1. Hello World: Print "Hello, World!" to the console.

#include<stdio.h>

int main()

{

FILE \*sp;

sp=fopen("output.txt","w");

if(sp==NULL)

{

printf("not found\n");

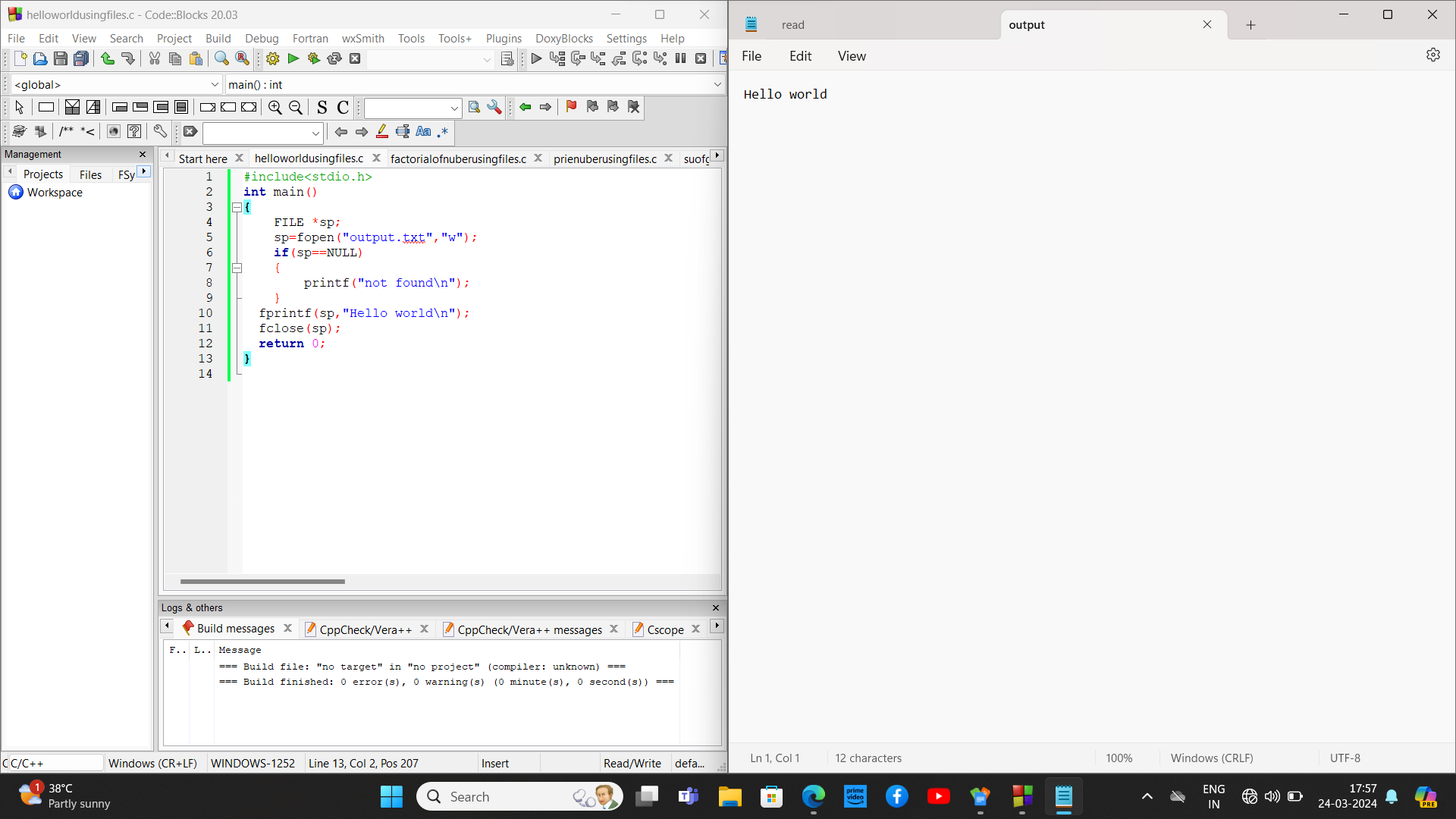
}

fprintf(sp,"Hello world\n");

fclose(sp);

return 0;

}



1. Factorial: Calculate the factorial of a given number.

#include <stdio.h>

#include<stdlib.h>

#include <stdio.h>

#include<stdlib.h>

int main()

{

FILE \*fp,\*fp2;

fp2=fopen("read.txt","r");

fp=fopen("output.txt","w");

if(fp==NULL || fp2==NULL)

{

printf("notfound\n");

}

int a,f=1;

fscanf(fp2,"%d",&a);

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

{

f=f\*i;

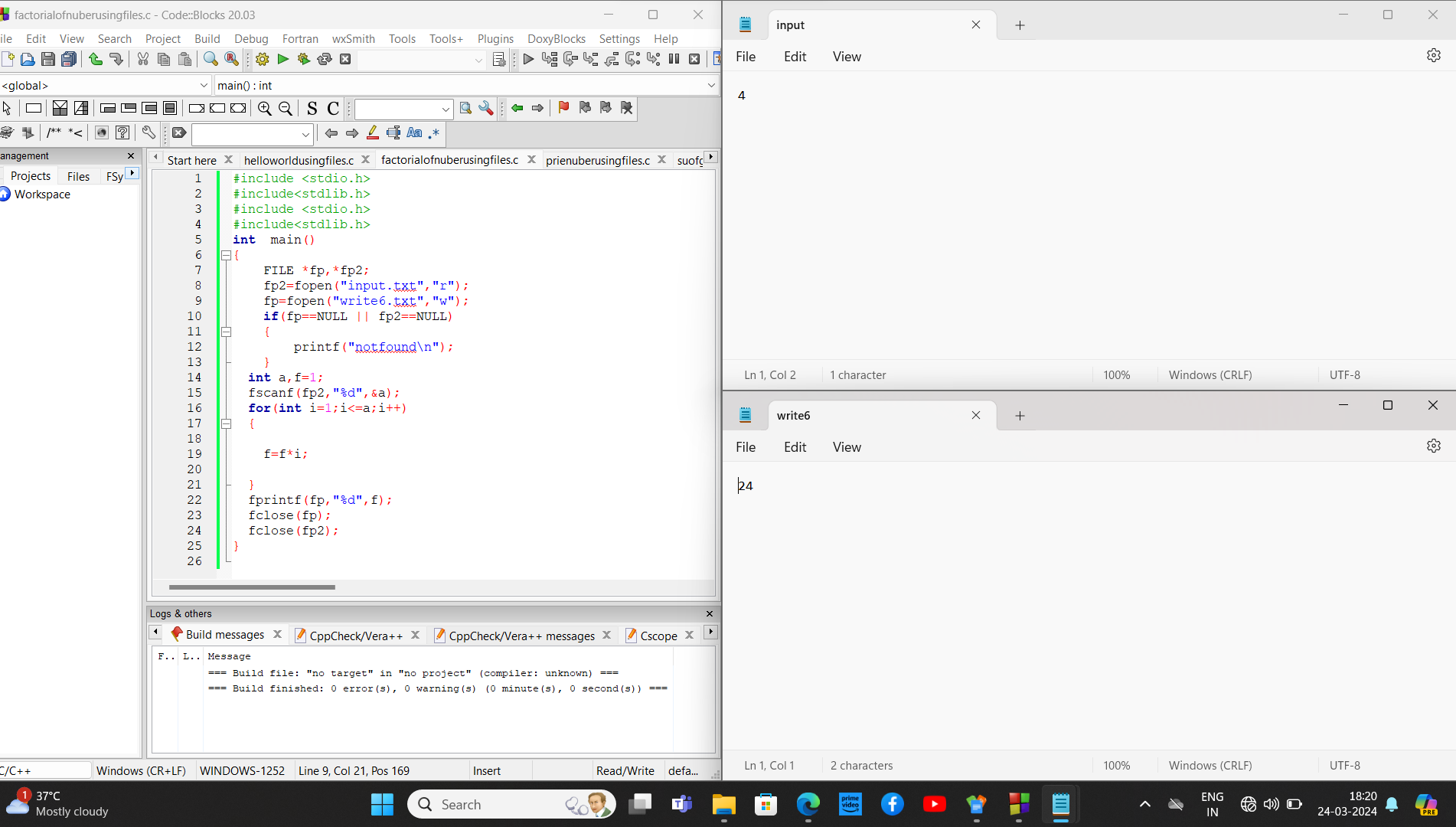
}

fprintf(fp,"%d",f);

fclose(fp);

fclose(fp2);

}



1. Prime Numbers: Determine whether a given number is prime.

#include<stdio.h>

#include<stdlib.h>

int main()

{

FILE \*fp, \*fp2;

int flag=0;

fp = fopen("read.txt", "r");

fp2 = fopen("write.txt", "w");

int n;

if (fp == NULL || fp2 == NULL) {

printf("File not found\n");

return 1;

}

fscanf(fp,"%d",&n);

printf("%d",n);

if(n==0 || n==1)

{

flag=1;

}

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

{

if(n%i==0)

{

flag=1;

break;

}

}

if(flag==0)

{

fprintf(fp2," %d a prime no",n);

}

else if(flag==1)

