**6 – Topshiriq**

***Ilhomjonov Iqbolshoh***

**2. Saralash algoritmlari murakkabligi nima ? Eng odddiy saralash algoritmlarining murakkabligi qanday ?**

Saralash algoritmlari bir nechta elementlar yoki obyektlarni tartibga solish uchun foydalaniladigan algoritmlardir. Algoritmlar murakkablik darajasiga ko'ra farqli bo'lishi mumkin. Qo'shimcha, murakkablik darajasi bir nechta faktorlarga bo'lingan. Algoritmlar murakkablik darajasi o'nlik sonlar sistemida ifodalangan va Murakkablik darajasi O(1), O(log n), O(n), O(n log n), O(n^2), O(2^n), O(n!) kabi bo'lishi mumkin. Oddiy saralash algoritmlari murakkablik darajasi murakkab emas, ya'ni O(n) yoki undan kam bo'lishi mumkin. Misol uchun, "Bubble Sort" algoritmi n^2 murakkablik darajasiga ega, ya'ni barcha elementlarni o'zaro taqqoslash va almashtirish uchun qiymatlar kattalashuvi yuzaga keladi. Bunday algoritmlar murakkablik darajasi yuqori bo'lgan katta miqdordagi ma'lumotlar uchun ishlatilishi to'g'risida maslahat berilmaydi. Boshqa oddiy saralash algoritmlari O(n log n) murakkablik darajasiga ega, ya'ni merge sort, quicksort va heap sort kabi algoritmlar. Bu algoritmlar katta miqdordagi ma'lumotlar uchun ishlatilishi uchun yaxshi vaqt kiritsa-da, ularning murakkablik darajasi yuqori bo'lgan katta miqdordagi ma'lumotlar uchun juda yuqori vaqt talab qilishi mumkin. Shuningdek, murakkablik darajasi O(2^n) yoki undan ham yuqori bo'lgan algoritmlar juda murakkab hisoblanadi va faqat kichik miqdordagi ma'lumotlar uchun ishlatilishi mumkin.

**4. Pufaksimon saralash algoritmining dasturini tuzing va murakkabligini baholang.**

#include <iostream>

using namespace std;

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]) {

                swap(arr[j], arr[j+1]);

            }

        }

    }

}

int main() {

    int arr[] = {64, 34, 25, 12, 22, 11, 90};

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

    bubbleSort(arr, n);

    cout<<"Sorted array: ";

    for(int i=0; i<n; i++) {

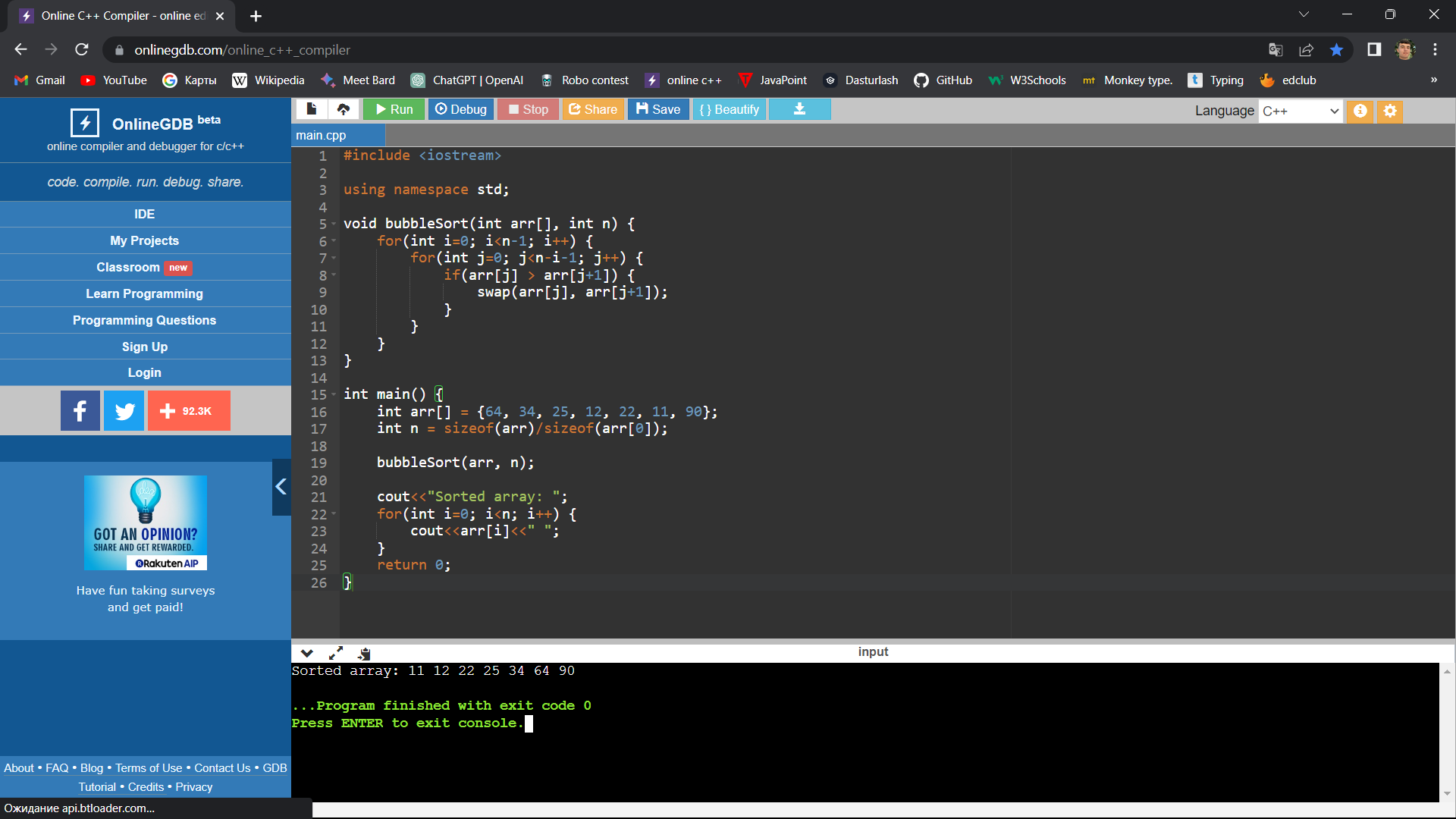
        cout<<arr[i]<<" ";

