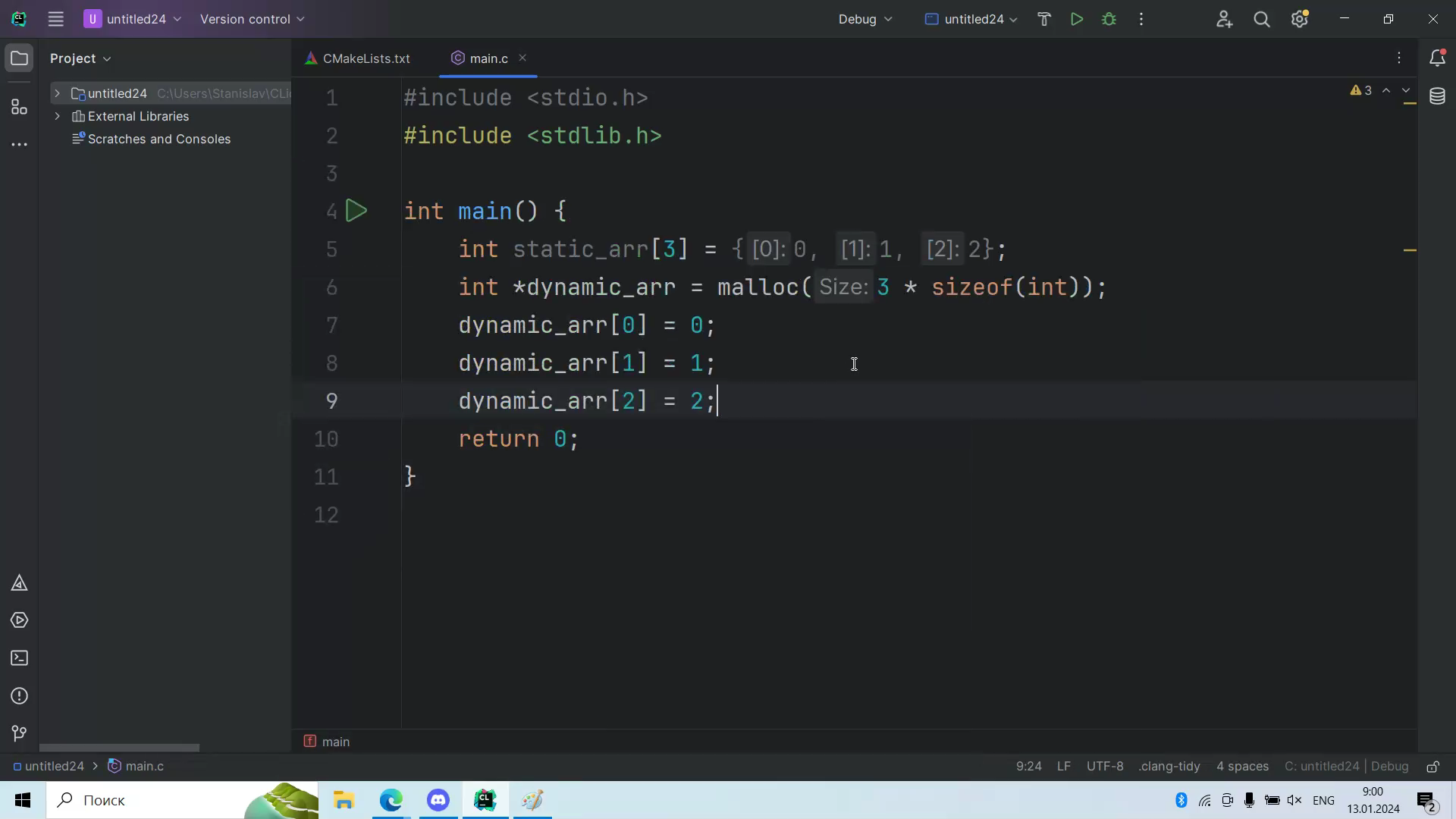
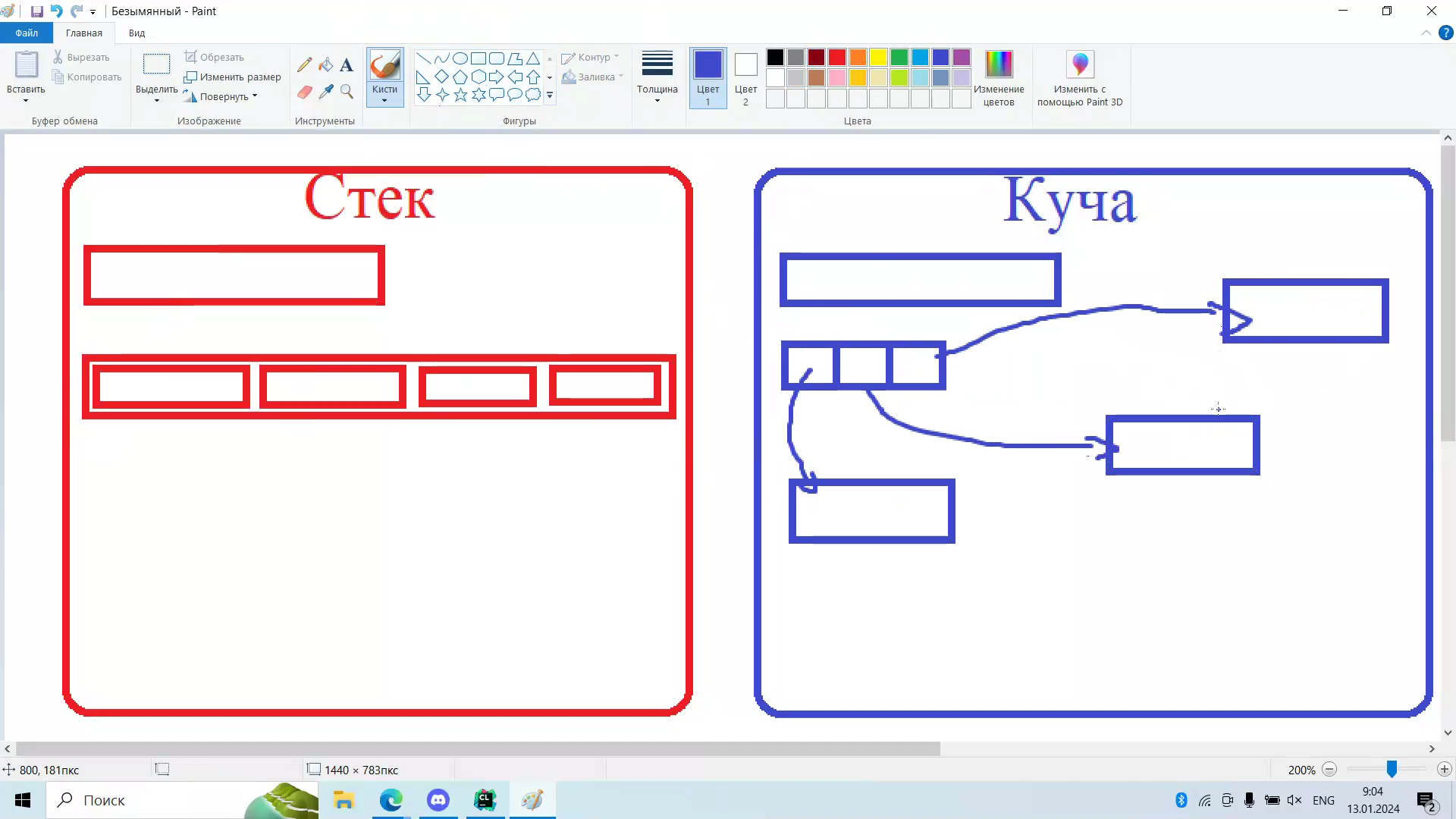