{

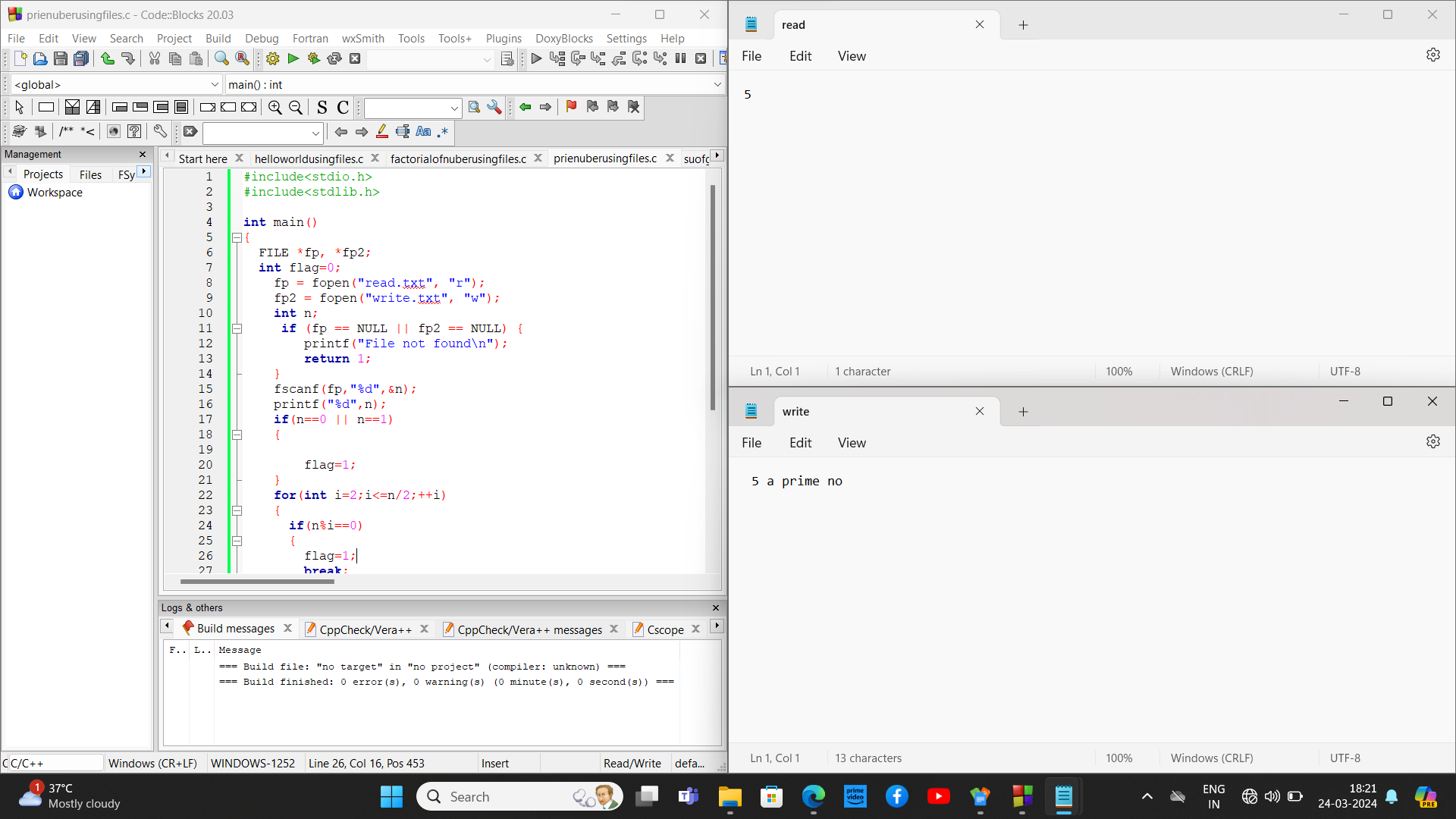
fprintf(fp2," %d not a prime no",n);

}

fclose(fp);

fclose(fp2);

}



1. Fibonacci Series: Generate the Fibonacci series up to a certain limit.

//fibonacii

#include <stdio.h>

#include<stdlib.h>

int main() {

FILE \*fp,\*fp2;

fp=fopen("output.txt","w");

fp2=fopen("read.txt","r");

if(fp==NULL ||fp2==NULL )

{

printf("not found\n");

}

int n, s = 0;

fscanf(fp2,"%d", &n);

int prev = 0;

int curr = 1;

fprintf(fp,"%d\n", prev);

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

s = prev + curr;

fprintf(fp,"%d\n", s);

prev = curr;

curr = s;

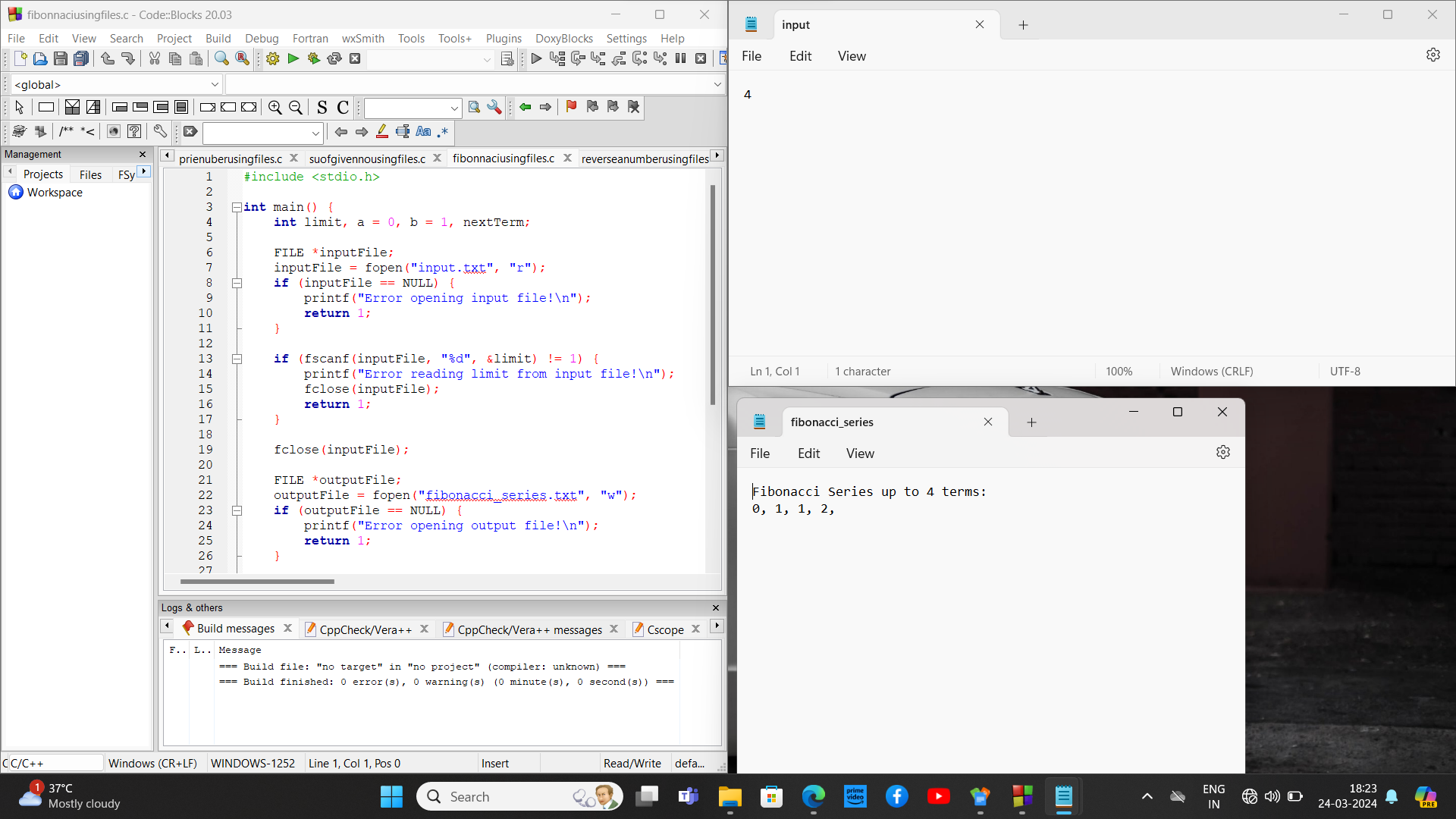
}

fclose(fp2);

fclose(fp);

return 0;

}



5)Sum of Digits: Calculate the sum of digits of a given number.

#include<stdio.h>

#include<stdlib.h>

int main() {

int n;

int s = 0;

FILE \*fp, \*fp2;

fp = fopen("read.txt", "r");

fp2 = fopen("output.txt", "w");

if (fp == NULL || fp2 == NULL) {

printf("File not found\n");

return 1; // Exit the program with error code 1

}

fscanf(fp, "%d", &n);

if (n < 0) {

fprintf(fp2, "sum=%d\n", n);

} else {

while (n != 0) {

s = s + n % 10;

n = n / 10;

}

fprintf(fp2, "sum = %d\n", s);

