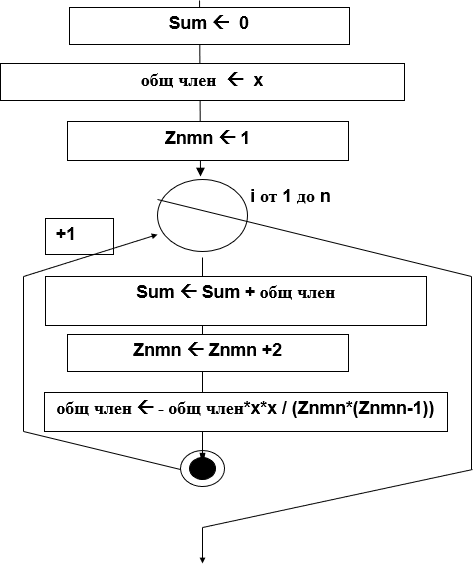
count++; goto start;

}

cout << result \* 4 << endl; return 0;

}

Задача 2:



Първи начин:

#include <iostream> #include <cmath> using namespace std;

unsigned long long factorial(int n) { long fact = 1;

for (int i = 1; i <= n; ++i) { fact \*= i;

}

return fact;

}

int main() {

double angleDegrees;

cout << "Vuvedete ugula v gradusi: "; cin >> angleDegrees;

double angleRadians = angleDegrees \* 3.14159265358979323846 / 180.0; cout << "Ugulut v radiani: " << angleRadians << endl;

double sinX = 0; int n = 0;

double result = angleRadians;

while (result != 0) { sinX += result;

++n;

result \*= -(angleRadians \* angleRadians) / ((2 \* n) \* (2 \* n + 1));

}

cout << "sin(" << angleDegrees << "): " << sinX << endl;

cout << n << " chlena sa dostignati predi prepulvane na tochnostta." << endl;

return 0;

}

Втори начин:

#include <iostream> #include <cmath>

using namespace std;

int main() {

double angleDegrees;

cout << "Vuvedete ugula v gradusi: "; cin >> angleDegrees;

double angleRadians = angleDegrees \* 3.14159265358979323846 / 180.0; cout << "Ugulut v radiani: " << angleRadians << endl;

double sinX = 0;

double precision = 0.000001; int n = 0;

double result = angleRadians;

while (abs(result) >= precision) { sinX += result;

++n;

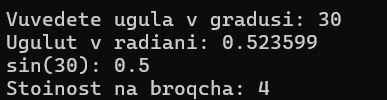
result \*= -(angleRadians \* angleRadians) / ((2 \* n) \* (2 \* n + 1));

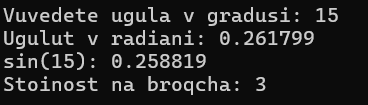
}

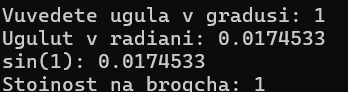
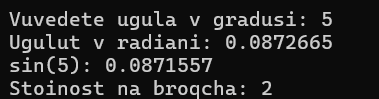
cout << "sin(" << angleDegrees << "): " << sinX << endl; cout << "Stoinost na broqcha: " << n << endl;

return 0;

}



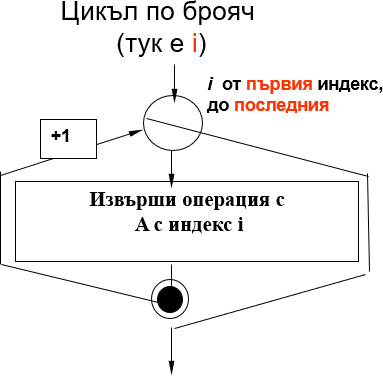




# Домашна работа

6

Задача 1:



#include <iostream> using namespace std;

int main() {

int facultyNumber[6]; int x = 0;

cout << "Vuvedete fakulteten nomer: "; for (int i = 0; i < 6; ++i) {

cin >> facultyNumber[i];

}

for (int i = 0; i < 6; ++i) {

if (facultyNumber[i] == 1 || facultyNumber[i] == 2 || facultyNumber[i]