    }

    return 0;

}

//NATIJA:



**6. 20 ta elemantdan iborat massiv hosil qiling. Ularni ixtiyoriy sonlar generatori bilan to’ldiring va massivni kamayish tartibida saralang.**

#include <iostream>

using namespace std;

void bubble(int a[], int n)

{

    for (int i = 0; i < n - 1; i++)

    {

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

        {

            if (a[j] < a[j + 1])

            {

                swap(a[j], a[j + 1]);

            }

        }

    }

}

int main()

{

    int n;

    cout << " N = ";

    cin >> n;

    int a[n];

    for (int i = 0; i < n; i++)

    {

        cout << i+1 << " - son = ";

        cin >> a[i];

    }

    bubble(a, n);

    cout << " Kamayish tartibida " << endl;

    for (int i = 0; i < n; i++)

    {

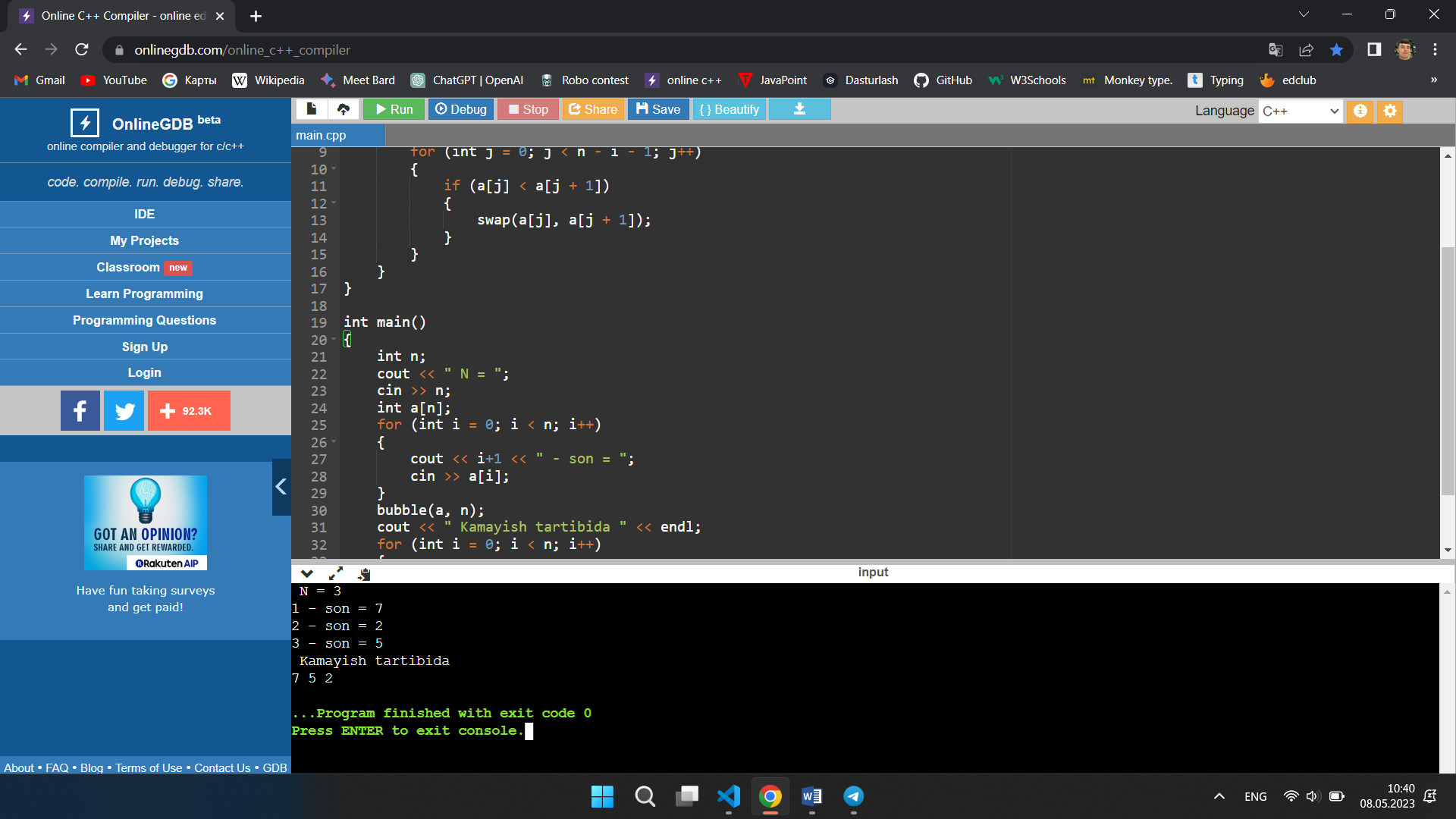
        cout << a[i] << " ";

