#1

#include <stdio.h>

#define N 11

void print\_array(int a[]) {

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

printf("%3d", a[i]);

}

printf("\n");

}

int search\_element(int x, int a[]){

int L = 0; // 0

int R = N; // 10

int mid;

while (L < R) {

mid = (L + R)/2;

if (x == a[mid]) {

printf("position of '%d' is %d", x, mid);

break;

} else if(x < a[mid]) {

R = mid - 1;

} else if(x > a[mid]) {

L = mid + 1;

}

};

return 0;

}

int main(){

int arr[N];

int x = 11;

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

arr[i] = i+1;

}

print\_array(arr);

search\_element(x, arr);

return 0;

}

#2

|  |
| --- |
| #include <stdio.h> |
|  |  |
|  | #define N 11 |
|  |  |
|  | void print\_array(int a[]) { |
|  | for (int i = 0; i < N; ++i) { |
|  | printf("%3d", a[i]); |
|  | } |
|  | printf("\n"); |
|  | } |
|  |  |
|  | int search\_element(int x, int a[]){ |
|  | int L = 0; // 0 |
|  | int R = N; // 10 |
|  | int mid; |
|  |  |
|  | while (L < R) { |
|  | mid = (L + R)/2; |
|  |  |
|  | if (x == a[mid]) { |
|  | printf("position of '%d' is %d", x, mid); |
|  | break; |
|  | } else if(x < a[mid]) { |
|  | R = mid - 1; |
|  | } else if(x > a[mid]) { |
|  | L = mid + 1; |
|  | } |
|  | }; |
|  |  |
|  | return 0; |
|  | } |
|  |  |
|  |  |
|  | int main(){ |
|  |  |
|  | int arr[N] = {0, 1, 1, 1, 5, 5, 7, 8, 9, 10,11}; |
|  | int x = 11; |
|  |  |
|  | print\_array(arr); |
|  | search\_element(x, arr); |
|  |  |
|  | return 0; |
|  | } |

#3

#include <stdio.h>

#include <stdbool.h>

const int n = 10;

const int m = 8;

int A[8][10] = { { -2, 0, 0, 0, 0, 0, 0, 0, 0, 0 },

{ 2, 0, 0, 0, 0, 0, 0, 0, 0, 0 },

{ 4, 0, 0, 0, 0, 0, 0, 0, 0, 0 },

{ 5, 0, 0, 0, 0, 0, 0, 0, 0, 0 },

{ 7, 0, 0, 0, 0, 0, 0, 0, 0, 0 },

{ 9, 0, 0, 0, 0, 0, 0, 0, 0, 0 },

{ 11, 0, 0, 0, 0, 0, 0, 0, 0, 0 },

{ 13, 14, 16, 17, 19, 20, 22, 25, 27, 30 }};

void BiSearch(int answer)

{

int count = 0;

bool Flag = false;

while (count < 30)

{

if (answer == A[count][count])

{

printf("This number located at index:( %d , %d )\n", count, count);

count += 1;

Flag = true;

}

else if (answer < A[count][count])

count += 1;

else if (answer > A[count][count])

count += 1;

}

if (Flag == false) {

printf("This number doesn't exist\n");

}

}

void print(int A[8][10], int N, int M)

{

for (int R = 0; R < N; R++) {

for (int C = 0; C < M; C++)

printf("%d", A[R][C]);

printf("\n");

}

}

int main()

{

print(A, m, n);

int answer;

printf("\nPlease input number\n\n");

scanf("%d", &answer);

if (answer < 1 || answer > 5)

{

printf("Sorry but your number doesnt exist in array\n");

return 0;

}

else

{

BiSearch(answer);

return 1;

}

}

#4

#include <stdio.h>

#define M 10

#define N 9

int matrix[M][N] = {

{ -2, -2, -2, -2, -2, -2, -2, -2 },

{ -2, -2, -2, 9, 12, 16, 19, 22 },

{ -2, -2, 6, 8, 12, 16, 19, 22 },

{ -2, -2, 5, 8, 12, 15, 18, 21 },

{ -1, 2, 5, 8, 11, 14, 18, 21 },

{ -1, 2, 5, 8, 11, 14, 18, 21 },

{ -1, 2, 5, 7, 11, 14, 17, 20 },

{ 1, 1, 4, 7, 10, 14, 17, 20 },

{ 1, 1, 4, 7, 10, 13, 17, 20 },

{ 1, 1, 4, 7, 10, 13, 17, 20 } };

int main(void) {

int col = 1, key;

printf("Input your key: ");

scanf("%d", &key);

if (key < -2 || key > 22) {

printf("Your key doesn't exist in matrix");

return 0;

