**Fibonacci series using recursion**

#include<iostream>

using namespace std;

int fibo(int n)

{

if(n<=1)

return n;

return fibo(n-1)+fibo(n-2);

}

int main()

{

int i,n=15;

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

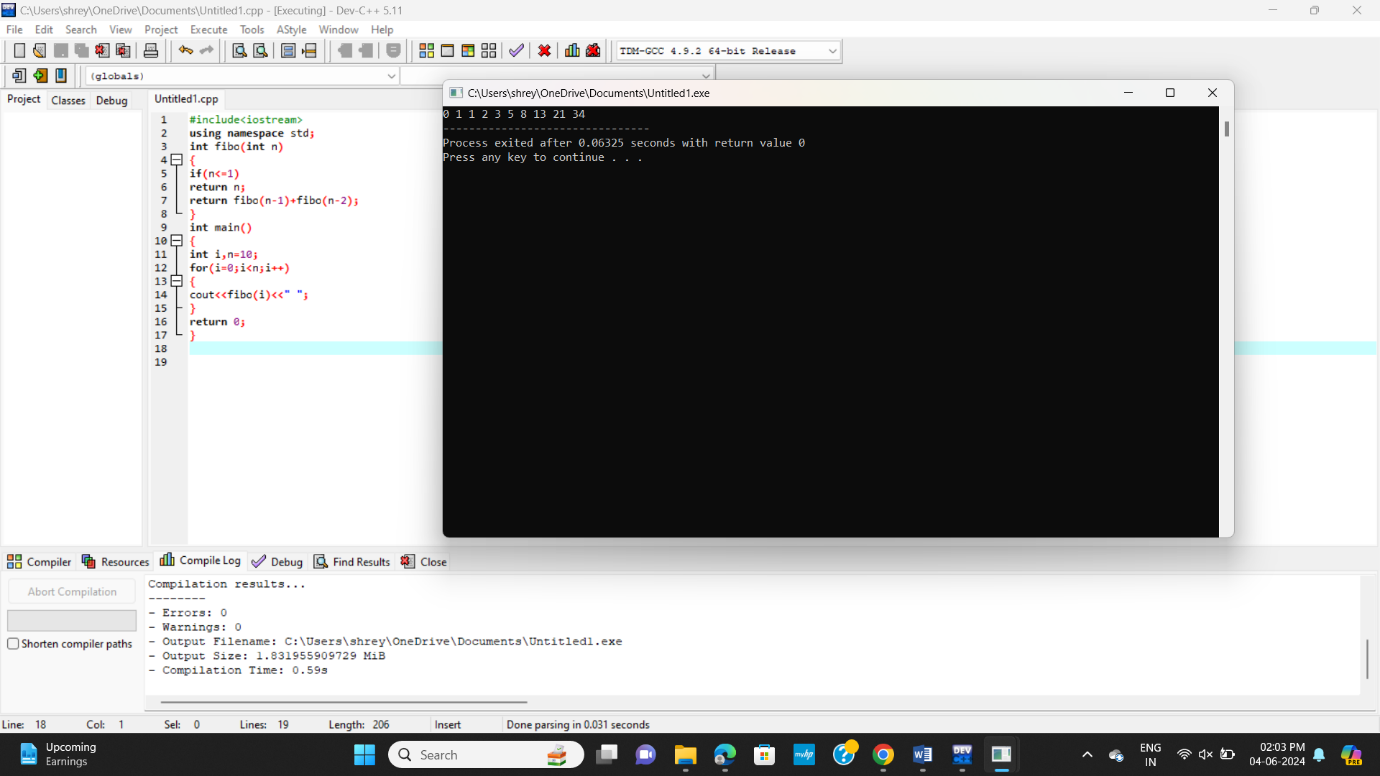
{

cout<<fibo(i)<<" ";

}

return 0;

}



**Armstrong**

#include <iostream>

#include <cmath>

using namespace std;

int check\_ArmstrongNumber(int num) {

if (num > 0)

return (pow(num % 10, 3) + check\_ArmstrongNumber(num / 10));

}

int main() {

int num;

cout << "Enter a number: ";

cin >> num;

if (check\_ArmstrongNumber(num) == num)