== 3 || facultyNumber[i] == 4) {

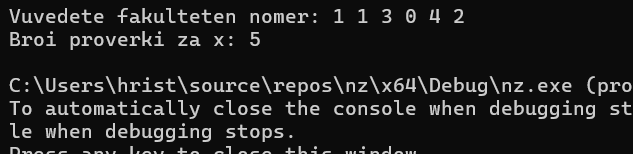
++x;

}

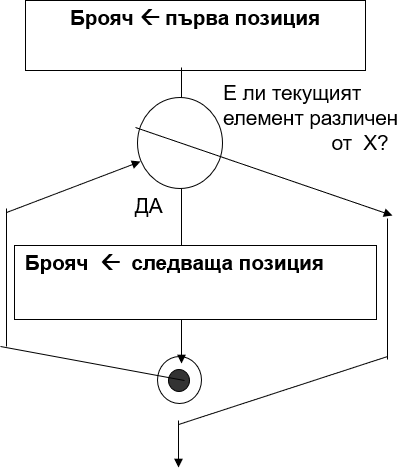
}

cout << "Broi proverki za x: " << x << endl; return 0;

}



Задача 2:



#include <iostream>

using namespace std;

int main() {

char lastName[] = { 'M', 'i', 'n', 'c', 'h', 'e', 'v'}; char targetValues[] = { 'a', 'b','v', 'g'};

int x = 0;

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

char target = targetValues[i]; if (lastName[i] == target) {

++x;

break;

}

++x;

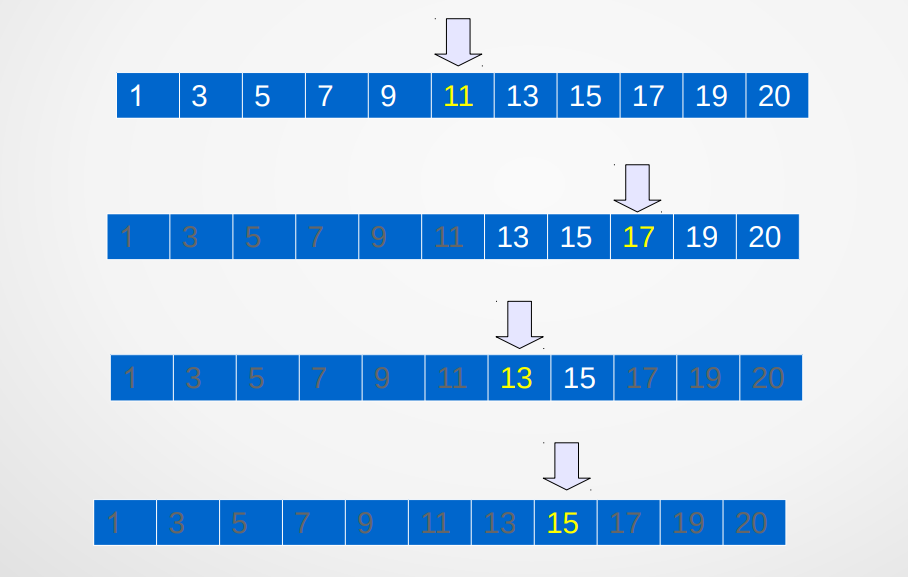
}

cout << "Broi proverki za x: " << x << endl; return 0;

}



Задача 3:



#include <iostream> #include <algorithm> using namespace std;

int binarySearch(char arr[], int l, int r, char x, int& countX) { while (l <= r) {

int m = l + (r - l) / 2; if (arr[m] == x)

return m;

++countX;

if (arr[m] < x) l = m + 1;

else

r = m - 1;

}

return -1;

}

int main() {

char FacName[] = { '1', '1', '3', '0', '4', '2', 'm', 'i', 'n', 'c', 'h',

'e', 'v', 'q', '9' }; // Addnal sum oshte dva elementa nakraq, poneje

elementite v fak.nomera i familiqta mi ne figurirat v masiva s elementite koito se tursqt int size = sizeof(FacName) / sizeof(FacName[0]);

cout << "Masiva: ";

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

{

cout << FacName[i];

}

cout << " " << endl; sort(FacName, FacName + size); cout << "Masiva sortiran: " ; for (int i = 0; i < size; i++)

{

cout << FacName[i];

}

cout << " " << endl;

char targets[] = { '9', '8', '7', 'q', 'u', 'y', 'o' }; int target = sizeof(targets) / sizeof(targets[0]);

int countX = 0;