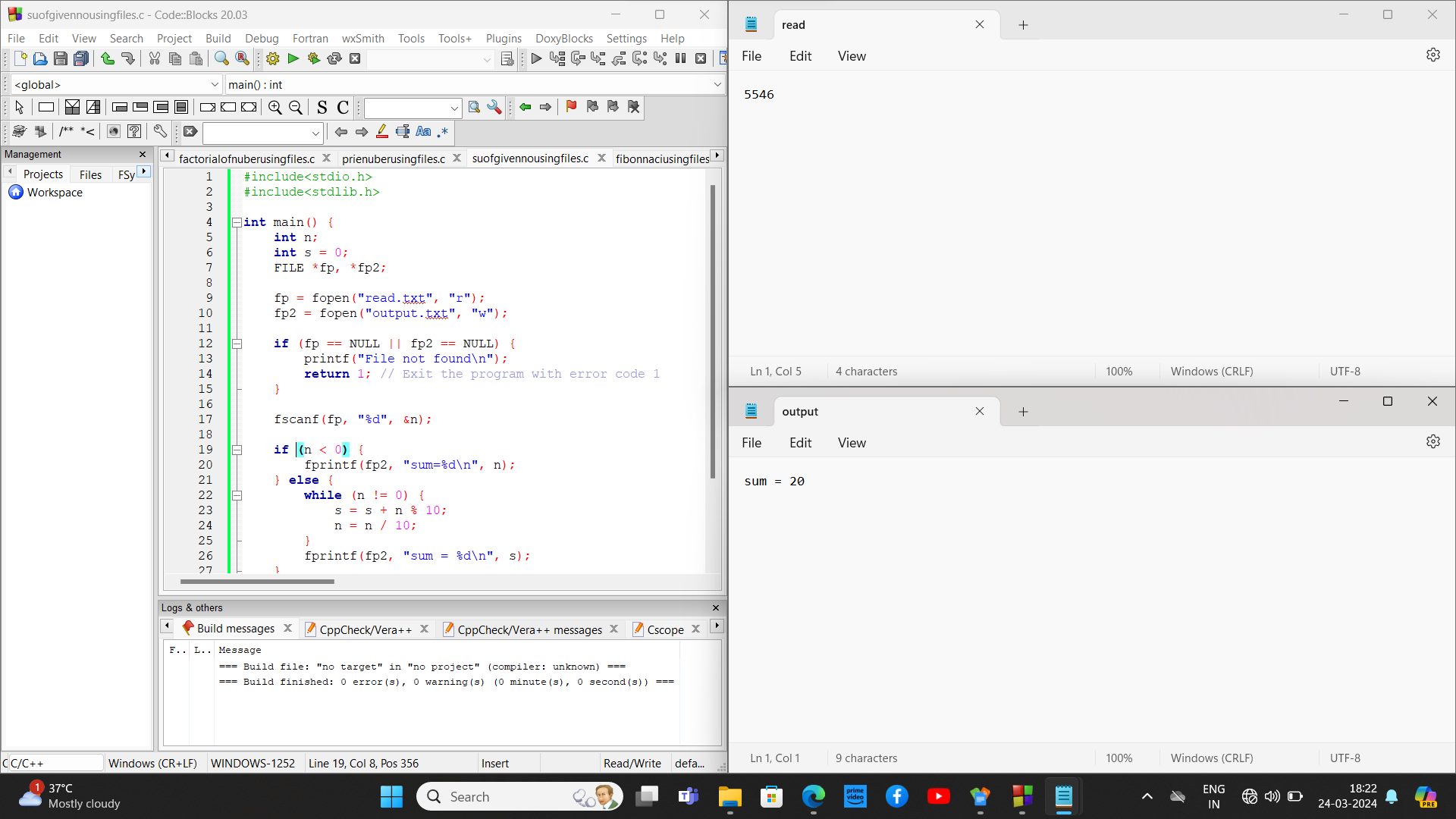
}

fclose(fp);

fclose(fp2);

return 0;

}



6)Reverse a Number: Reverse the digits of a given number.

//REVERSE NO

#include <stdio.h>

#include<stdlib.h>

int main() {

FILE \*sp, \*sp2;

sp = fopen("read.txt", "r");

sp2 = fopen("output.txt", "w");

if (sp == NULL || sp2 == NULL) {

printf("File not found\n");

return 1;

}

int n, s = 0;

fscanf(sp, "%d", &n);

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

while (n != 0) {

int r = n % 10;

s = s \* 10 + r;

n /= 10;

}

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

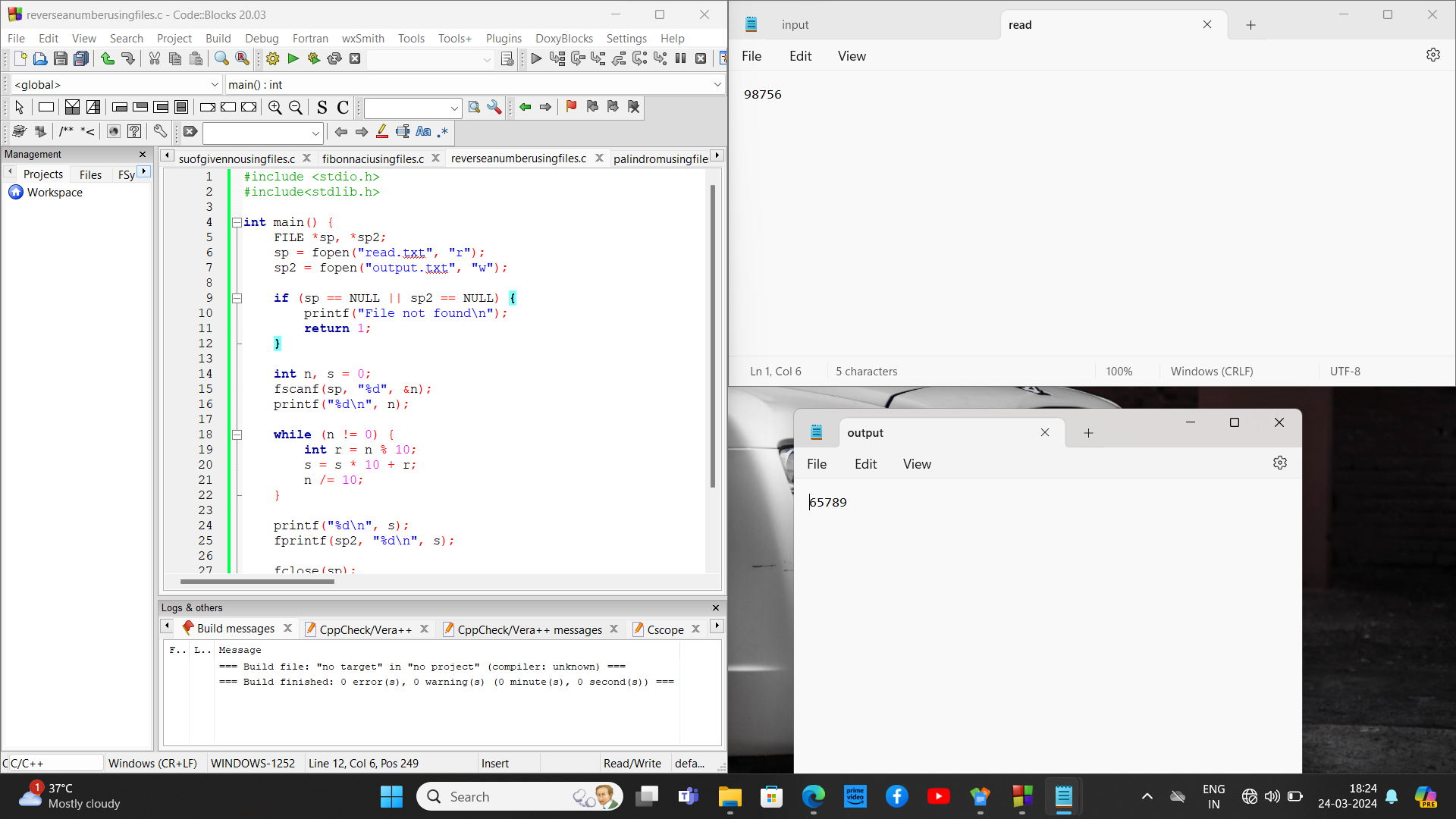
fprintf(sp2, "%d\n", s);

fclose(sp);

fclose(sp2);

return 0;

}



1. Palindrome Check: Check if a given number or string is a palindrome.

#include <stdio.h>

#include<stdlib.h>

int main() {

FILE \*sp,\*sp2;

int n, r, s = 0;

sp2=fopen("read.txt","r");

fscanf(sp2,"%d", &n);

int o = n;

sp=fopen("output.txt","w");

if(sp==NULL || sp2==NULL) {

printf("Error opening file\n");

return 1;

}

while (n != 0) {

r = n % 10;

s = s \* 10 + r;

n = n / 10;

}

if (o == s) {

printf("Palindrome\n");

fprintf(sp, "%d Palindrome\n",o);

} else {

printf("Not a Palindrome\n");

fprintf(sp, " %d Not a Palindrome\n",o);