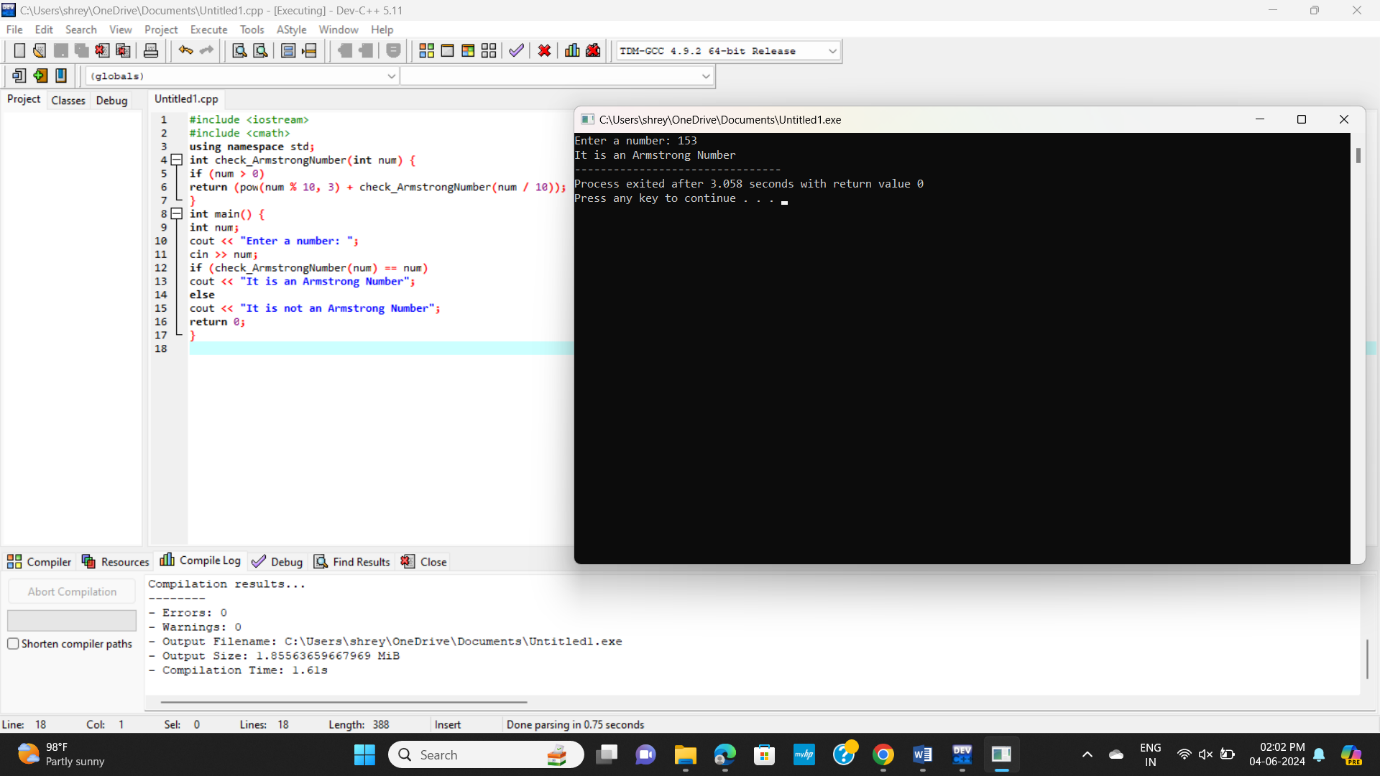
cout << "It is an Armstrong Number";

else

cout << "It is not an Armstrong Number";

return 0;

}



**Gcd**

#include<iostream>

using namespace std;

int gcd(int a, int b) {

if (a == 0 || b == 0)

return 0;

else if (a == b)

return a;

else if (a > b)

return gcd(a-b, b);

else return gcd(a, b-a);