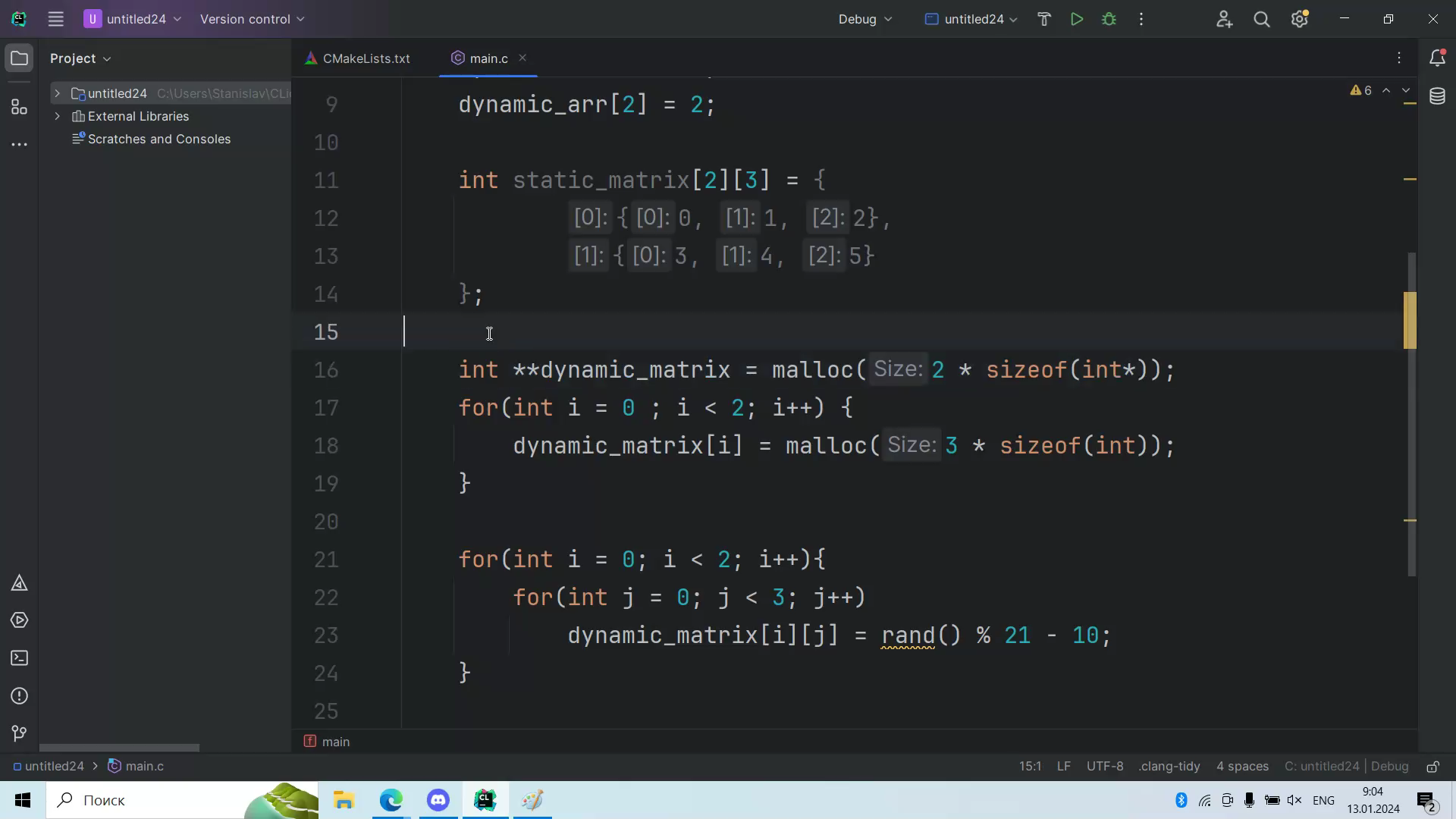
Двумерные массивы с подробностями