    }

    return 0;

}

//NATIJA:



**8. Swap funksiyasinining vazifasi.**

swap() funksiyasi, iki o'zgaruvchini almashish uchun ishlatiladi. Odatda, bu o'zgaruvchilarning qiymatlarini almashtirish uchun ishlatiladi. swap() funksiyasi, C++-ning standart kutubxonasi ichida mavjud bo'lgan bir funksiya hisoblanadi.

swap() funksiyasi birlamchi parametr sifatida iki o'zgaruvchi qabul qiladi. Bu o'zgaruvchilarning qiymatlari almashtiriladi. Bu funksiyada o'zgaruvchilar pointerlar orqali uzatilishi mumkin.

Quyidagi ko'rinishdagi funksiya swap() funksiyasini o'zgartiradi:

void swap(int\* x, int\* y)

{

    int temp = \*x;

    \*x = \*y;

    \*y = temp;

}

Bu funksiya ikki ta butun sonni almashtiradi. x va y pointerlar orqali uzatilgan holda, qiymatlari almashtiriladi. Birinchi o'zgaruvchi qiymati temp o'zgaruvchisiga saqlanadi. Keyin x o'zgaruvchisining qiymati, y o'zgaruvchisining qiymati bilan almashtiriladi. So'ngra y o'zgaruvchisining qiymati, avvalgi x qiymati bilan almashtiriladi va funksiya tugatiladi.

**10. Swap orqali, Bubble\_sort nomli qiymat qaytarmaydigan funksiya bilan 30 ta elementdan iborat massivni o’sish tartibida saralang.**

#include <iostream>

using namespace std;

void Bubble\_sort(int a[])

{

    for (int i = 0; i < 30 - 1; i++)

    {

        for (int j = 0; j < 30 - i - 1; j++)

        {

            if (a[j] > a[j + 1])

            {

                swap(a[j], a[j + 1]);

            }

        }

    }

}

int main()

{

    int a[30];

    for (int i = 0; i < 30; i++)

    {

        cout << "   ";

        cin >> a[i];

    }

    Bubble\_sort(a);

    cout << " O'sish tartibida " << endl;

    for (int i = 0; i < 30; i++)