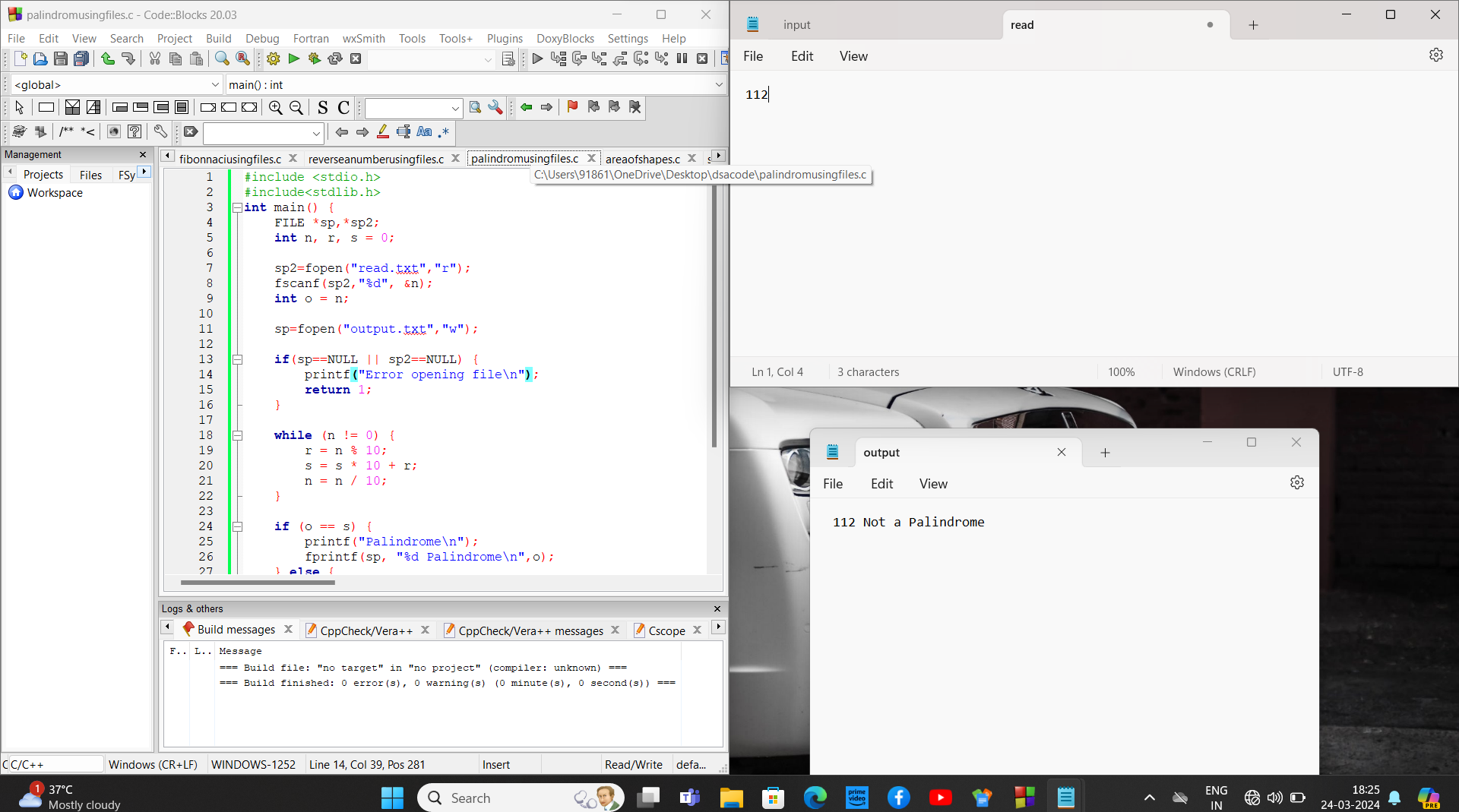
}

fclose(sp);

fclose(sp2);

return 0;

}



1. Area of Shapes: Calculate the area of shapes like rectangle, triangle, and circle

#include<stdio.h>

#include<math.h>

void areasqr(FILE \*fp, FILE \*fp2) {

int s,a=0;

fscanf(fp,"%d",&a);

s = a \* a;

fprintf(fp2,"%d\n",s);

}

void arearec(FILE \*fp, FILE \*fp2) {

int s1,s2,a=0;

fscanf(fp,"%d %d",&s1,&s2);

a = s1 \* s2;

fprintf(fp2,"%d\n",a);

}

void areatri(FILE \*fp, FILE \*fp2) {

int b,h,a=0;

fscanf(fp,"%d %d",&b,&h);

a = 0.5 \* b \* h;

fprintf(fp2,"%d\n",a);

}

void areac(FILE \*fp, FILE \*fp2) {

int r;

fscanf(fp,"%d",&r);

double a = 3.14 \* r \* r;

fprintf(fp2,"%lf\n",a);

}

int main() {

FILE \*fp,\*fp2;

fp = fopen("read.txt","r");

fp2 = fopen("output.txt","a");

if(fp == NULL || fp2 == NULL) {

printf("File not found\n");

return 1;

}

int n;

fscanf(fp,"%d",&n);

char shape;

fscanf(fp," %c",&shape);

if(shape=='s') {

areasqr(fp, fp2);

} else if(shape == 'r') {

arearec(fp, fp2);

} else if(shape == 'c') {

areac(fp, fp2);

} else if(shape == 't') {

areatri(fp, fp2);

} else {

printf("Invalid shape\n");

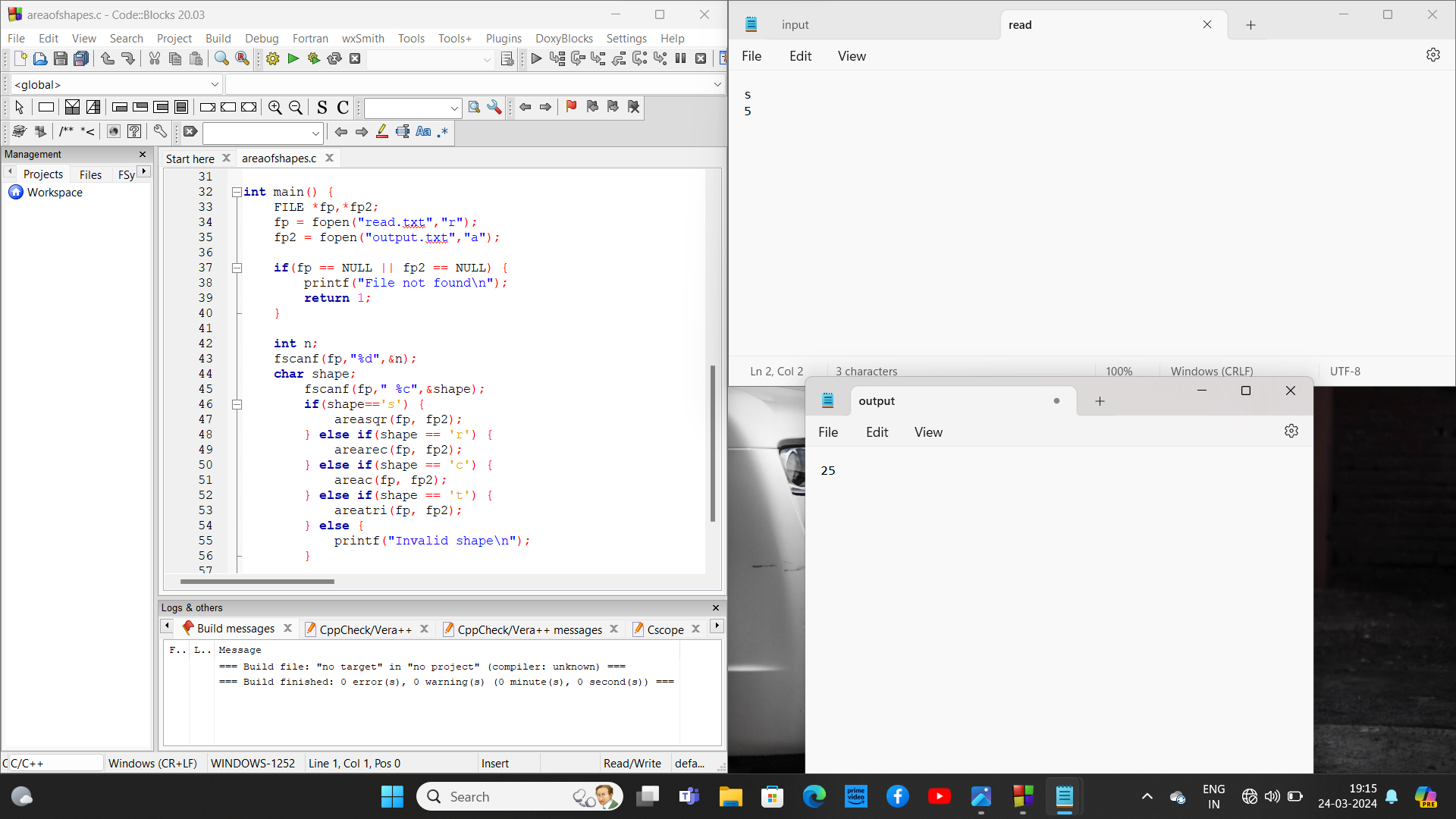
}

fclose(fp);

fclose(fp2);

return 0;

}



9. Simple Calculator: Implement a basic calculator with arithmetic operations.

#include<stdio.h>

#include<stdlib.h>

int main()

{

FILE \*fp,\*fp2;

fp=fopen("read.txt","r");

fp2=fopen("output.txt","a");

if (fp==NULL||fp2==NULL)

{

printf("not found\n");

}

int sw;

fscanf(fp,"%d",&sw);

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

int a,b,r;

fscanf(fp,"%d %d",&a,&b);

printf("%d %d\n",a,b);

switch(sw)

{

case 1:r=a+b;

fprintf(fp2,"%d\n",r);

break;

case 2:r=a-b;

fprintf(fp2,"%d\n",r);

break;

case 3:r=a\*b;

fprintf(fp2,"%d\n",r);

break;

case 4:r=a/b;

fprintf(fp2,"%d\n",r);

break;

case 5:r=a%b;

fprintf(fp2,"%d\n",r);