1 arr - стек 2 arr - куча

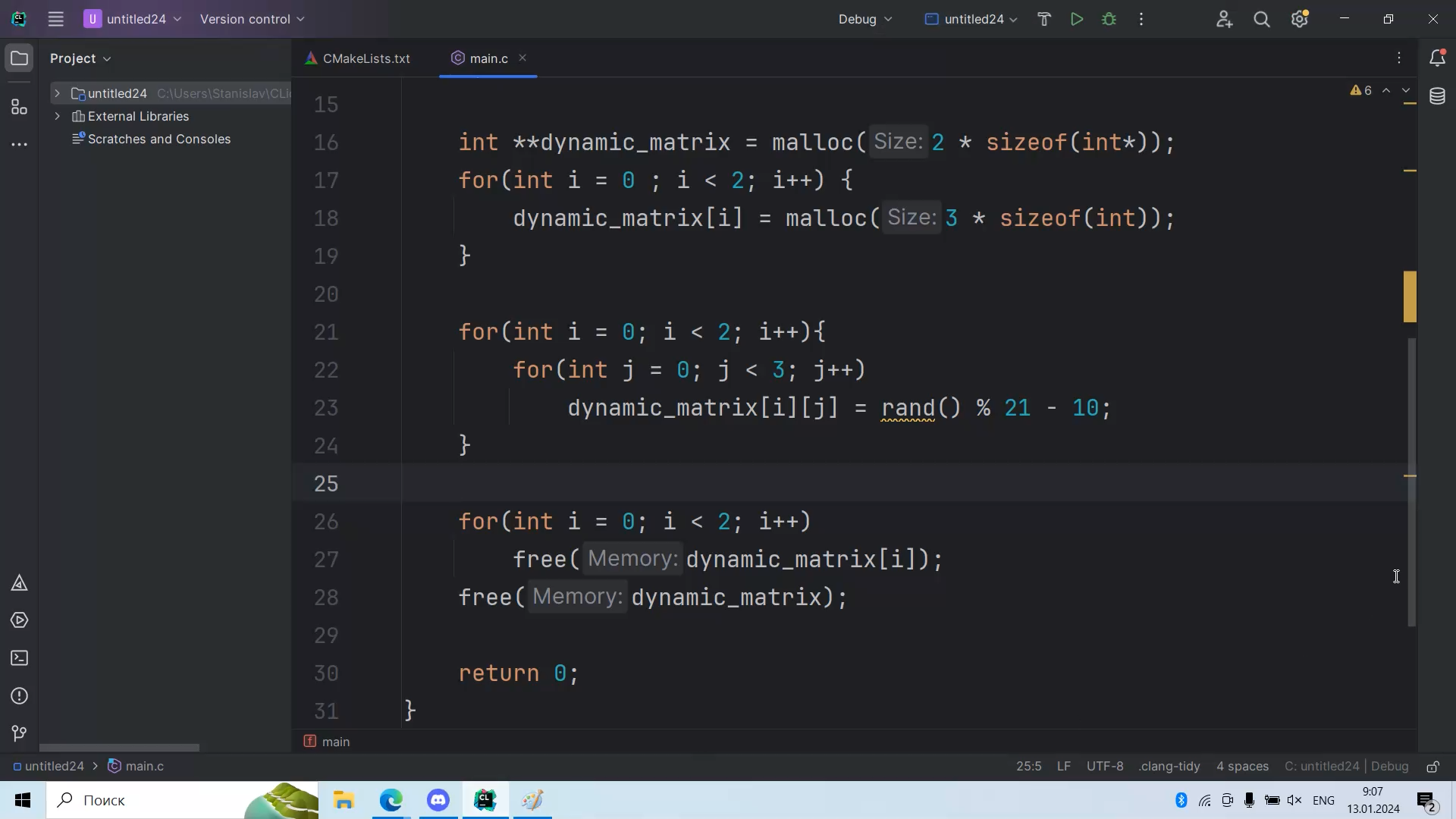


Двумерные массивы

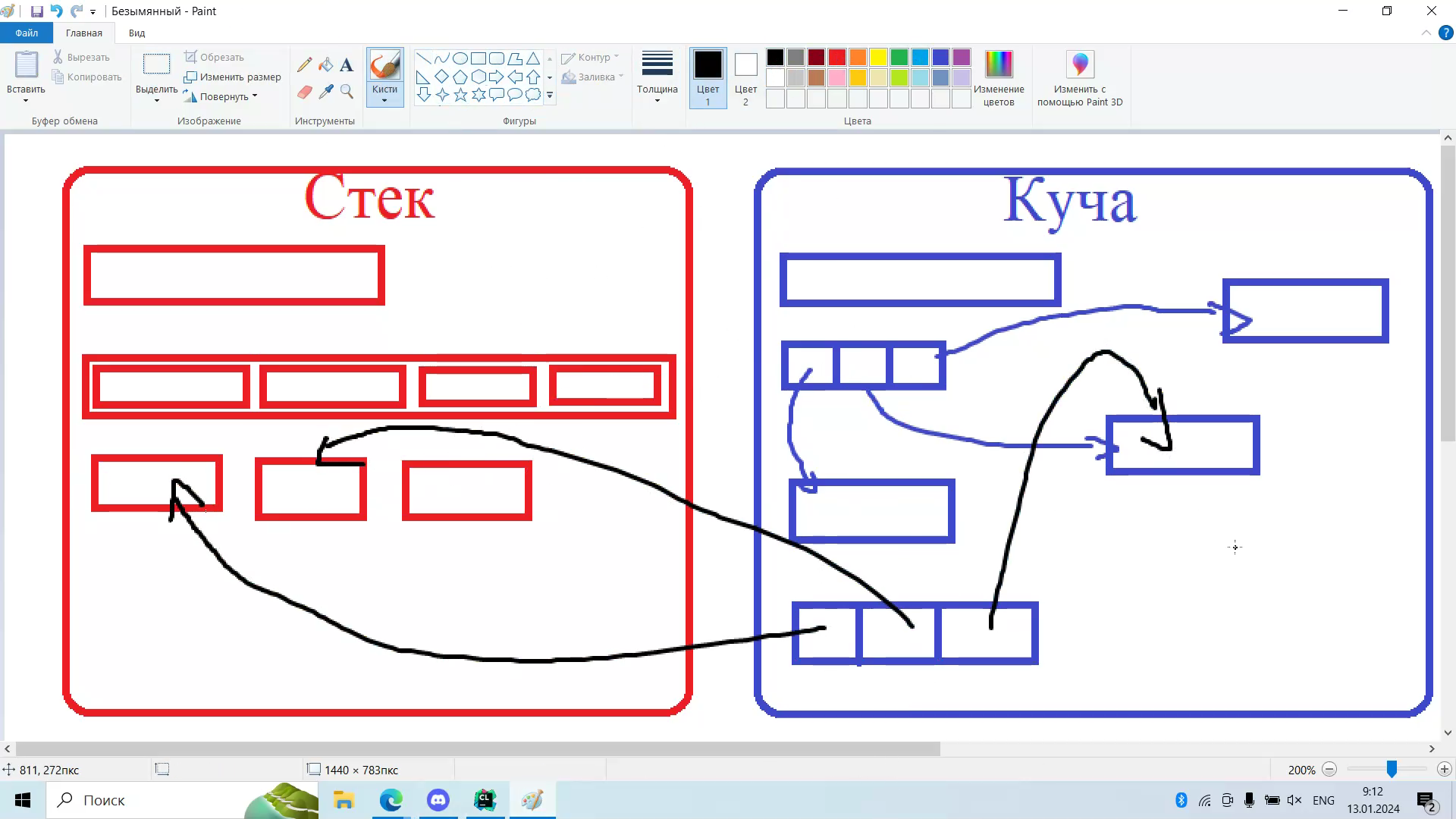


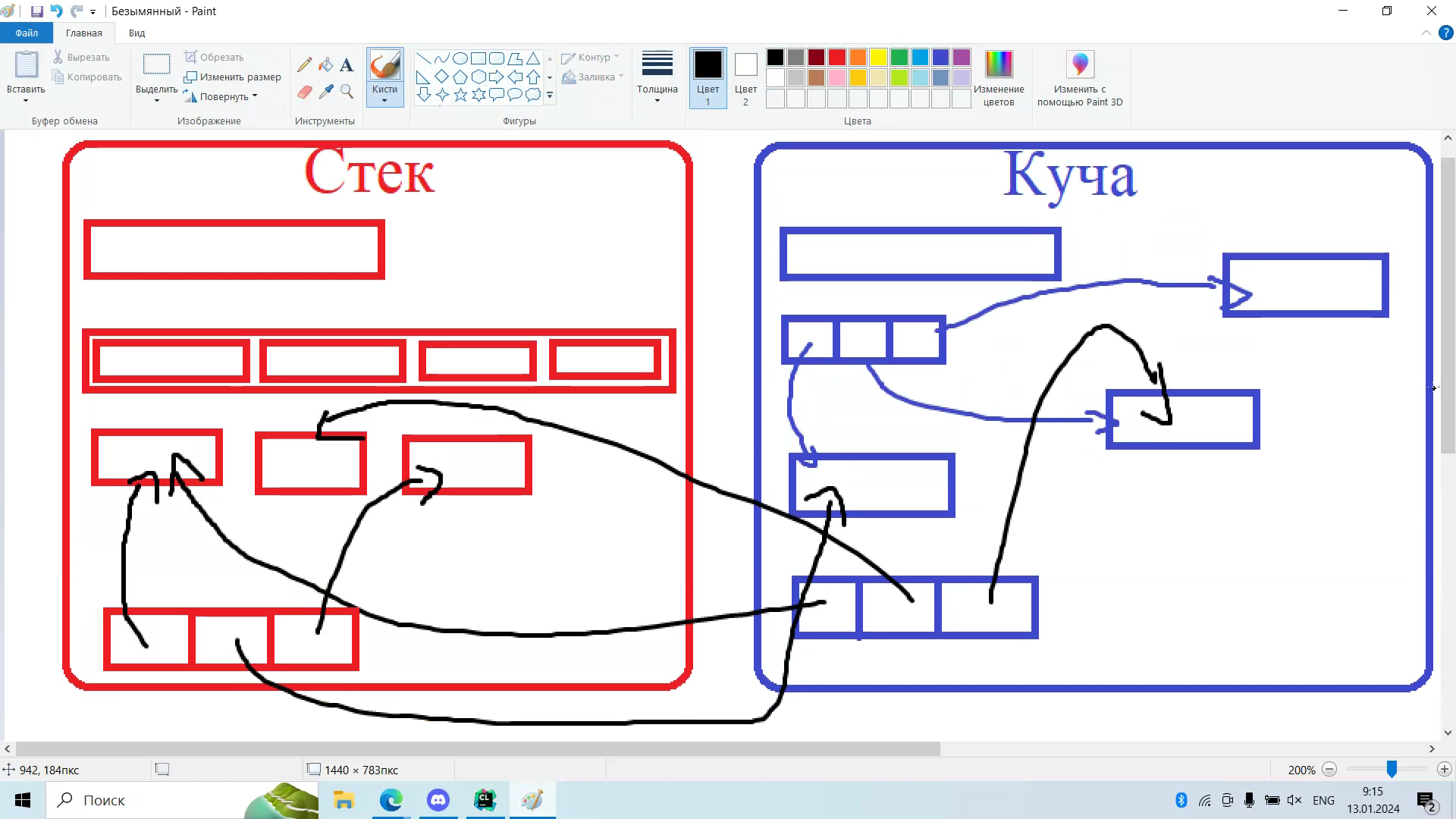


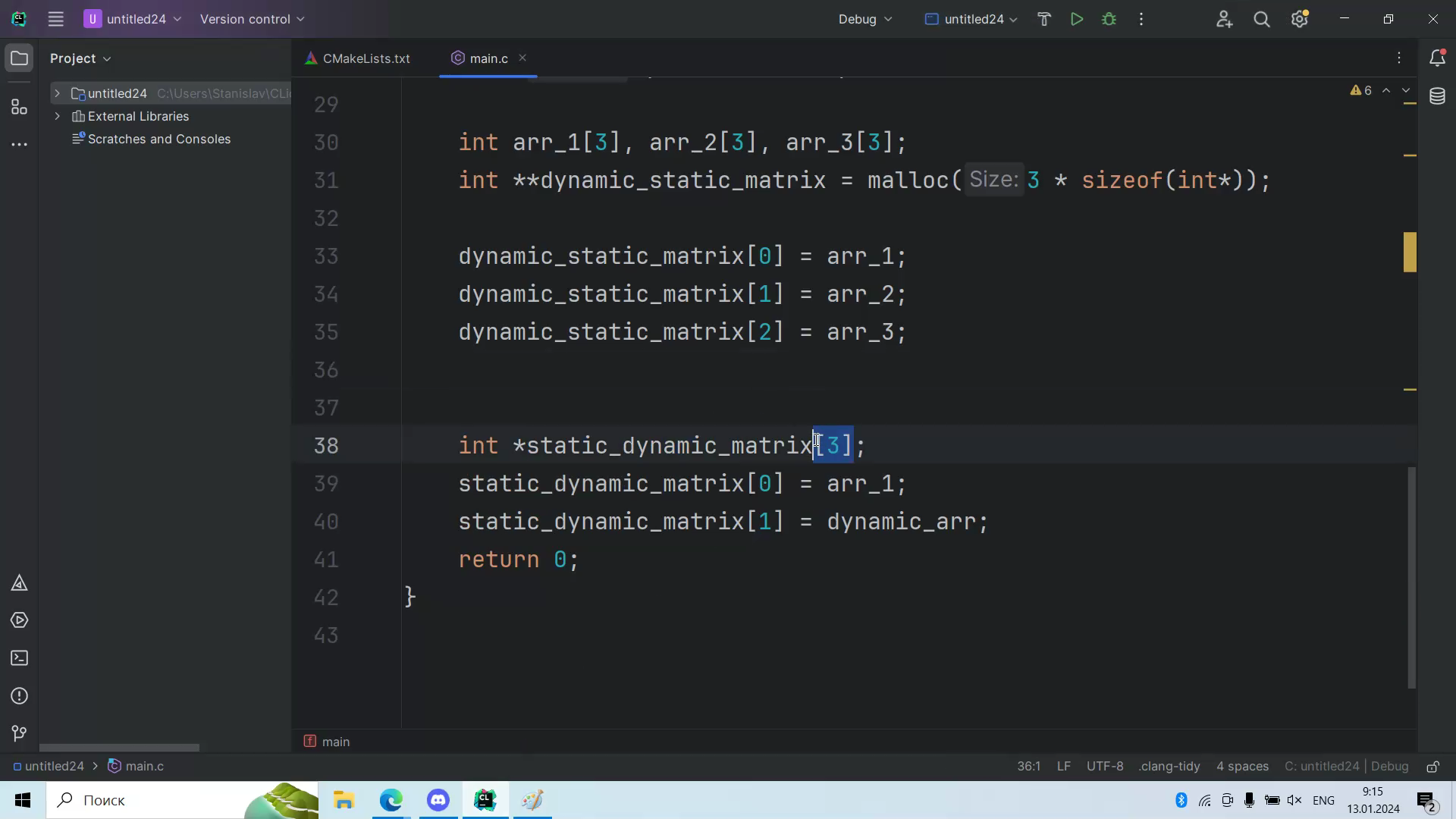
Освобождение динамической матрицы



В теории массив из указателей может ссылаться и на стек и на кучу







#include <stdio.h>

#include <stdlib.h>

int main() {

int static\_arr[3] = {0, 1, 2};

int \*dynamic\_arr = malloc(3 \* sizeof(int));

dynamic\_arr[0] = 0;

dynamic\_arr[1] = 1;

dynamic\_arr[2] = 2;

int static\_matrix[2][3] = {

{0, 1, 2},

{3, 4, 5}

};

int \*\*dynamic\_matrix = malloc(2 \* sizeof(int\*));

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

dynamic\_matrix[i] = malloc(3 \* sizeof(int));

}

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

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

dynamic\_matrix[i][j] = rand() % 21 - 10;

}

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