    {

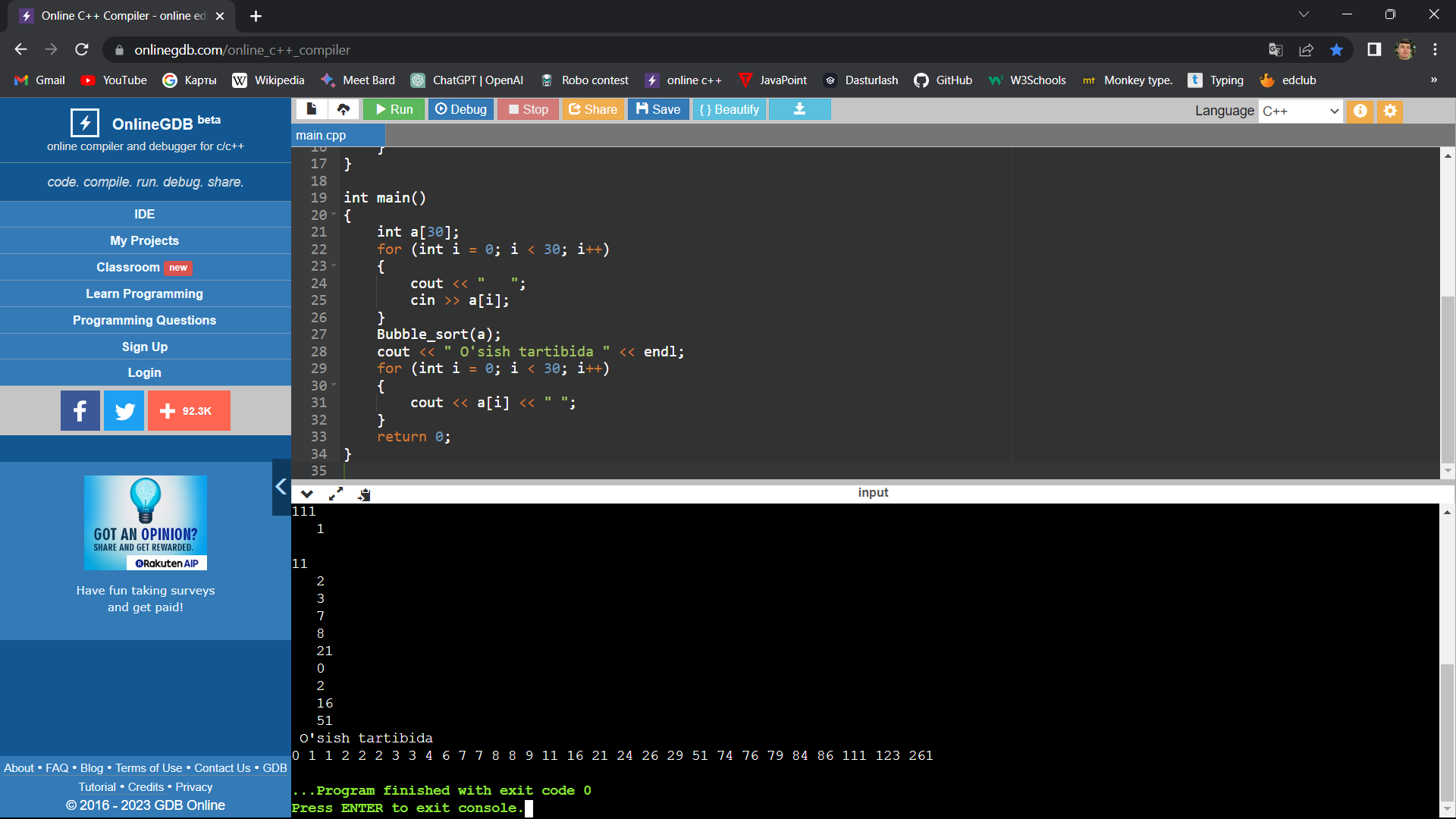
        cout << a[i] << " ";

    }

    return 0;

}

//NATIJA:



**12. Sheyker saralash algoritmining murakkabligini quyidagi holatlar bo’yicha baholang. Eng yaxshi baho, o’rtacha baho, eng yomon baho.**

Sheyker saralash algoritmi, murakkabligi o'nliklar xodisadagi raqamning darajasiga bog'liq bo'lgan qiymatlarni uziga qo'shish orqali amalga oshiriladi. Murakkablik esa qavsni saralashga bog'liq bo'lgan parametrlarga bog'liqdir.

**Eng yaxshi baho:** Sheyker saralash algoritmi, bo'shliqlikni tashqi (external) barcha saralash algoritmlari bilan bir xil miqdorda tugatadi, ammo barcha qiymatlarni yomon holatda bo'lmagan holda o'z ichiga oladi. Shuning uchun, bu algoritmning murakkabligi O(n log n) oralig'ida bo'ladi.

**O'rta baho:** Sheyker saralash algoritmi, unidirectional shifrlashlar yordamida takrorlanuvchi elementlarni bekor qiladi, shuningdek, qiymatlarni uziga qo'shish va ayirishlar yordamida barcha qiymatlarni bir xil miqdorda saralash imkonini beradi. Bunday sharoitda, murakkabligi O(n log n) va O(n log n) oralig'ida bo'ladi.

**Eng yomon baho:** Sheyker saralash algoritmi, juft elementlari uchrashuvda bekor qiladi. Bu, murakkabligining ortga qaytarilishiga olib keladi va algoritm murakkabligi O(n²) oralig'ida bo'ladi.