}

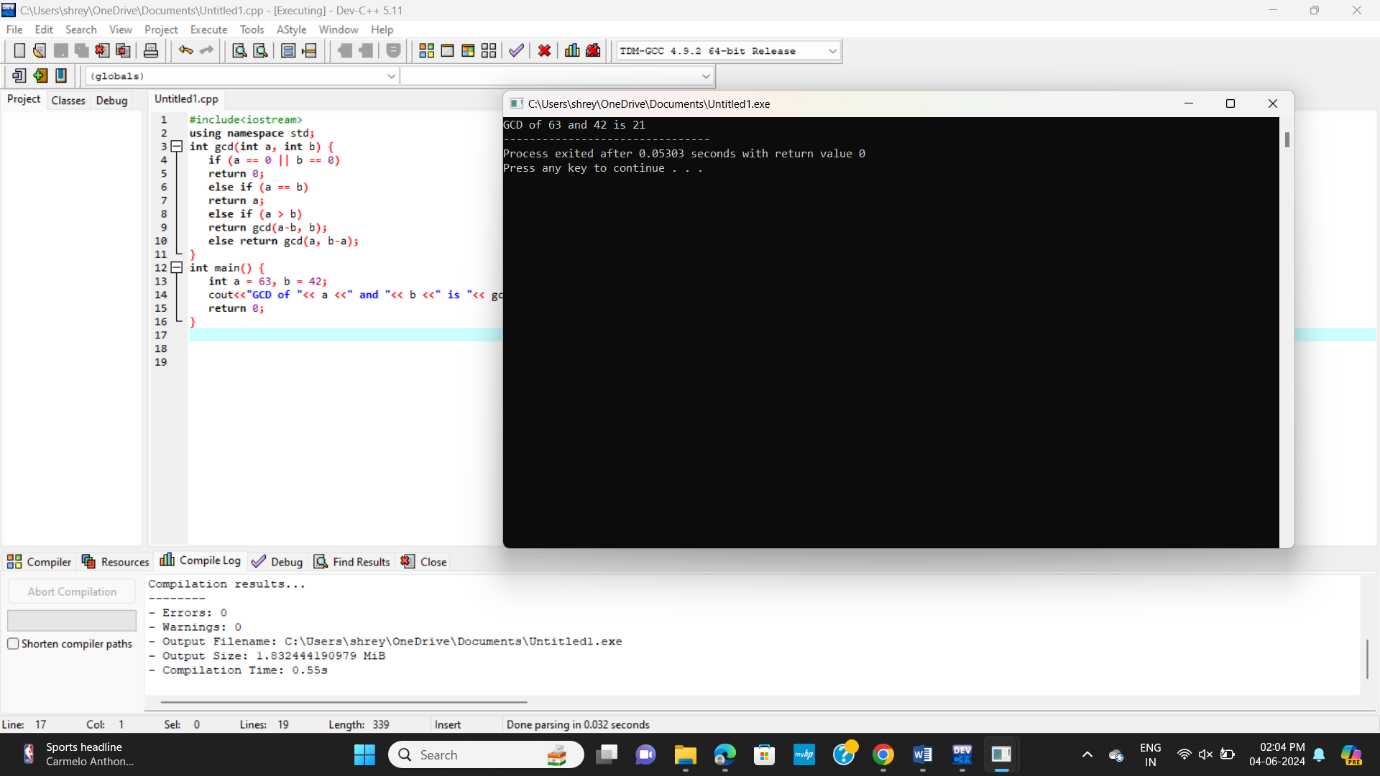
int main() {

int a = 63, b = 42;

cout<<"GCD of "<< a <<" and "<< b <<" is "<< gcd(a, b);

return 0;

}



**Largest number in an array**

#include <iostream>

using namespace std;

int findLargest(int arr[], int n) {

if (n == 1) {

return arr[0];

}

else {

return max(arr[n – 1], findLargest(arr, n – 1));

}

}

int main() {

int arr[] = {10, 23, 5, 45, 99, 20};