free(dynamic\_matrix[i]);

free(dynamic\_matrix);

int arr\_1[3], arr\_2[3], arr\_3[3];

int \*\*dynamic\_static\_matrix = malloc(3 \* sizeof(int\*));

dynamic\_static\_matrix[0] = arr\_1;

dynamic\_static\_matrix[1] = arr\_2;

dynamic\_static\_matrix[2] = arr\_3;

int \*static\_dynamic\_matrix[3];

static\_dynamic\_matrix[0] = arr\_1;

static\_dynamic\_matrix[1] = dynamic\_arr;

static\_dynamic\_matrix[1][2] = 15;

return 0;

}

2. Функции можно сначала объявлять а потом инициализировать

Сигнатура функции это объявление функции (некоторые определения говорят что название), типом входящего и выходящего значения.

#include <stdio.h>

#include <stdlib.h>

// Сложение двух целых чисел

int sum(int, int);

// Умножение двух целых чисел

int multiply(int, int);

int main() {

int b; // объявление

b = 3; // инициализация

int c = 5; // объявление с инициализацией

printf("%d\n", sum(2, 3));

return 0;

}

int sum(int a, int b){

return a + b;

}

int multiply(int a, int b){

return a \* b;

}

3. Статический массив нельзя передать из функции так как он удаляется, когда заканчивается блок функции, (после выполнения функции она передаёт указатель на массив, который умер)

#include <stdio.h>

#include <stdlib.h>

int matrix\_sum\_int(int \*\*matrix, int rows, int cols, int increment){

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

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

matrix[i][j] += increment;

}

int print\_matrix(int \*\*matrix, int rows, int cols){

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

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

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

printf("\n");

}

}

int rand\_matrix(int \*\*matrix, int rows, int cols){

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

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

matrix[i][j] = rand() % 21 - 10;