}

while (col < 9) {

int low = 1, high = 10, mid = (low + high) / 2;

while (low <= high) {

if (matrix[mid][col] == key) {

printf("Found key at coord [%d, %d]\n", col, mid);

break;

}

else if (matrix[mid][col] > key) {

low = mid + 1;

}

else if (matrix[mid][col] < key) {

high = mid - 1;

}

mid = (low + high) / 2;

}

col++;

}

return 0;

}

№5

#include <stdio.h>

#define M 10

#define N 9

int matrix[M][N] = {

{ -2, -2, -2, -2, -2, -2, -2, -2 },

{ -2, -2, -2, -2, -2, -2, -2, -2 },

{ -2, -2, -2, -2, -2, -2, -2, -2 },

{ -2, -2, -2, -2, -2, -2, -2, -2 },

{ -1, -1, -1, -1, -1, -1, -1, -1 },

{ -1, -1, -1, -1, -1, -1, -1, -1 },

{ -1, -1, -1, -1, -1, -1, -1, -1 },

{ 1, 1, 1, 1, 1, 1, 1, 1 },

{ 1, 1, 1, 1, 1, 1, 1, 1 },

{ 1, 1, 4, 7, 10, 13, 17, 20 } };

int main(void) {

int col = 1, key;

printf("Input your key: ");

scanf("%d", &key);

if (key < -2 || key > 22) {

printf("Your key doesn't exist in matrix");

return 0;

}

while (col < 9) {

int low = 1, high = 10, mid = (low + high) / 2;

while (low <= high) {

if (matrix[mid][col] == key) {

printf("Found key at coord [%d, %d]\n", col, mid);

break;

}

else if (matrix[mid][col] > key) {

low = mid + 1;

}

else if (matrix[mid][col] < key) {

high = mid - 1;

}

mid = (low + high) / 2;

}

col++;

}

return 0;

}

№6

#include <stdio.h>

#define M 10

#define N 8

int matrix[10][8] = {

{ -2, 1, 4, 7, 10, 13, 17, 20 },

{ -2, 1, 4, 7, 10, 13, 17, 20 },

{ -2, 1, 4, 7, 10, 14, 17, 20 },

{ -1, 2, 5, 7, 11, 14, 17, 20 },

{ -1, 2, 5, 8, 11, 14, 18, 21 },

{ -1, 2, 5, 8, 11, 14, 18, 21 },

{ -1, 2, 5, 8, 11, 15, 18, 21 },

{ 0, 2, 5, 8, 11, 15, 18, 21 },

{ 0, 2, 5, 8, 11, 15, 18, 21 },

{ 0, 3, 6, 9, 12, 16, 19, 22 }};

int matrix\_test[5][4] = {

{ -2, 1, 4, 7, },

{ -2, 1, 4, 7, },

{ -2, 1, 4, 7, },

{ -1, 2, 5, 7, },

{ -1, 2, 5, 8 }};

int BinarSearch(int arr[M][N], int rows, int cols, int key)

{

int first,last,mid,temp;

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

first = 0;

last = cols - 1;

printf("last %d \n", last);

while (first < last) {

mid = (first + last) / 2;

printf("mid %d \n", mid);

if (key <= arr[i][mid]) {

last = mid;

printf("Last: %d \n", last);

}

else {

first = mid + 1;

printf("First: %d \n", first);

}

}

if ( arr[i][last] == key) {

printf("Found value, coordinates: [%d, %d] \n", i, last);

break;

goto stop;

} else {

printf("Value not found \n");

goto stop;

}

stop: printf("Jumped to stop \n" );

}

}

int main(void) {

// int col = 0, key;

// printf("Input your key: ");

// scanf("%d", &key);

// if (key < -2 || key > 22) {

// printf("Your key doesn't exist in matrix");

// return 0;

// }

BinarSearch(matrix, 10, 8, -1);

// while (col < 8) {

// int low = 0, high = 9, mid = (low + high) / 2;

// while (low <= high) {

// if (matrix[mid][col] == key) {

// printf("Found key at coord [%d, %d]\n", col, mid);

// break;

// }

// else if (matrix[mid][col] > key) {

// low = mid + 1;

// }

// else if (matrix[mid][col] < key) {

// high = mid - 1;

// }

// mid = (low + high) / 2;

// }

// col++;

//}

return 0;

}