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

int result = binarySearch(FacName, 0, size - 1, targets[i], countX); if (result != -1) {

cout << "Stoinostta " << targets[i] << " se namira na poziciq " << result << endl;

}

else {

cout << "Stoinostta " << targets[i] << " ne e namerena" << endl;

}

}

cout << "Broi proverki za x: " << countX << endl;

return 0;

}

Задача 4:

#include <iostream> #include <ctime> #include <cstdlib> #include <algorithm> #include <chrono>

using namespace std;

double linearSearch(int arr[], int n, int target) {

auto start = std::chrono::high\_resolution\_clock::now(); for (int i = 0; i < n; ++i) {

if (arr[i] == target) { break;

}

}

auto end = std::chrono::high\_resolution\_clock::now();

return std::chrono::duration\_cast<std::chrono::nanoseconds>(end - start).count();

}

double anchorSearch(int arr[], int n, int target) {

auto start = std::chrono::high\_resolution\_clock::now();

arr[n] = target; int i = 0;

while (arr[i] != target) {

++i;

}

auto end = std::chrono::high\_resolution\_clock::now();

return std::chrono::duration\_cast<std::chrono::nanoseconds>(end - start).count();

}

double binarySearch(int arr[], int n, int target) {

auto start = std::chrono::high\_resolution\_clock::now(); int l = 0;

int r = n - 1; while (l <= r) {

int m = l + (r - l) / 2; if (arr[m] == target) {

break;

}

if (arr[m] < target) { l = m + 1;

}

else {

r = m - 1;

}

}

auto end = std::chrono::high\_resolution\_clock::now();

return std::chrono::duration\_cast<std::chrono::nanoseconds>(end - start).count();

}

int main() {

const int MAX\_SIZE = 100000;

int\* arr = new int[MAX\_SIZE + 1]; int\* binaryArr = new int[MAX\_SIZE]; srand(time(nullptr));

for (int i = 0; i < MAX\_SIZE; ++i) { arr[i] = rand() % 100000; binaryArr[i] = i + 1;

}

int targets[5];

cout << "Vuvedete 5 stoinosti koito da se tursqt: "; for (int j = 0; j < 5; ++j) {

cin >> targets[j];

}

double total\_duration\_linear = 0; double total\_duration\_anchor = 0; double total\_duration\_binary = 0;

for (int j = 0; j < 5; ++j) { double duration\_linear = 0; double duration\_anchor = 0; double duration\_binary = 0; for (int k = 0; k < 10; ++k) {

duration\_linear += linearSearch(arr, MAX\_SIZE, targets[j]); duration\_anchor += anchorSearch(arr, MAX\_SIZE, targets[j]); duration\_binary += binarySearch(binaryArr, MAX\_SIZE, targets[j]);

}

total\_duration\_linear += duration\_linear / 10; total\_duration\_anchor += duration\_anchor / 10;

total\_duration\_binary += duration\_binary / 10;

}

cout << "Srednoto vreme za obhojdane na celiq masiv e: " << total\_duration\_linear << " nanosekundi" << endl;

cout << " Srednoto vreme za obhojdane na masiva chrez kotva e: " << total\_duration\_anchor << " nanosekundi " << endl;

cout << " Srednoto vreme za obhojdane na masiva chrez dihotomichno tursene e: " << total\_duration\_binary << " nanosekundi " << endl;

delete[] arr; delete[] binaryArr;

return 0;