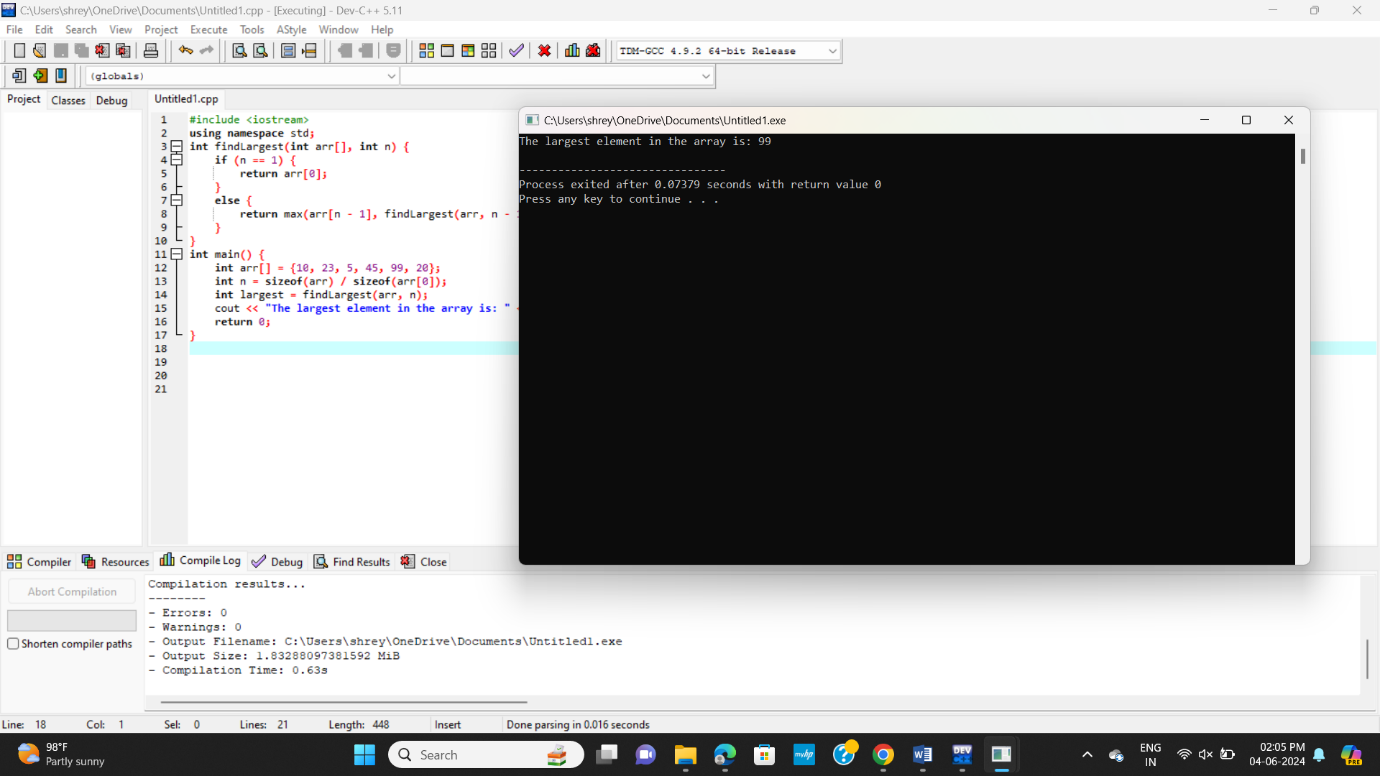
int n = sizeof(arr) / sizeof(arr[0]);

int largest = findLargest(arr, n);

cout << “The largest element in the array is: “ << largest << endl;

return 0;

}



**Factorial using recursion**

#include <iostream>

using namespace std;

int factorial(int n) {

if (n == 0) {

return 1;

}

else {

return n \* factorial(n - 1);

}

}

int main() {

int num;

cout << "Enter a integer: ";

cin >> num;

if (num < 0) {

cout << "Factorial is not defined for negative numbers." << endl;

} else {

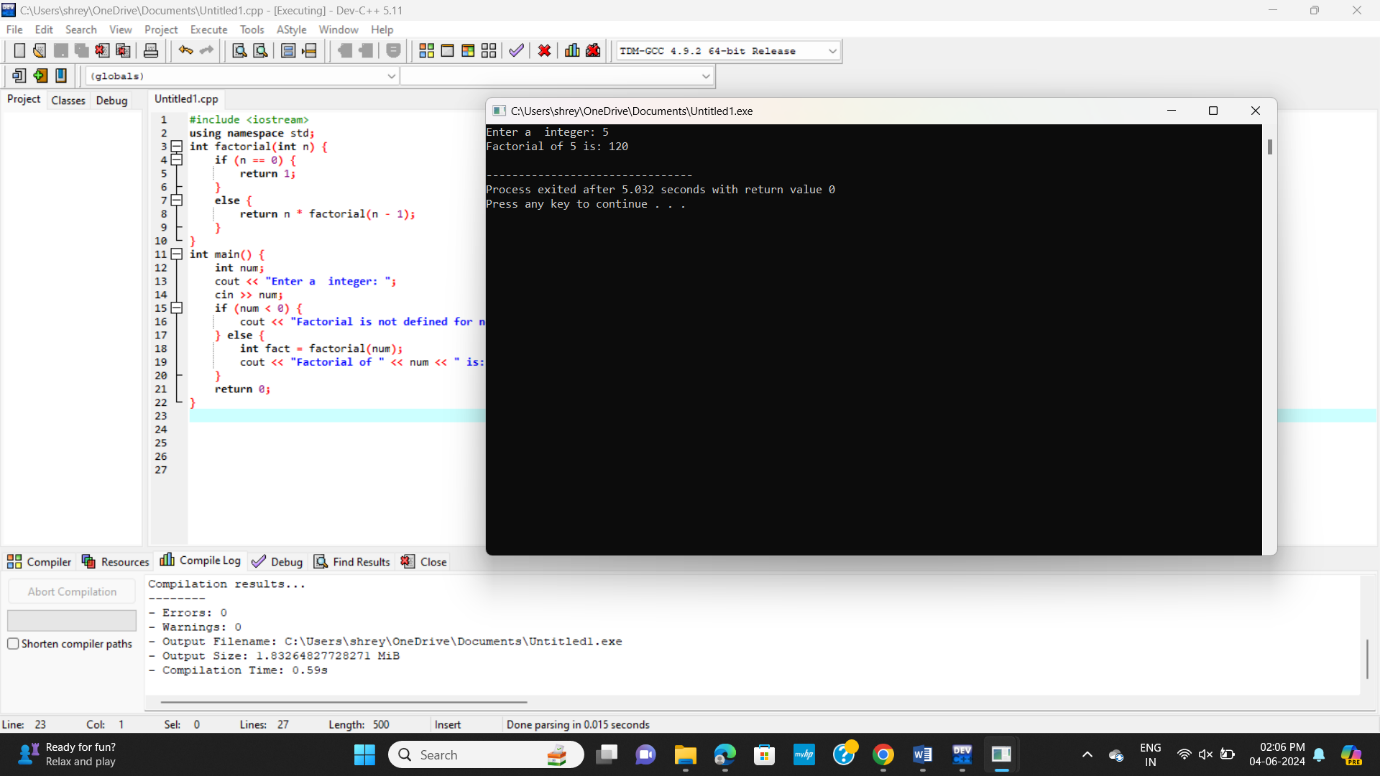
int fact = factorial(num);