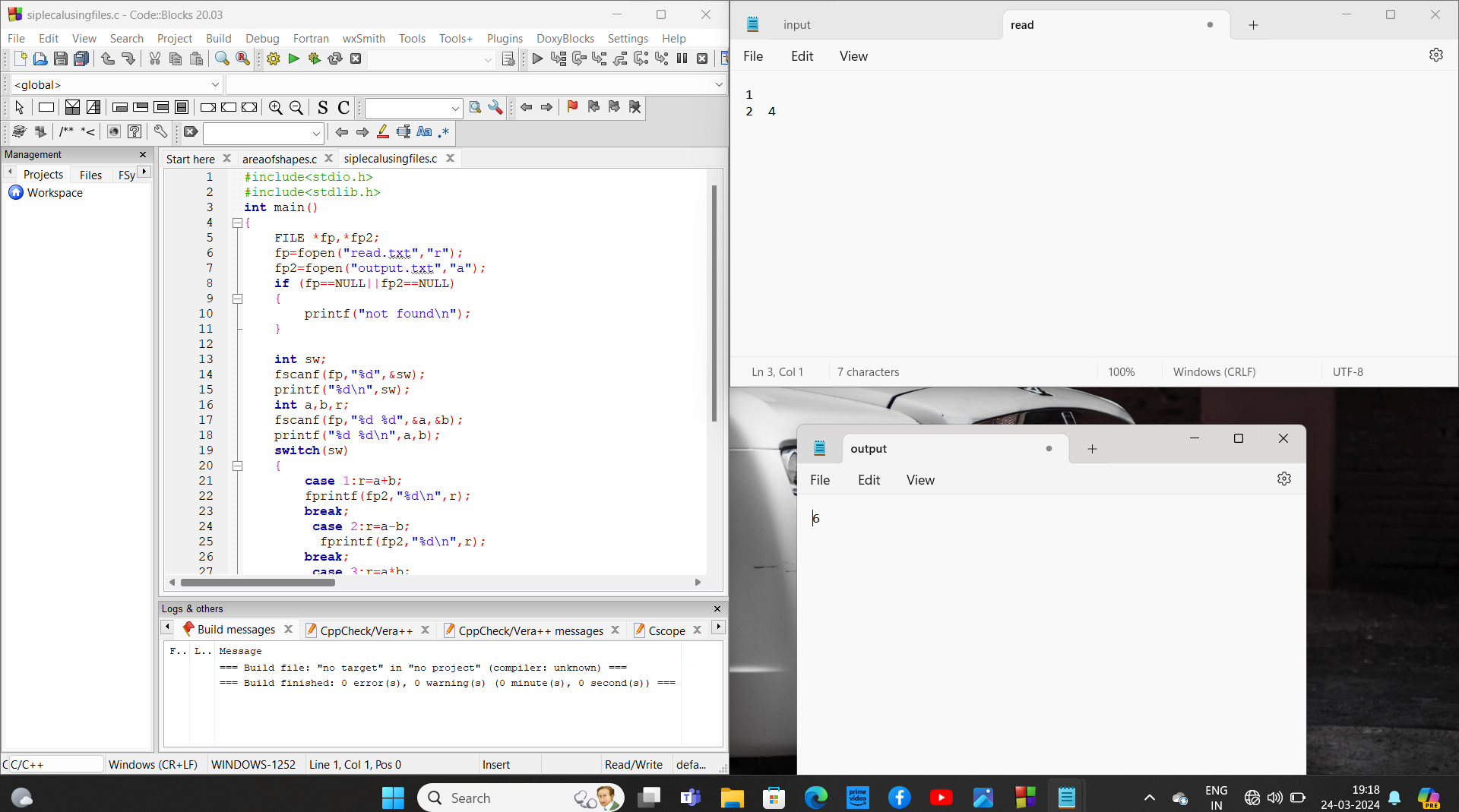
break;

}

fclose(fp);

fclose(fp2);

}



10.Array Operations: Perform operations like finding the largest/smallest element, sum, and average of an array

#include <stdio.h>

int main() {

int a[100];

FILE \*fp, \*fp2;

fp = fopen("read.txt", "r");

fp2 = fopen("output.txt", "a");

if (fp == NULL || fp2 == NULL) {

printf("File not found\n");

return 1;

}

int n, sw;

int s = 0;

int min, max;

fscanf(fp, "%d", &n);

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

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

fscanf(fp, "%d", &a[i]);

fprintf(fp2, "%d\t", a[i]);

}

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

fscanf(fp, "%d", &sw);

{ switch (sw) {

case 1:

max = a[0];

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

if (a[i] > max) {

max = a[i];

}

}

fprintf(fp2, "\nMax: %d\n", max);

break;

case 2:

min = a[0];

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

if (a[i] < min) {

min = a[i];

}

}

fprintf(fp2, "\nMin: %d\n", min);

break;

case 3:

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

s += a[i];

}

fprintf(fp2, "\nSum: %d\n", s);

double avg = (double)s / n;

fprintf(fp2, "Average: %lf\n", avg);

break;

default:

printf("Invalid switch case\n");

}

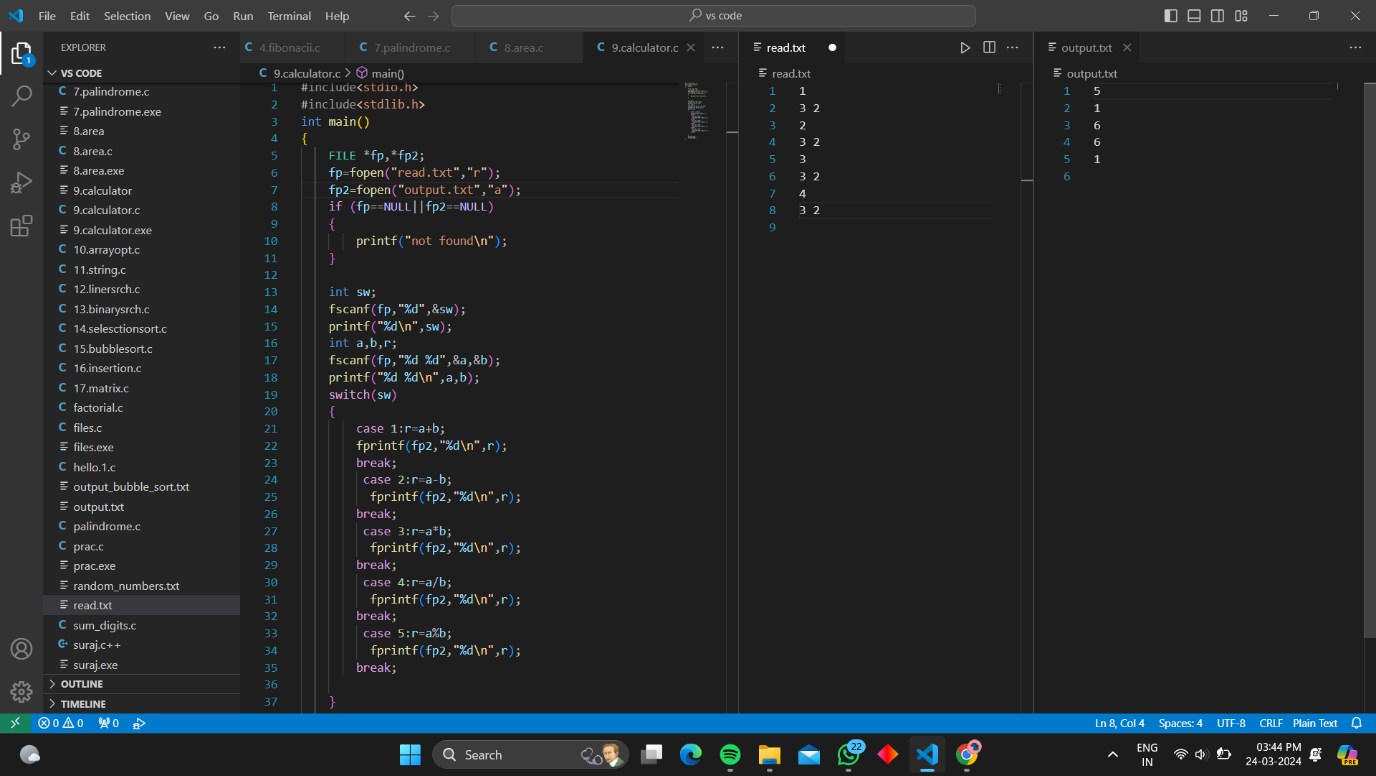
}

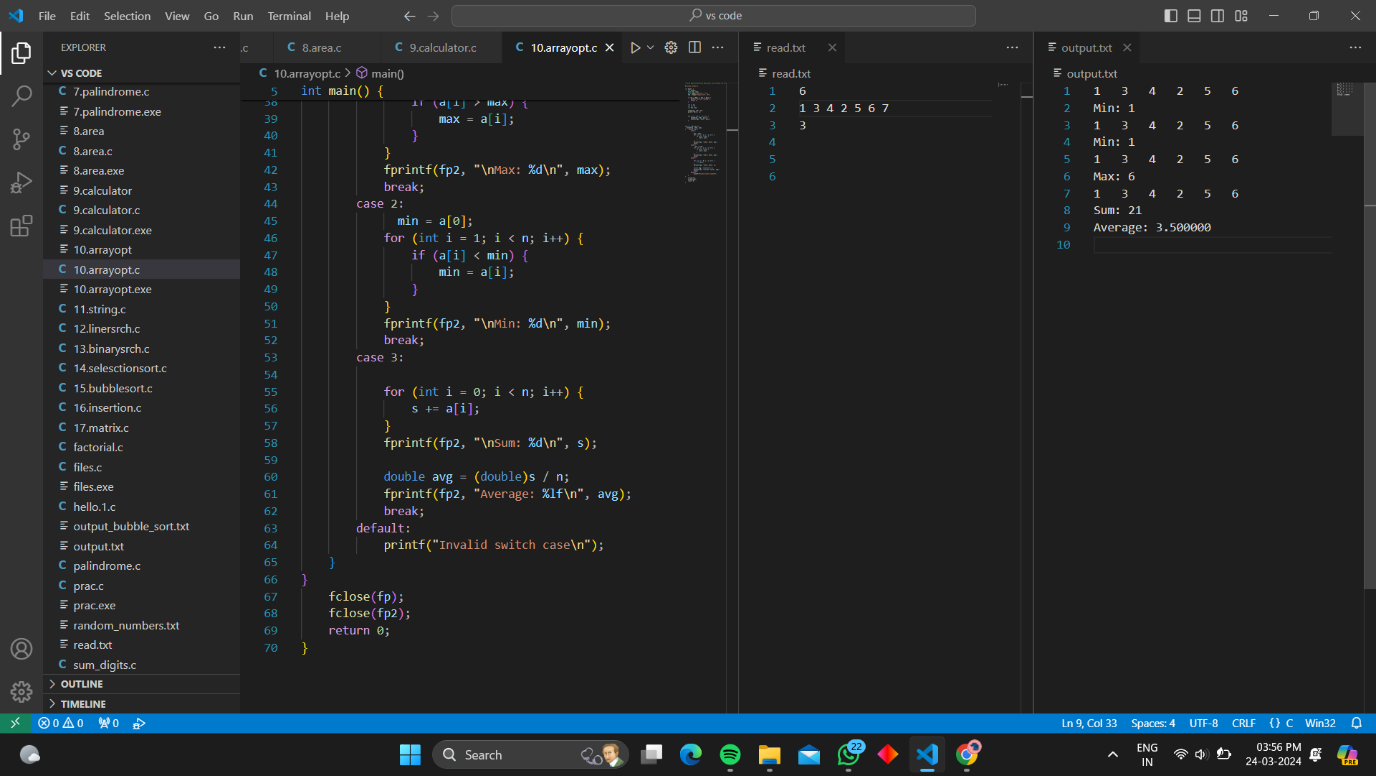
fclose(fp);

fclose(fp2);

return 0;