}

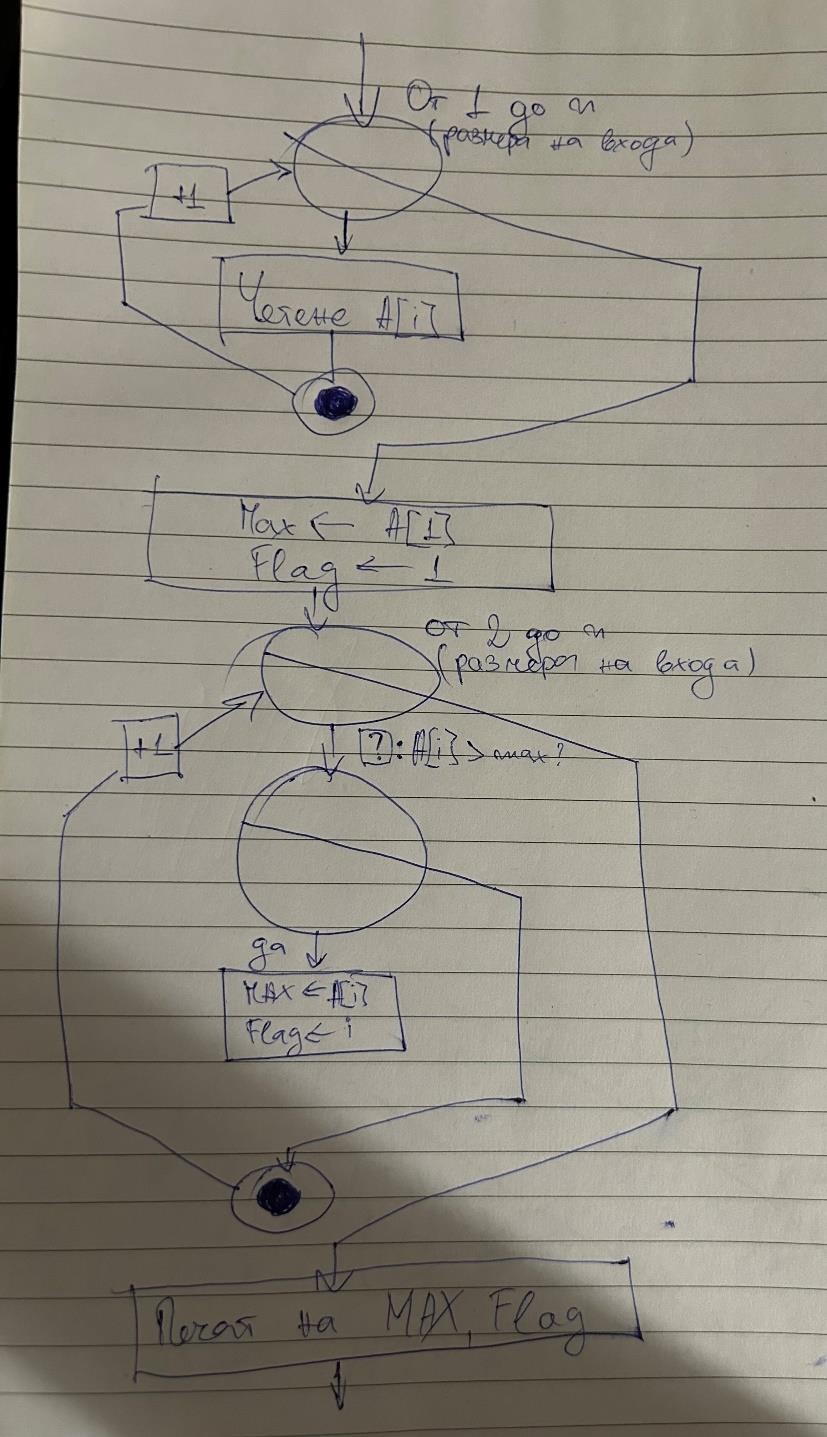
|  |  |  |  |
| --- | --- | --- | --- |
| Време за изпълнение | | | |
| Метод/брой подадени елементи | Търсене чрез обхождане на целия масив | Търсене до намиране с котва | Дихотомично търсене |
| 100 000 | 1.15509e+06  nanoseconds | 1.11439e+06  nanoseconds | 1750  nanoseconds |
| 200 000 | 2.32964e+06  nanoseconds | 2.23178e+06  nanoseconds | 4170  nanoseconds |
| 300 000 | 3.44767e+06  nanoseconds | 3.31268e+06  nanoseconds | 3680  nanoseconds |
| 400 000 | 4.29869e+06  nanoseconds | 3.99526e+06  nanoseconds | 3820  nanoseconds |
| 500 000 | 5.53944e+06  nanoseconds | 5.15638e+06  nanoseconds | 4810  nanoseconds |
| 600 000 | 6.347e+06  nanoseconds | 6.01713e+06  nanoseconds | 4270  nanoseconds |
| 700 000 | 7.48006e+06  nanoseconds | 7.22593e+06  nanoseconds | 4520  nanoseconds |

|  |  |  |  |
| --- | --- | --- | --- |
| 800 000 | 8.90818e+06  nanoseconds | 8.39432e+06  nanoseconds | 5540  nanoseconds |
| 900 000 | 9.38508e+06  nanoseconds | 9.05452e+06  nanoseconds | 5640  nanoseconds |
| 1 000 000 | 1.08371e+07  nanoseconds | 1.04133e+07  nanoseconds | 5400  nanoseconds |