cout << "Factorial of " << num << " is: " << fact << endl;

}

return 0;

}



**Reverse of a string**

#include <iostream>

using namespace std;

string reverse(string str, int len) {

if (len < 1) {

return “”;

}

if (len == 1) {

return string(1, str[0]);

}

return str[len – 1] + reverse(str, len – 1);

}

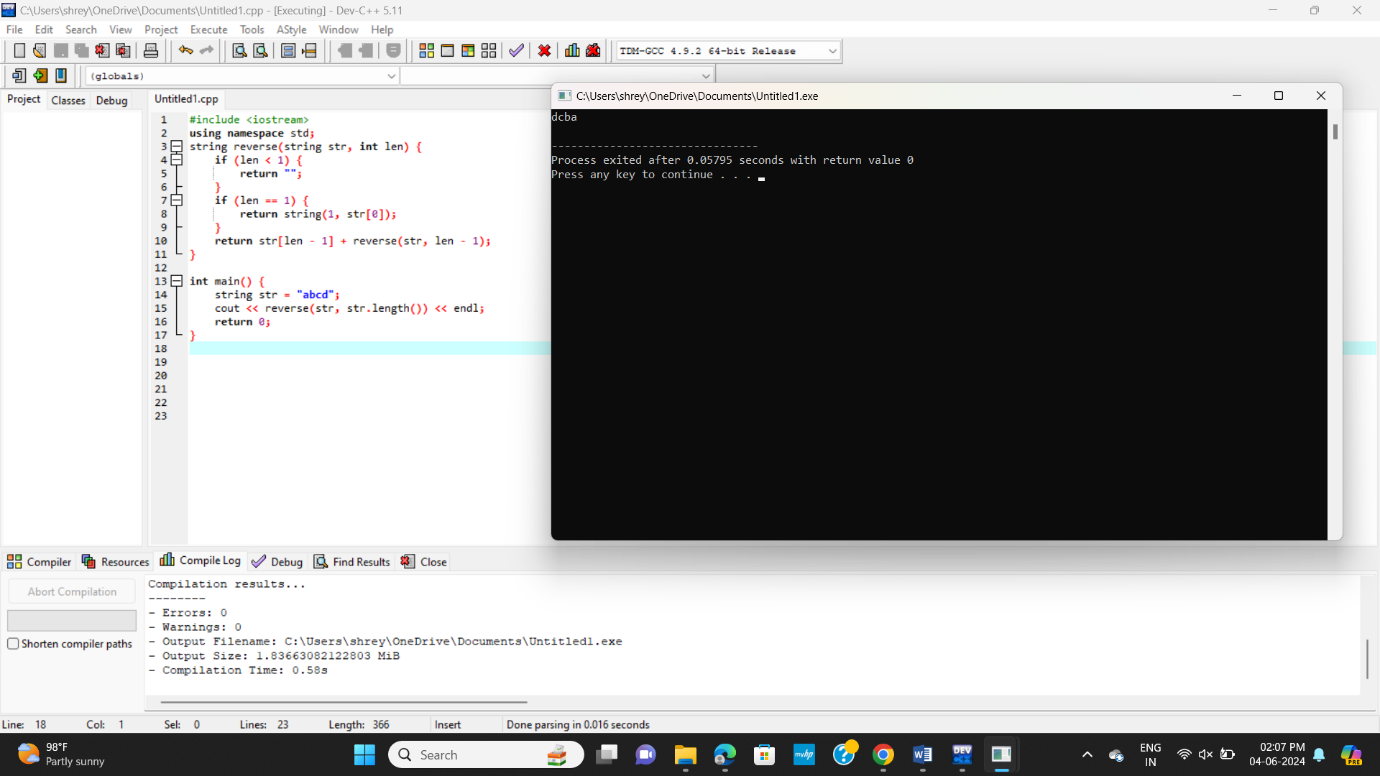
int main() {

string str = “abcd”;

cout << reverse(str, str.length()) << endl;

return 0;

}