}





11. String Operations: Manipulate strings such as concatenation, copying, and comparison.

#include <stdio.h>

#include <string.h>

int main() {

char s[100];

char k[100];

FILE \*fp, \*fp2;

fp = fopen("read.txt", "r");

fp2 = fopen("output.txt", "a");

if (fp == NULL || fp2 == NULL) {

printf("File not found\n");

return 1;

}

int sw;

fscanf(fp, "%d", &sw);

switch (sw) {

case 1:

fscanf(fp, "%s", s);

fscanf(fp, "%s", k);

strcat(s, k);

fprintf(fp2, "%s", s);

break;

case 2:

fscanf(fp, "%s", s);

fscanf(fp, "%s", k);

fprintf(fp2, "\nb/f Copyingf string: %s\n", s);

strcpy(s, k);

fprintf(fp2, "Copied string: %s", s);

break;

case 3:

fscanf(fp, "%s", s);

fscanf(fp, "%s", k);

int res = strcmp(s, k);

if (res == 0)

fprintf(fp2, "\nStrings are equal\n");

else

fprintf(fp2, "\nStrings are not equal\n");

break;

default:

printf("Invalid option\n");

break;

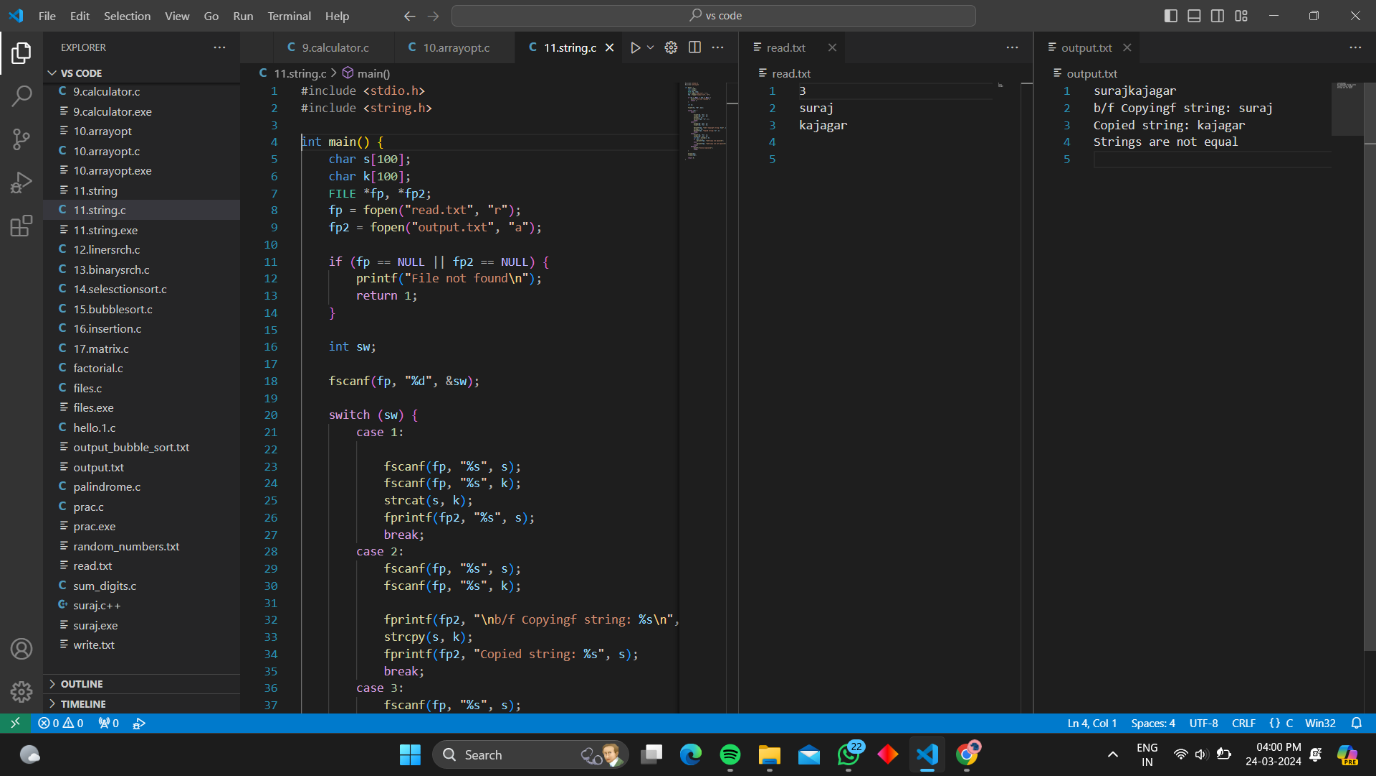
}

fclose(fp);

fclose(fp2);

return 0;

}



12. Linear Search: Search for an element in an array using linear search

#include <stdio.h>

#include <stdio.h>

#include <string.h>

int main() {

FILE \*fp, \*fp2;

fp = fopen("read.txt", "r");

fp2 = fopen("output.txt", "w");

int n, x, c = 0;

int a[100];

if (fp == NULL || fp2 == NULL) {

printf("File not found\n");

return 1;

}

fscanf(fp, "%d", &x);

fscanf(fp, "%d", &n);

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

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

fscanf(fp, "%d", &a[i]);

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

}

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

if (a[i] == x) {

c++;

fprintf(fp2, "found at position = %d\n", i + 1);

}

}

if (c == 0) {

fprintf(fp2, "not found\n");

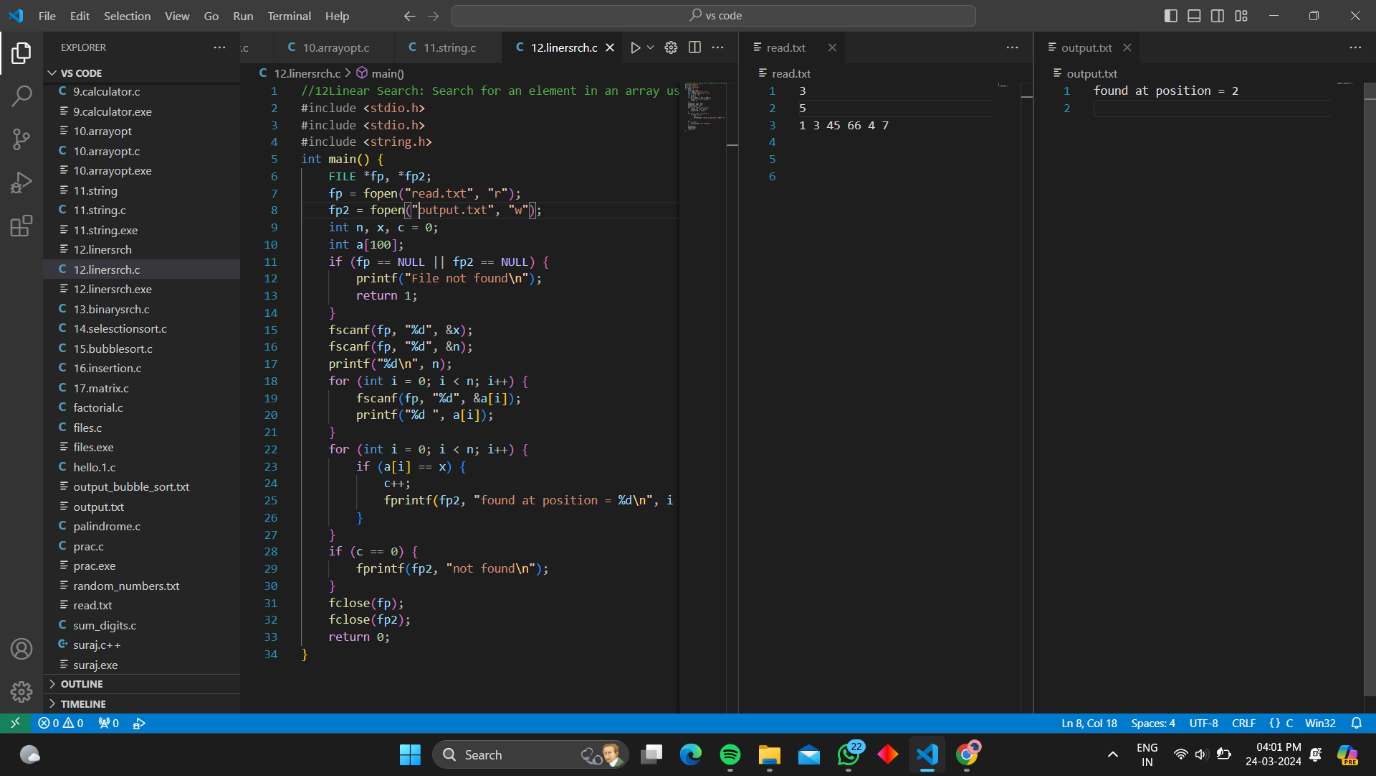
}

fclose(fp);

fclose(fp2);

return 0;

}



13. Binary Search: Search for an element in a sorted array using binary search

#include <stdio.h>

int main() {

FILE \*fp, \*fp2;

int flag = 0;

fp = fopen("read.txt", "r");

fp2 = fopen("output.txt", "w");

int n;

if (fp == NULL || fp2 == NULL) {

printf("File not found\n");

return 1;