**14. Sheykar saralash algoritmining dasturini tuzing va murakkabligini baholang.**

#include <iostream>

using namespace std;

// Sheykar saralash funksiyasi

void sheykar\_sort(int arr[], int n)

{

    int m = n;

    bool swapped = true;

    while (m > 1 || swapped) {

        // m hajmi kamaytiriladi

        if (m > 1) {

            m = (m \* 10) / 13;

        }

        swapped = false;

        // elementlar almashishini bajarish

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

            int j = i + m;

            if (arr[i] > arr[j]) {

                swap(arr[i], arr[j]);

                swapped = true;

            }

        }

    }

}

int main()

{

    int arr[] = {5, 2, 9, 7, 1, 8};

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

    // Sheykar saralash funksiyasini chaqirish

    sheykar\_sort(arr, n);

    // saralgan massivni chiqarish

    for (int i = 0; i < n; i++) {

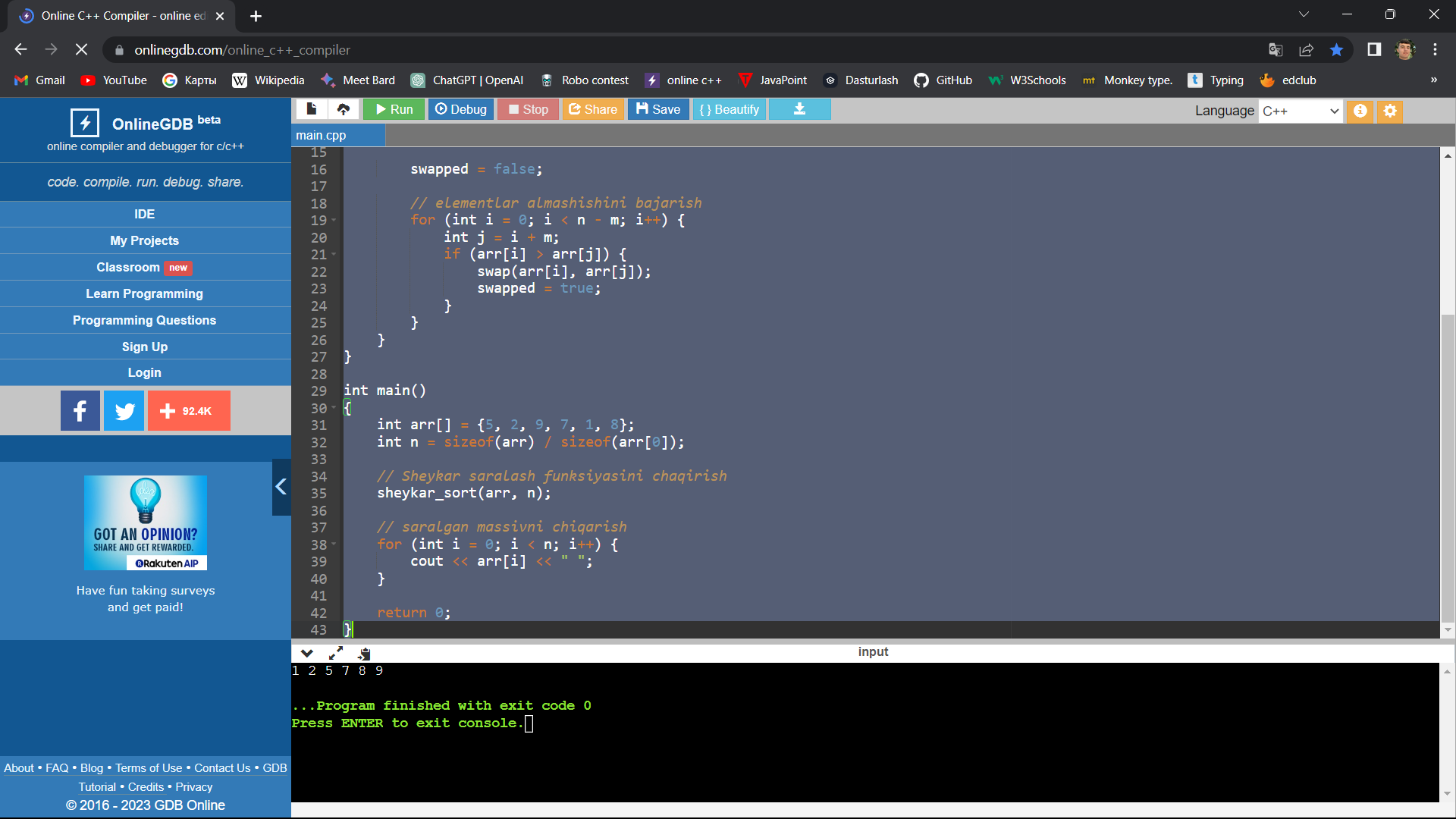
        cout << arr[i] << " ";

    }

    return 0;

}

//NATIJA:



**16. 20 ta elemantdan iborat massiv hosil qiling. Ularni ixtiyoriy sonlar generatori yordamida juft sonlar bilan to’ldiring va massivni kamayish tartibida saralang. (Sheyker saralash usuli yordamida)**

#include <bits/stdc++.h>

using namespace std;

void sheykar\_sort(int a[])

{

    int m = 20;

    bool swapped = true;

    while (m > 1 || swapped)

    {

        if (m > 1)

        {

            m = (m \* 10) / 13;

        }

        swapped = false;

        for (int i = 0; i < 20 - m; i++)

        {

            int j = i + m;

            if (a[i] > a[j])