**Prime number**

#include <iostream>

bool isPrime(int n, int i = 2) {

if (n <= 2) {

return (n == 2) ? true : false;

}

if (n % i == 0) {

return false;

}

if (i \* i > n) {

return true;

}

return isPrime(n, i + 1);

}

int main() {

int num;

printf("Enter a positive integer: ");

scanf("%d", &num);

if (isPrime(num)) {

printf("%d is a prime number.\n", num);

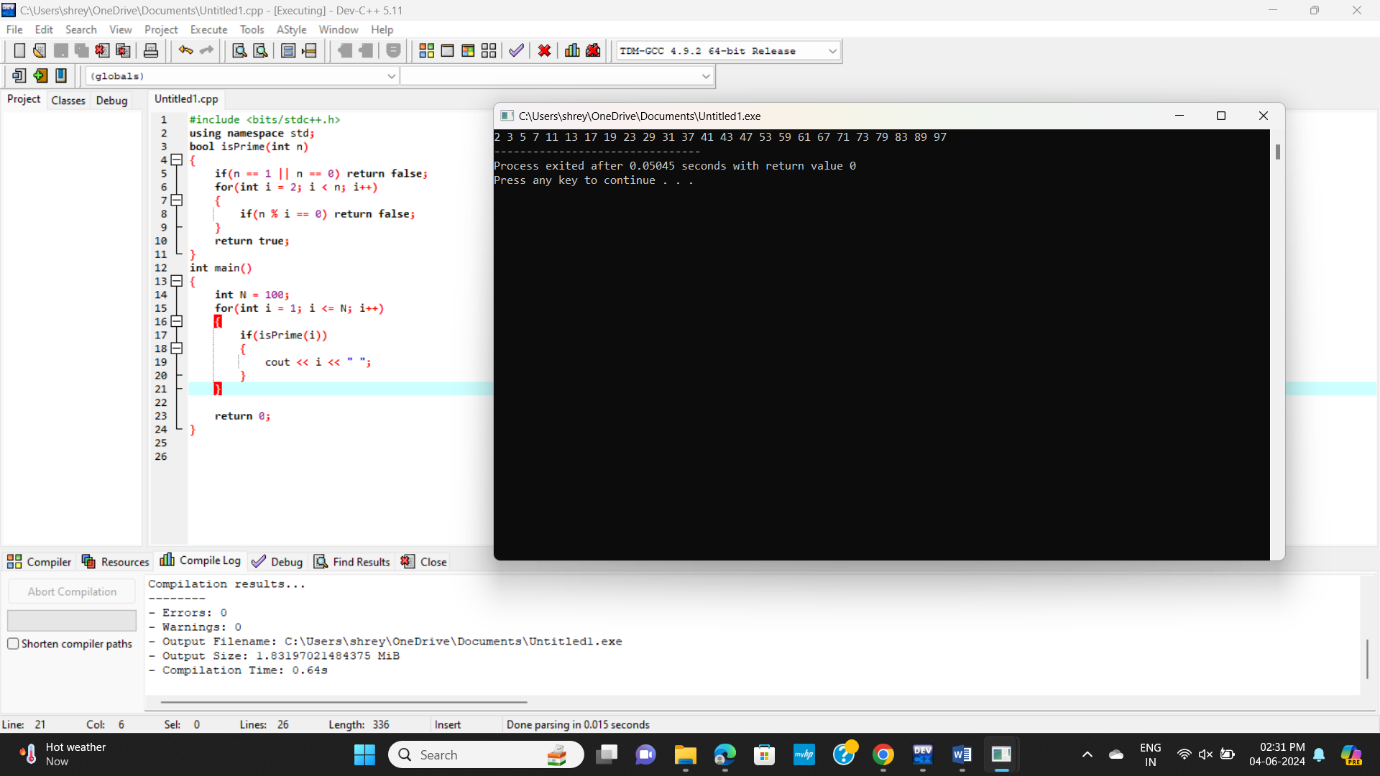
} else {

printf("%d is not a prime number.\n", num);