}

int\*\* make\_matrix(int rows, int cols){

int \*\*matrix = calloc(rows, sizeof(int\*));

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

matrix[i] = calloc(cols, sizeof(int));

return matrix;

}

// возвращается указатель на уничтоженный массив

int\*\* make\_static\_matrix(int rows, int cols){

int matrix[rows][cols];

return matrix;

}

int main() {

int rows = 3, cols = 2;

int \*\*matrix = make\_static\_matrix(rows, cols);

rand\_matrix(matrix, rows, cols);

print\_matrix(matrix, rows, cols);

matrix\_sum\_int(matrix, rows, cols, 2);

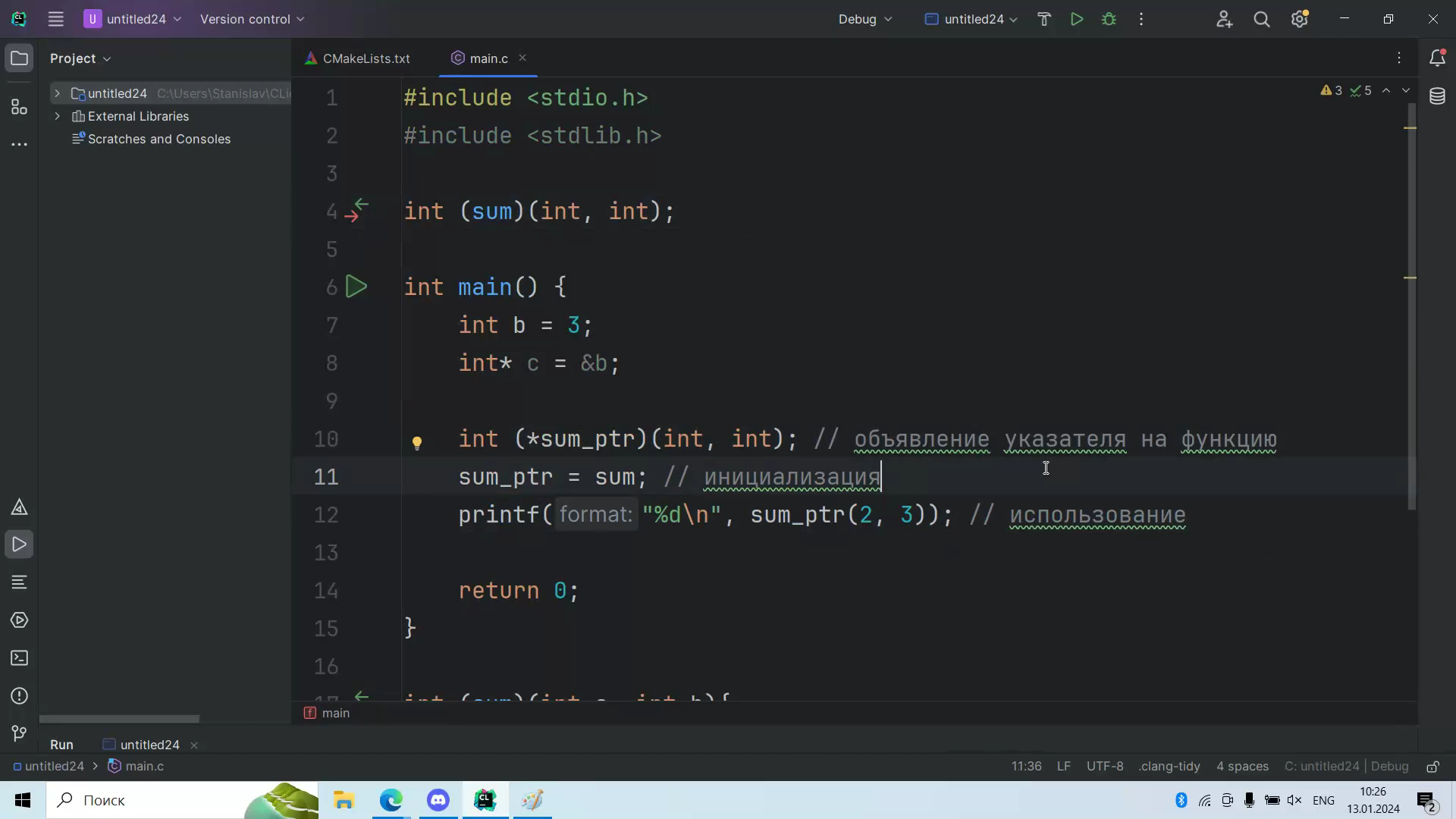
printf("result\n");