            {

                swap(a[i], a[j]);

                swapped = true;

            }

        }

    }

}

int main()

{

    int a[20];

    for (int i = 0; i < 20; i++)

    {

        a[i] = rand()\*2%100;

    }

    sheykar\_sort(a);

    for (int i = 0; i < 20; i++)

    {

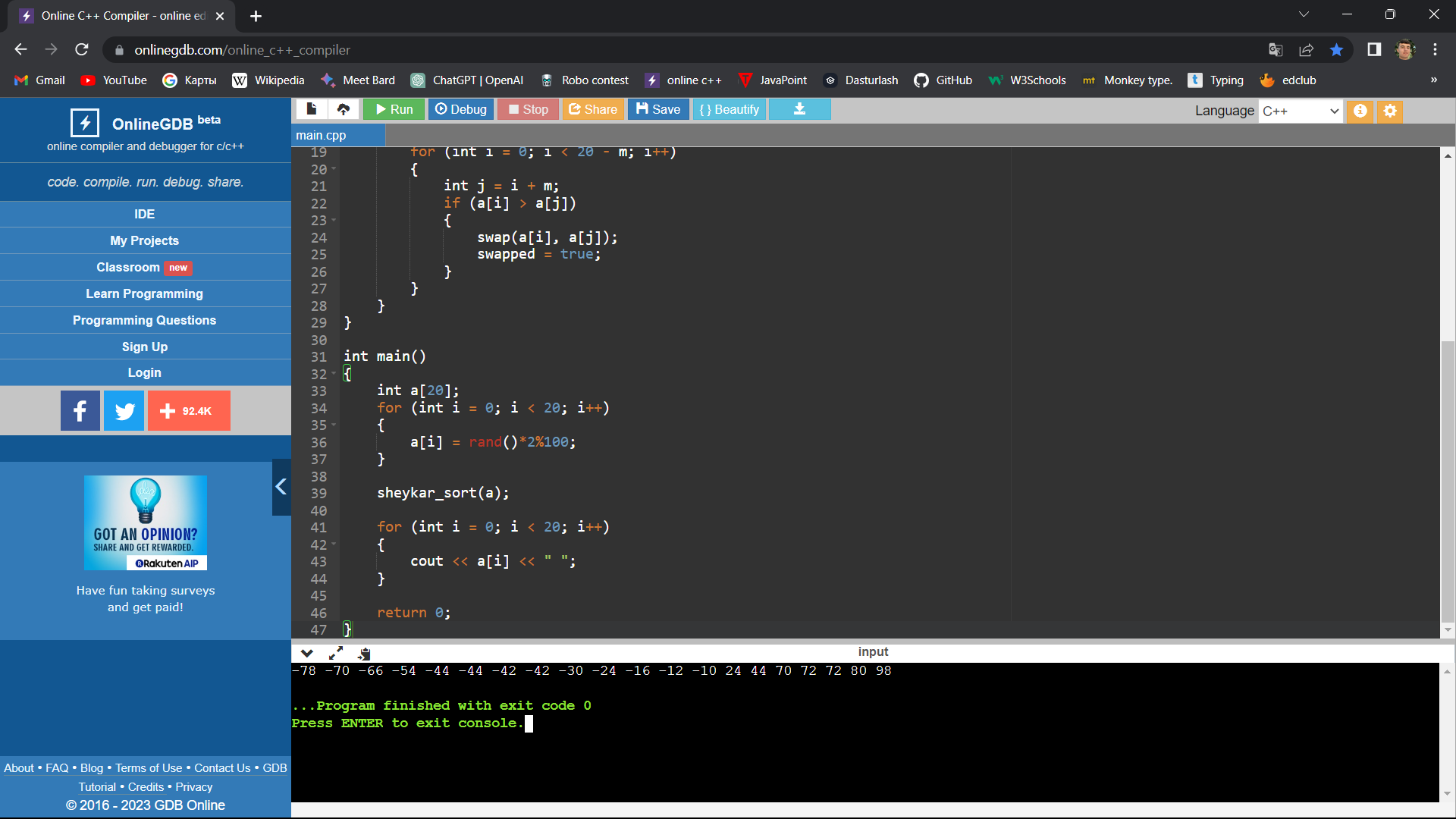
        cout << a[i] << " ";

    }

    return 0;

}

//NATIJA:



**18. Массивни 10 та ихтиёрий [0..100] оралиқдаги сонлар билан тўлдиринг ва охирги рақами бўйича сараланг. ( Sheyker usuli yordamida)**

#include <bits/stdc++.h>

using namespace std;

void sheykar\_sort(int a[])

{

    int m = 10;

    bool swapped = true;

    while (m > 1 || swapped)

    {

        if (m > 1)

        {

            m = (m \* 10) / 13;

        }

        swapped = false;

        for (int i = 0; i < 10 - m; i++)

        {

            int j = i + m;

            if (a[i] % 10 > a[j] % 10)