}

return 0;

}



**Check prime or not**

#include <iostream>

bool isPrime(int n, int divisor = 2) {

if (n <= 2) {

return (n == 2) ? true : false;

}

if (n % divisor == 0) {

return false;

}

if (divisor \* divisor > n) {

return true;

}

return isPrime(n, divisor + 1);

}

int main() {

int num;

printf("Enter a positive integer: ");

scanf("%d", &num);

if (isPrime(num)) {

printf("%d is a prime number.\n", num);

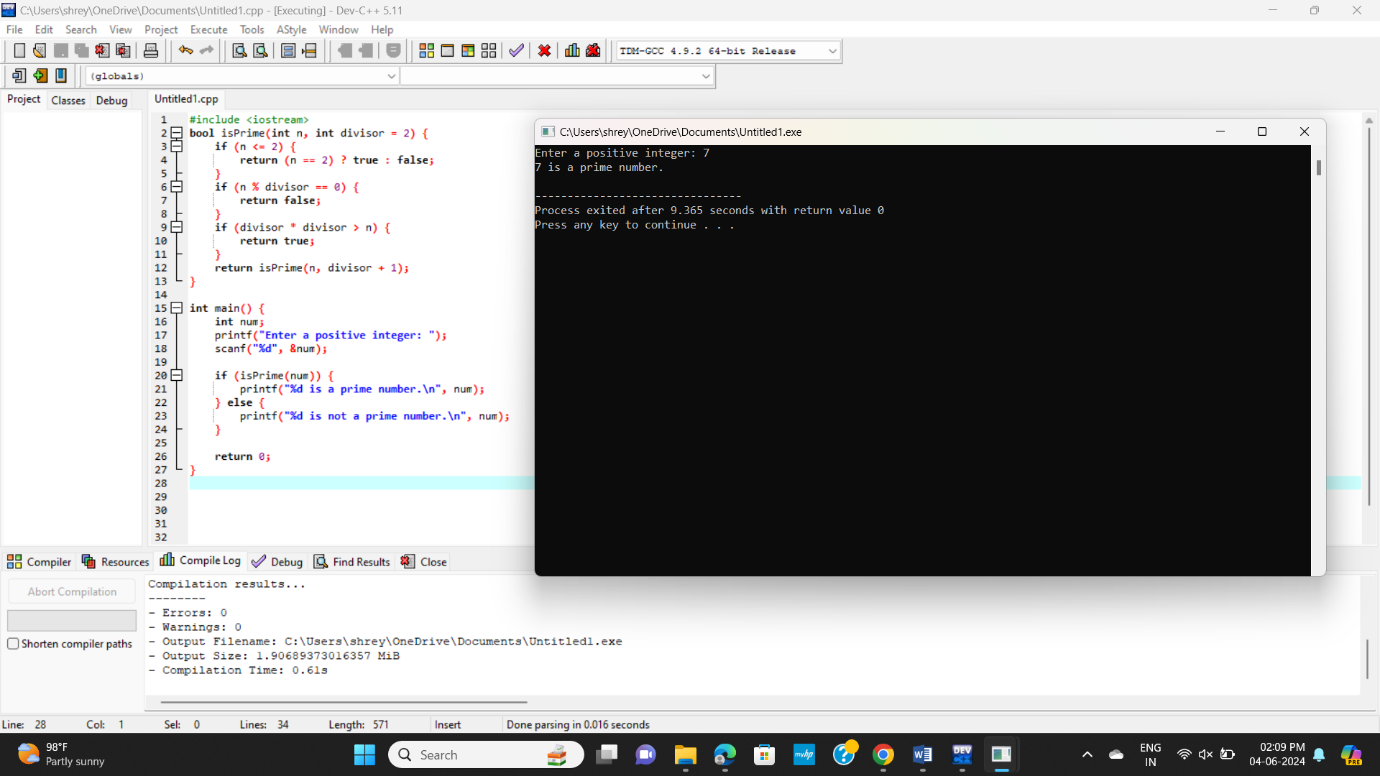
} else {

printf("%d is not a prime number.\n", num);

}

return 0;