# Домашна работа

7

Задача 1:



Алгоритъмът прави 2n - 1 сравнения

#include <iostream> using namespace std;

int main() {

int arr[6] = { 6, 8, 3, 7, 4, 9 };

int min = INT\_MAX; int minIndex = -1;

for (int i = 0; i < 6; ++i) { if (arr[i] < min) {

min = arr[i]; minIndex = i;

}

}

cout << "Minimum value is " << min << " at index " << minIndex << endl; return 0;

}



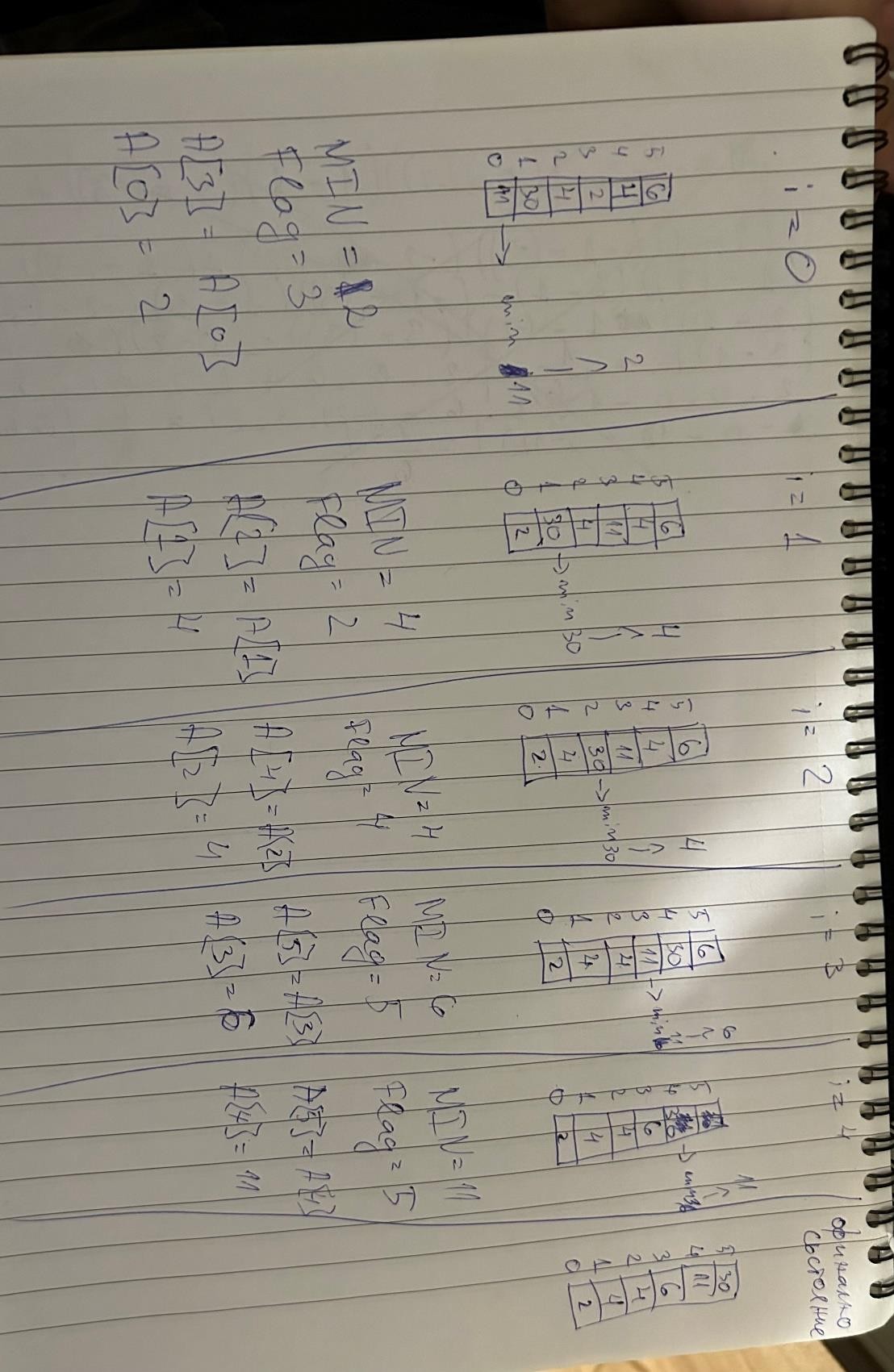
Задача 2:

а. Този алгоритъм за сортиране работи като обхожда всяка позиция от началото до края на масива. За всяка позиция избира най-малката стойност и я попълва на текущата позиция. След като е избрана минималната стойност, тя се дезактивира, така че да не бъде избрана отново при следващите обхождания. Процесът се повтаря за всеки елемент от масива.

б. Операторът, който позволява да се изтегля всеки път следващата по ред минимална стойност, е операторът за намиране на минимум. Той се използва за намиране на най-малката стойност в текущата част от масива и връща индекса на тази стойност.

в. Във всеки обхождащ цикъл се изтегля минималната стойност само веднъж. Ако масивът е с n елемента, минималната стойност се изтегля n пъти.

Задача 3:

Задача 4:

Задача 5:

#include <iostream> using namespace std;

int main() {

int arr[] = { 6, 8, 3, 7, 4, 9 };

int n = sizeof(arr) / sizeof(arr[0]);

for (int i = 0; i < n - 1; ++i) { int MIN = arr[i];

int flag = i; int compare = 0;

for (int j = i + 1; j < n; ++j) { compare++;

if (arr[j] < MIN) { MIN = arr[j]; flag = j;

}

}

int temp = arr[flag]; arr[flag] = arr[i]; arr[i] = temp;

cout << "Sustoqnie " << i << ": "; for (int k = 0; k < n; ++k) {

cout << arr[k]; if (k != n - 1)

cout << ",";

}

cout << endl;

cout << "obhojdane " << i + 1 << ", sravneniq " << compare << endl; cout << endl;

}

return 0;

}

# Домашна работа

10

|  |  |
| --- | --- |
| Задача 1: |  |
| A - [0,8,9,4 | B - [4,3 |
| 3,4,1,7 | 8,4 |
| 6,4,2,2] | 9,1 |
|  | 4,7] |

А. - C = | | |

| | |

| | |

Матрицата С има 3 реда и 2 стълба.

Б. - Индексирането на редовете на C ги бележа с i, а стълбовете с j.

В. – C[1,1]=0⋅4+8⋅8+9⋅9+4⋅4=0+64+81+16=161 C[1,2]= 0⋅3+8⋅4+9⋅1+4⋅7=0+32+9+28=69 C[2,1]= 3⋅4+4⋅8+1⋅9+7⋅4=12+32+9+28=81 C[2,2]= 3⋅3+4⋅4+1⋅1+7⋅7=9+16+1+49=75 C[3,1]= 6⋅4+4⋅8+2⋅9+2⋅4=24+32+18+8=82

C[3,2]= 6⋅3+4⋅4+2⋅1+2⋅7=18+16+2+14=50

C – [161, 69

81, 75

82, 50]

Г. - for (int i = 0; i < 3; i++) Д. - for (int i = 0; i < 3; i++) Е. - for (int i = 0; i < 3; i++) {

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

}

}

Ж. - На схемата горе редовете на матрицата А и В се бележат с индекси i, а стълбовете с индекси j.

З. - C[i][j] = ∑(k=1 до 3) A[i][k] \* B[k][j] И. - for (int k = 0; k < 3; ++k) {

C[i][j] += A[i][k] \* B[k][j];

}

Задача 2. –

#include <iostream>

using namespace std;

int main() {

int A[3][4] = { {0, 8, 9, 4},

{3, 4, 1, 7},

{6, 4, 2, 2} };

int B[4][2] = { {4, 3},

{8, 4},

{9, 1},

{4, 7} };

int C[3][2] = { 0 };

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

for (int j = 0; j < 2; j++) { for (int k = 0; k < 4; k++) {

C[i][j] += A[i][k] \* B[k][j];

}

}

}

cout << "C:" << endl;

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

for (int j = 0; j < 2; j++) { cout << C[i][j] << " ";

}

cout << endl;

}

return 0;

}