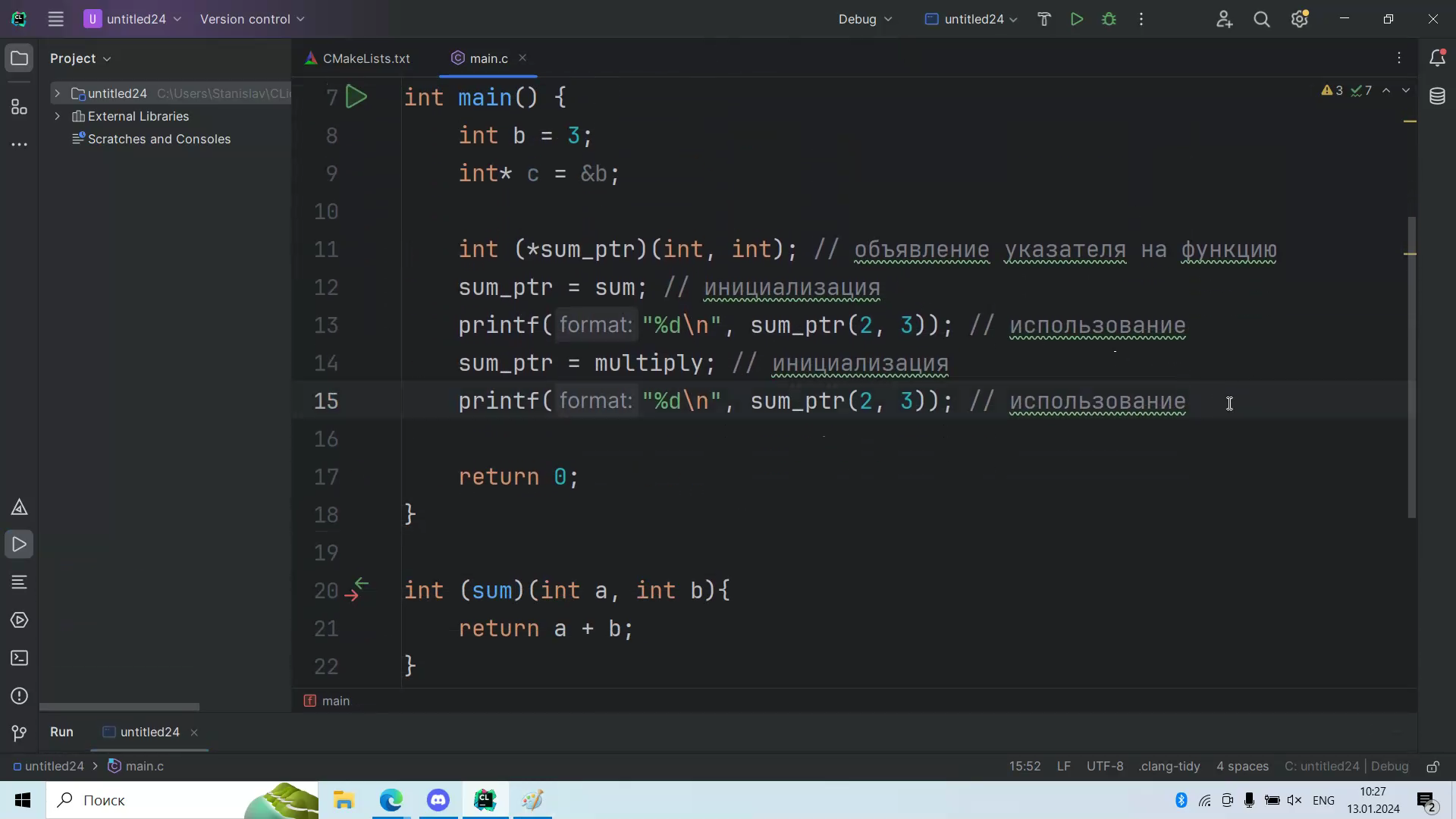
print\_matrix(matrix, rows, cols);

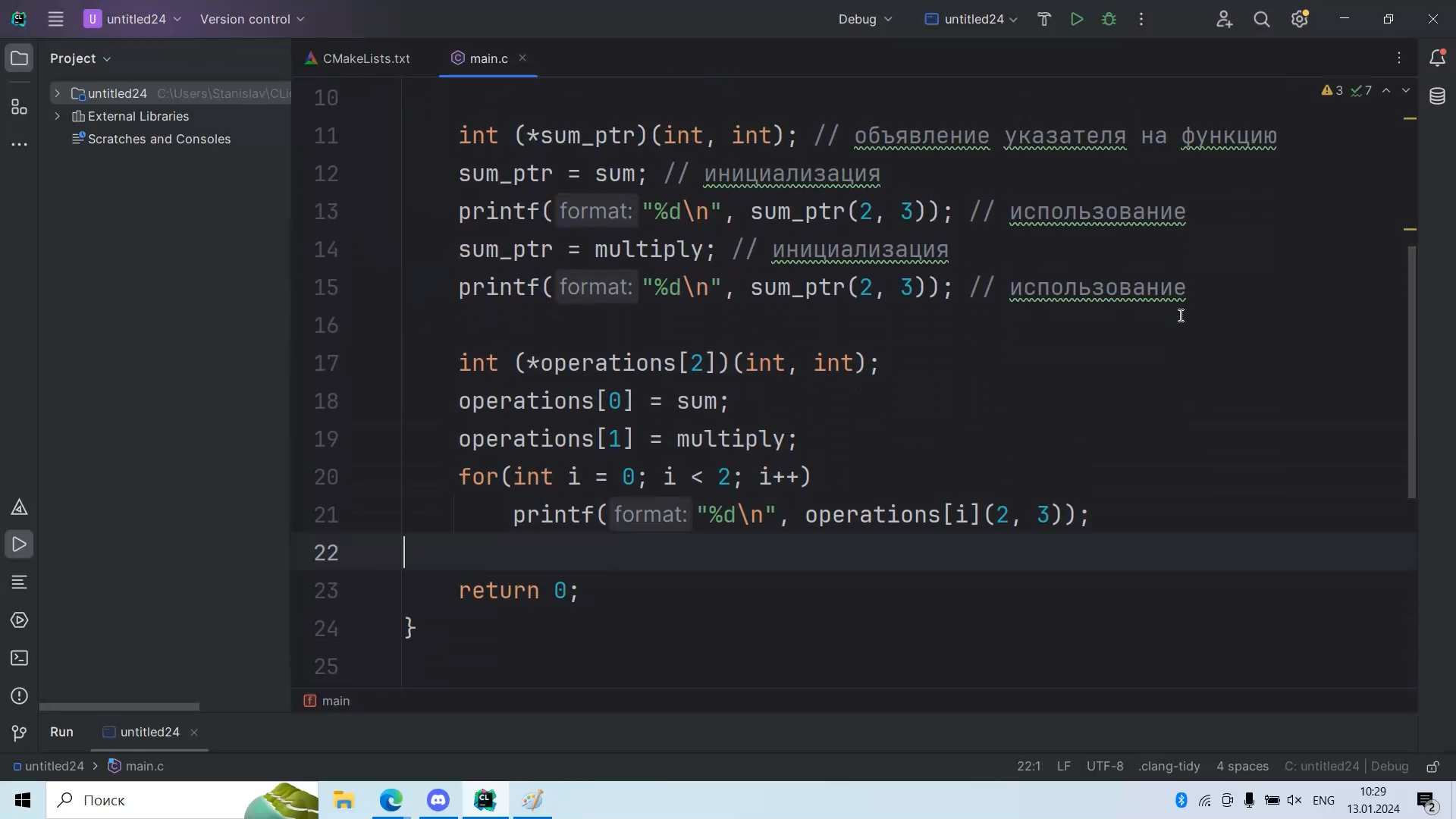
return 0;