}



**Palindrome**

#include <iostream>

#include <cstring>

bool isPalindrome(const char \*str, int start, int end) {

if (start >= end) {

return true;

}

if (str[start] != str[end]) {

return false;

}

return isPalindrome(str, start + 1, end - 1);

}

int main() {

char str[100];

printf("Enter a string: ");

fgets(str, 100, stdin);

int length = strlen(str);

if (str[length - 1] == '\n') {

str[length - 1] = '\0'; // removing the newline character

length--;

}

if (isPalindrome(str, 0, length - 1)) {

printf("%s is a palindrome.\n", str);

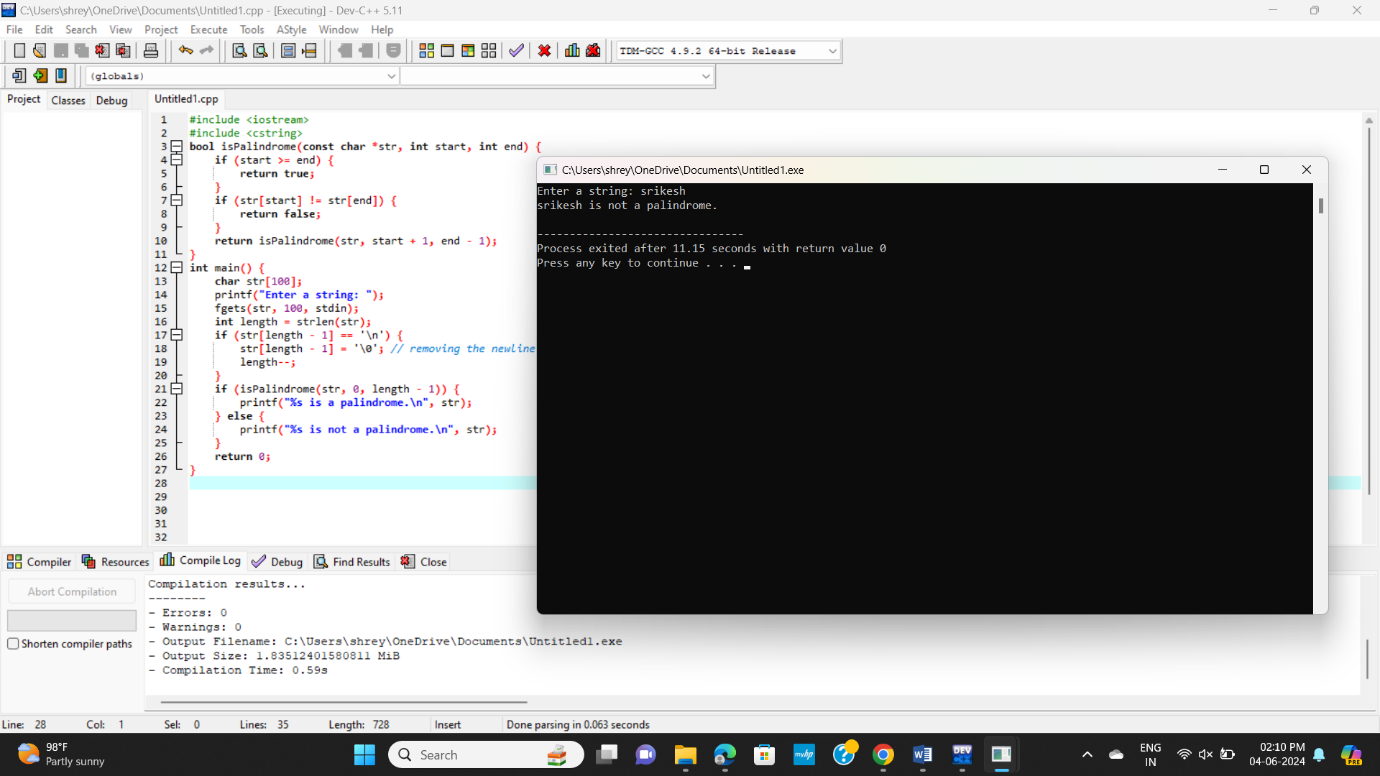
} else {

printf("%s is not a palindrome.\n", str);

}

return 0;

}



**Copy one string to another**

#include <iostream>

using namespace std;

void myCopy(char s1[], char s2[], int index = 0) {

s2[index] = s1[index];

if (s1[index] == '\0')

return;

myCopy(s1, s2, index + 1);

}

int main() {

char s1[100] = "abcd";

char s2[100] = "";

myCopy(s1, s2);

cout << s2;

return 0;

}