            {

                swap(a[i], a[j]);

                swapped = true;

            }

        }

    }

}

int main()

{

    int a[10];

    for (int i = 0; i < 10; i++)

    {

        a[i] = rand() % 100;

    }

    sheykar\_sort(a);

    for (int i = 0; i < 10; i++)

    {

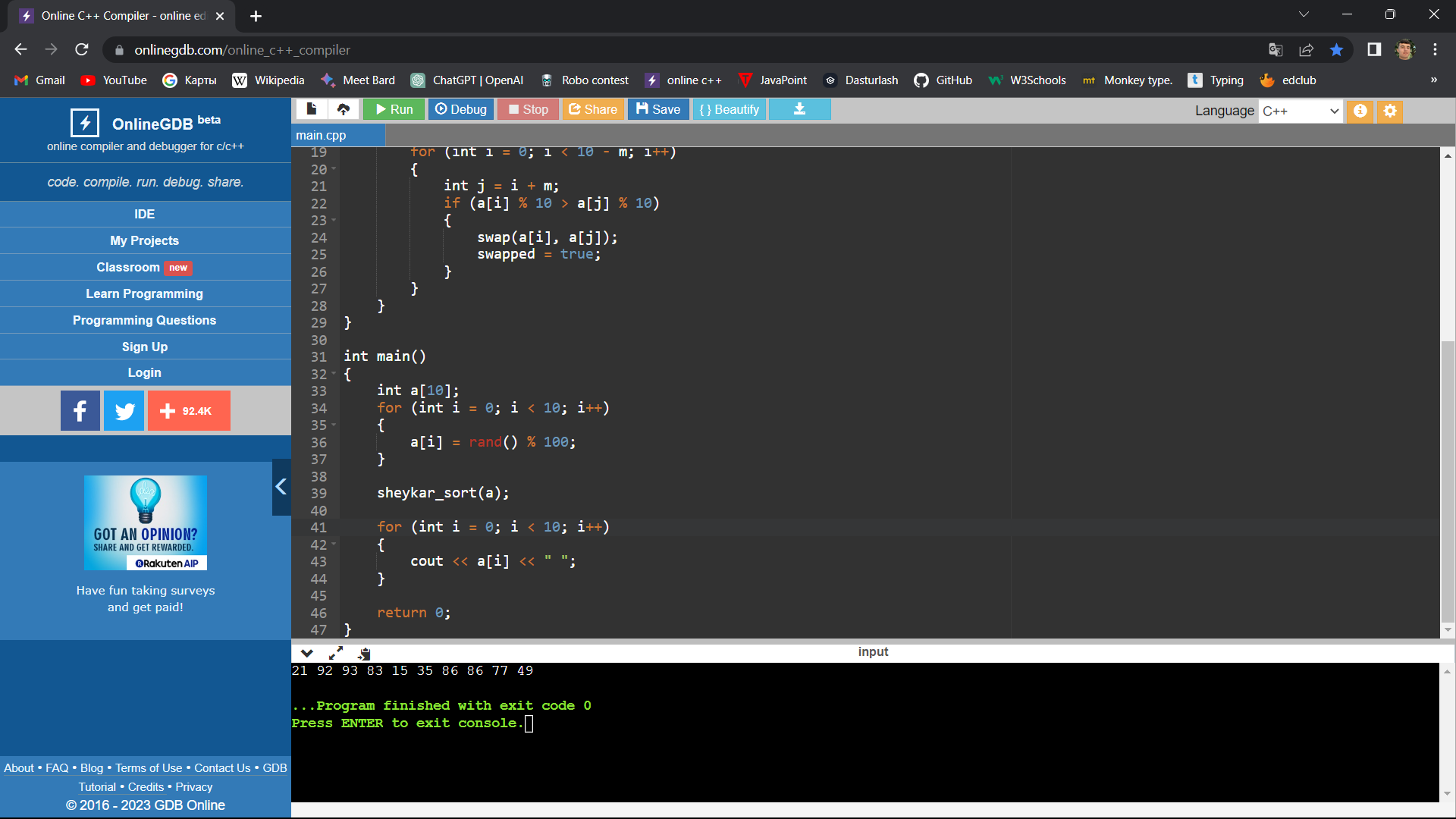
        cout << a[i] << " ";

    }

    return 0;

}

//NATIJA:



**20. Массивни 10 та ихтиёрий [0..100] ораллиқдаги сонлар билан тўлдиринг ва уни биринчи ярмини ўсиш тартибида, иккинчи ярмини эса камайиш тартибида сараланг. ( Bubble sort usuli yordamida)**

#include <iostream>

#include <cstdlib>

#include <ctime>

using namespace std;

void bubbleSort(int a[], int n, bool bul)

{

    for (int i = 0; i < n - 1; i++)

    {

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

        {

            if ((bul && a[j] > a[j + 1]) || (!bul && a[j] < a[j + 1]))