}

fscanf(fp, "%d", &n);

printf("%d ",n);

int a[100];

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

fscanf(fp, "%d", &a[i]);

}

int key;

fscanf(fp,"%d",&key);

printf("%d ",key);

int left = 0;

int right = n - 1;

int result = -1;

while (left <= right) {

int mid = left + (right - left) / 2;

if (a[mid] == key) {

result = mid;

break;

}

if (a[mid] < key)

left = mid + 1;

else

right = mid - 1;

}

if (result != -1)

fprintf(fp2, "Element %d found at index %d\n", key, result);

else

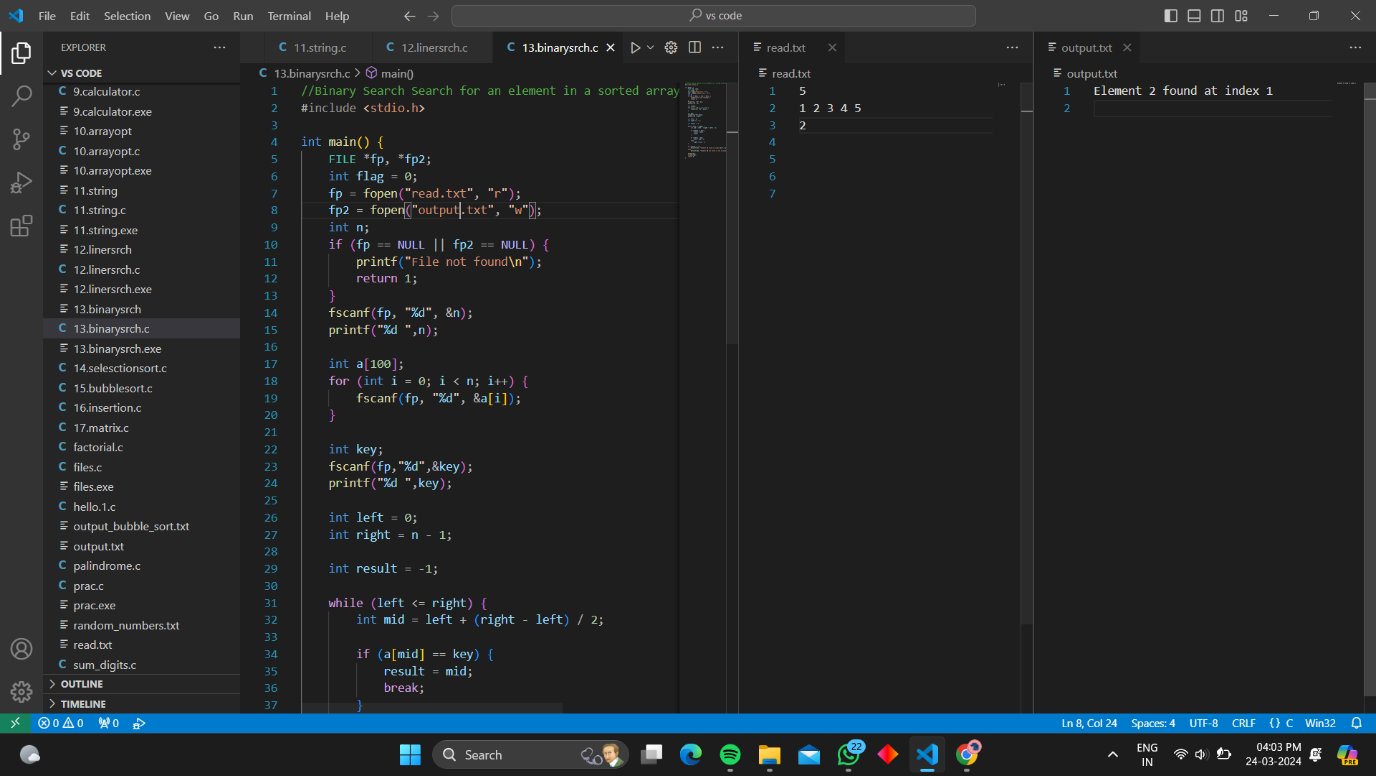
fprintf(fp2, "Element %d not found in the array\n", key);

fclose(fp);

fclose(fp2);

return 0;

}



14. Selection Sort: Sort an array using the selection sort

#include <stdio.h>

#include <stdio.h>

#include <string.h>

void selectionSort(int arr[], int n) {

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

int min\_idx = i;

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

if (arr[j] < arr[min\_idx])

min\_idx = j;

}

int temp = arr[i];

arr[i] = arr[min\_idx];

arr[min\_idx] = temp;

}

}

int main() {

FILE \*fp, \*fp2;

fp = fopen("read.txt", "r");

fp2 = fopen("output.txt", "w");

if (fp == NULL || fp2 == NULL) {

printf("File not found\n");

return 1;

}

int n;

fscanf(fp, "%d", &n);

int arr[n];

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

fscanf(fp, "%d", &arr[i]);

}

selectionSort(arr, n);

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

fprintf(fp2, "%d ", arr[i]);

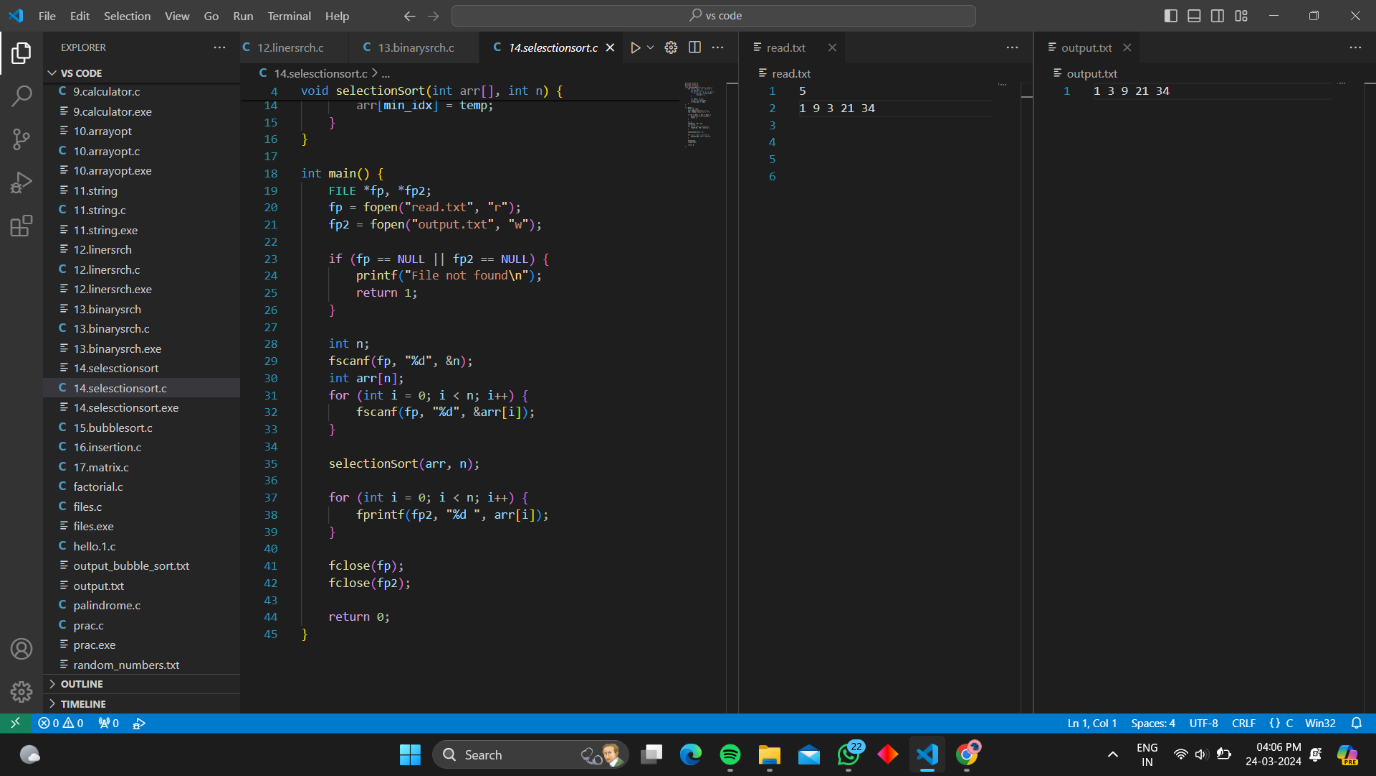
}

fclose(fp);

fclose(fp2);

return 0;

}



15. Bubble Sort: Sort an array using the bubble sort algorithm.

#include <stdio.h>

#include<stdio.h>

#include<stdlib.h>

void bubbleSort(int arr[], int n) {

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

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

if (arr[j] > arr[j + 1]) {

int temp = arr[j];

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

arr[j + 1] = temp;

}

}

}

}

int main() {

FILE \*fp, \*fp2;

fp = fopen("read.txt", "r");

fp2 = fopen("output.txt", "w");

if (fp == NULL || fp2 == NULL) {

printf("File not found\n");

return 1;

}

int n;

fscanf(fp, "%d", &n);

int arr[n];

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

fscanf(fp, "%d", &arr[i]);

}

bubbleSort(arr, n);

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

fprintf(fp2, "%d ", arr[i]);