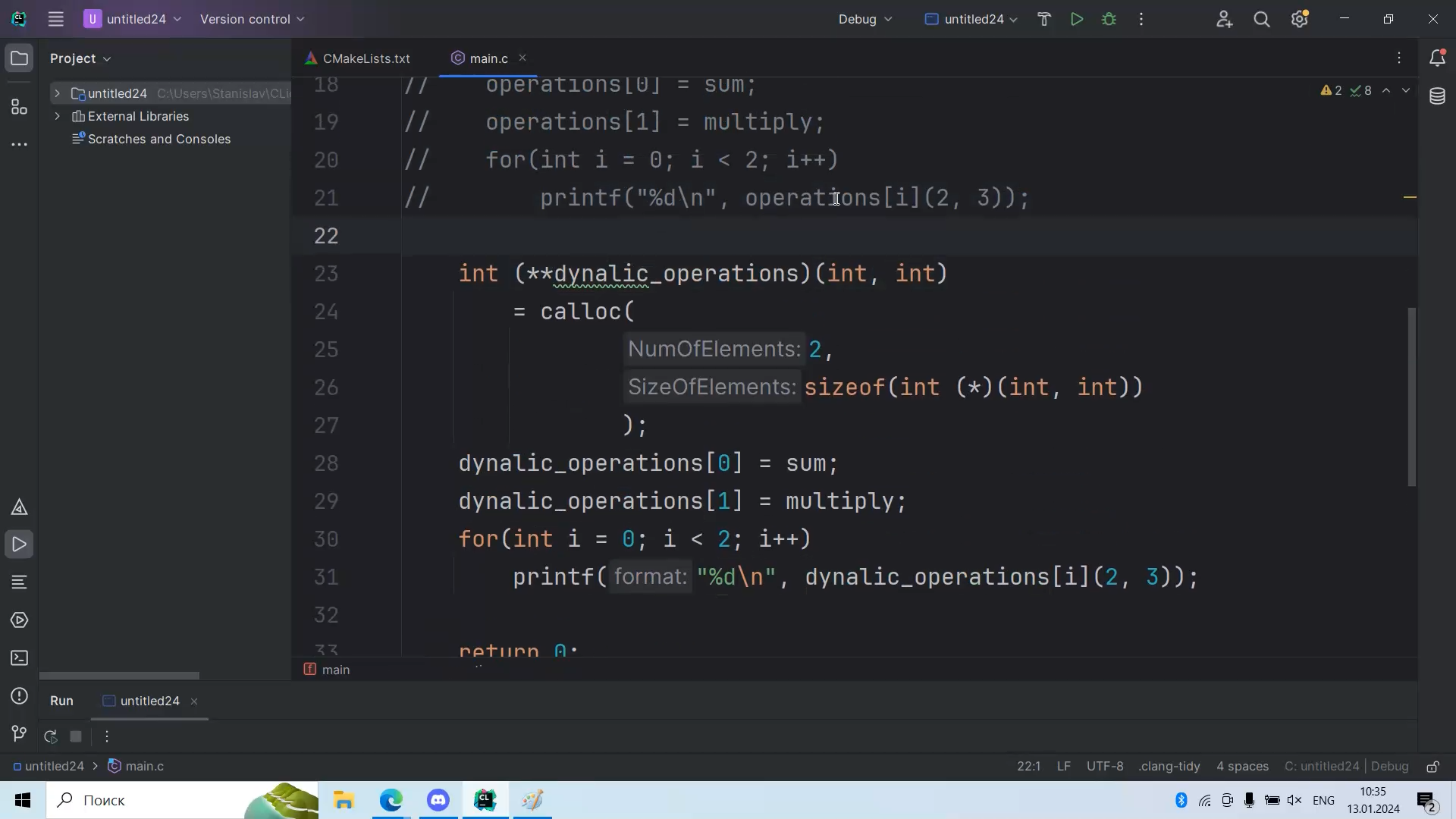
}

4. указатель на функцию









#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

bool order\_to\_min(int a, int b){

return a < b;

}

bool order\_to\_max(int a, int b){

return a > b;

}

bool order\_to\_two(int a, int b){

return a % 2;

}

bool order\_to\_no\_two(int a, int b){

return !(a % 2);

}

bool order\_to\_min\_last\_number(int a, int b){

return (a % 10) < (b % 10);

}

bool order\_to\_max\_last\_number(int a, int b){

return (a % 10) > (b % 10);

}

void custom\_sort(int \*arr, int n, bool (\*order)(int, int)){

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

int cur\_min = i, min = arr[i];

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

if(order(min, arr[j])) {

cur\_min = j;

min = arr[j];

}

}

int tmp = arr[i];

arr[i] = arr[cur\_min];

arr[cur\_min] = tmp;

}

}

int print\_arr(int\* arr, int n){

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

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

}

printf("\n");

}

int main() {

const int n = 11;

int arr[n];

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

arr[i] = rand() % 90 + 10;

}

print\_arr(arr, n);

custom\_sort(arr, n, order\_to\_min);

print\_arr(arr, n);

custom\_sort(arr, n, order\_to\_max);

print\_arr(arr, n);

custom\_sort(arr, n, order\_to\_two);

print\_arr(arr, n);

custom\_sort(arr, n, order\_to\_no\_two);

print\_arr(arr, n);

custom\_sort(arr, n, order\_to\_min\_last\_number);

print\_arr(arr, n);

custom\_sort(arr, n, order\_to\_max\_last\_number);

print\_arr(arr, n);

return 0;

}

Эту штуку используется для колбэк ( в функцию с попаданием в корабль отправляется функция отрисовки попадания )