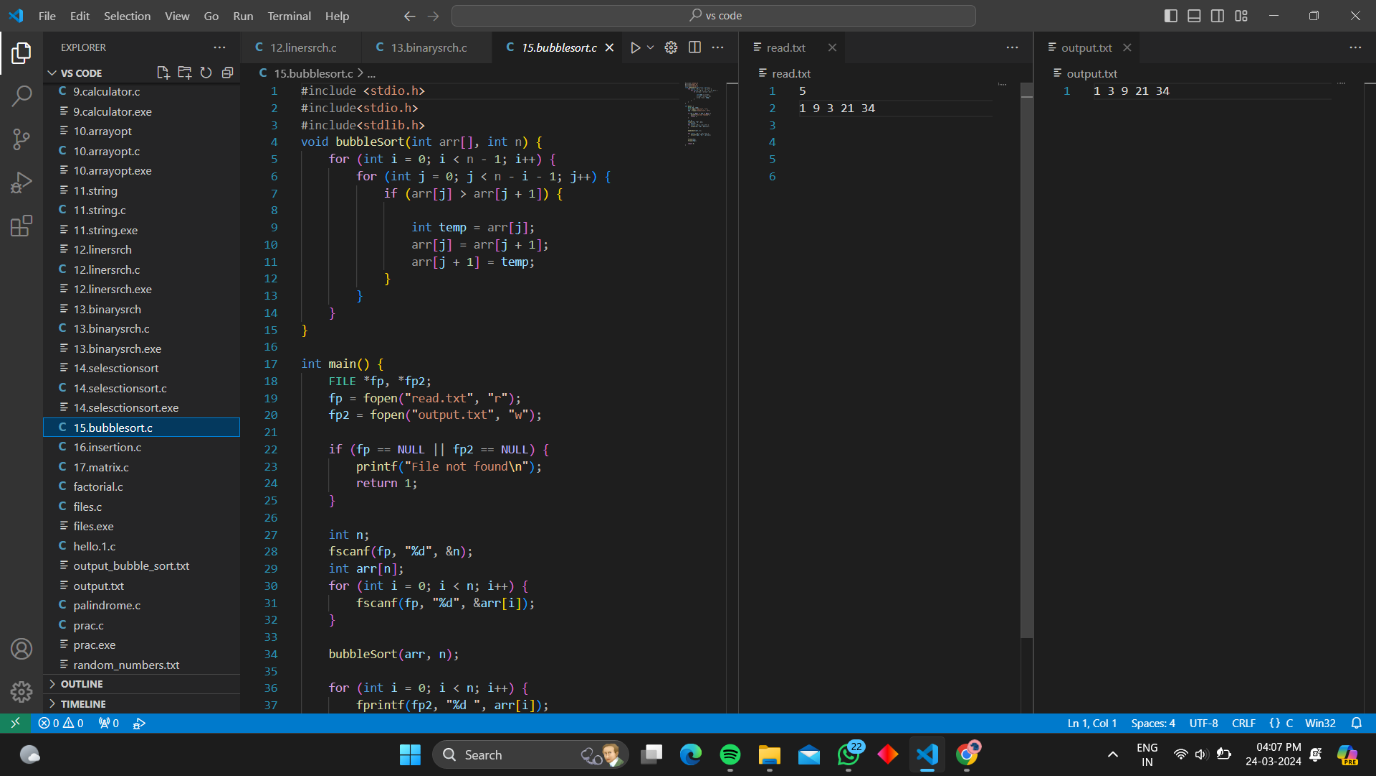
}

fclose(fp);

fclose(fp2);

return 0;

}



16. Insertion Sort: Sort an array using the insertion sort algorithm.

#include <stdio.h>

void insertionSort(int arr[], int n) {

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

int key = arr[i];

int j = i - 1;

while (j >= 0 && arr[j] > key) {

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

j = j - 1;

}

arr[j + 1] = key;

}

}

int main() {

FILE \*fp, \*fp2;

fp = fopen("read.txt", "r");

fp2 = fopen("output.txt", "w");

if (fp == NULL || fp2 == NULL) {

printf("File not found\n");

return 1;

}

int n;

fscanf(fp, "%d", &n);

int arr[n];

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

fscanf(fp, "%d", &arr[i]);

}

insertionSort(arr, n);

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

fprintf(fp2, "%d ", arr[i]);

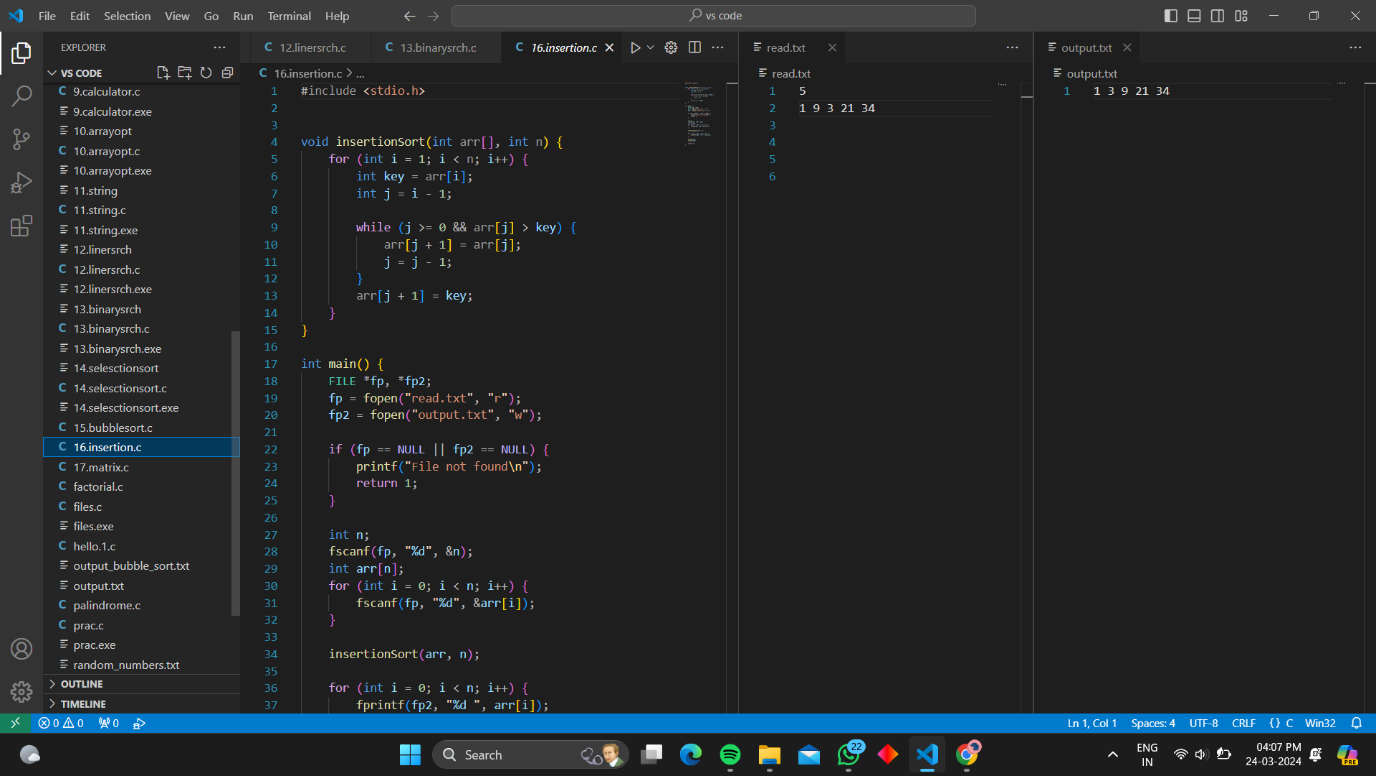
}

fclose(fp);

fclose(fp2);

return 0;

}



17. Matrix Operations: Perform matrix addition, subtraction, multiplication, and transpose.

#include <stdio.h>

void readM(FILE \*fp, int m[][100], int r, int c) {

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

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

fscanf(fp, "%d", &m[i][j]);

}

}

}

void print(FILE \*fp, int m[][100], int r, int c) {

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

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

fprintf(fp, "%d ", m[i][j]);

}

fprintf(fp, "\n");

}

}

void addM(int m1[][100], int m2[][100], int res[][100], int r, int c) {

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

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

res[i][j] = m1[i][j] + m2[i][j];

}

}

}

void subM(int m1[][100], int m2[][100], int res[][100], int r, int c) {

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

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

res[i][j] = m1[i][j] - m2[i][j];

}

}

}

void mulM(int mat1[][100], int mat2[][100], int res[][100], int r1, int c1, int c2) {

for (int i = 0; i < r1; i++) {

for (int j = 0; j < c2; j++) {

res[i][j] = 0;

for (int k = 0; k < c1; k++) {

res[i][j] += mat1[i][k] \* mat2[k][j];

}

}

}

}

void transpose(int m[][100], int tr[][100], int r, int c) {

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

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

tr[i][j] = m[j][i];

}

}

}

int main() {

FILE \*fp, \*fp2;

fp = fopen("read.txt", "r");

fp2 = fopen("output.txt", "w");

if (fp == NULL || fp2 == NULL) {

printf("File not found\n");

return 1;

}

int r1, c1, r2, c2;

fscanf(fp, "%d %d", &r1, &c1);

fscanf(fp, "%d %d", &r2, &c2);

if (r1 != r2 || c1 != c2) {

printf("Matrix operations require matrices of the same dimensions.\n");

fclose(fp);

fclose(fp2);

return 1;

}

int m1[100][100], m2[100][100], res[100][100], tr[100][100];

readM(fp, m1, r1, c1);

readM(fp, m2, r2, c2);

addM(m1, m2, res, r1, c1);

fprintf(fp2, "Matrix Addition Result:\n");

print(fp2, res, r1, c1);

subM(m1, m2, res, r1, c1);

fprintf(fp2, "\nMatrix Subtraction Result:\n");

print(fp2, res, r1, c1);

mulM(m1, m2, res, r1, c1, c2);

fprintf(fp2, "\nMatrix Multiplication Result:\n");

print(fp2, res, r1, c2);

transpose(m1, tr, r1, c1);

fprintf(fp2, "\nTranspose of Matrix 1:\n");

print(fp2, tr, c1, r1);

fclose(fp);

fclose(fp2);

return 0;

}