            {

                swap(a[i],a[j+1]);

            }

        }

    }

}

int main()

{

    int n = 10;

    int a[n];

    srand(time(nullptr));

    for (int i = 0; i < n; i++)

    {

        a[i] = rand() % 101;

    }

    cout << " Orginal massif : ";

    for (int i = 0; i < n; i++)

    {

        cout << a[i] << " ";

    }

    cout << endl;

    bubbleSort(a, n / 2, true);

    bubbleSort(a + n / 2, n / 2, false);

    cout << " Saralangan massif : ";

    for (int i = 0; i < n; i++)

    {

        cout << a[i] << " ";

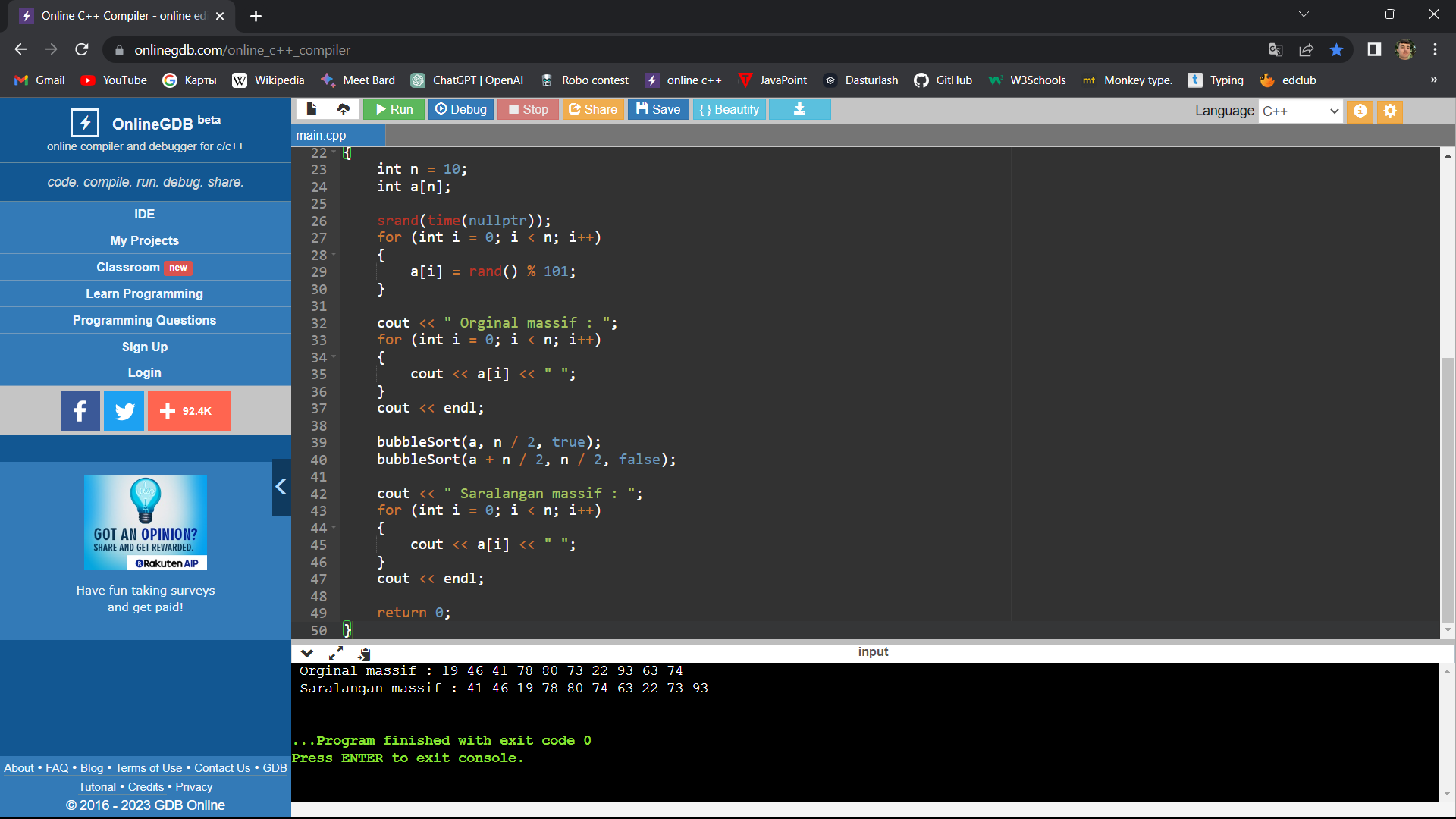
    }

    cout << endl;

    return 0;

}

//NATIJA:



**24. n ta elementdan iborat sonlar toplami berilgan. Toplam elementlari faqat 1 va 0 lardan iborat. Avval nollarni, keyin birlarni chop etuvchi dastur tuzing.**

#include <iostream>

using namespace std;

void bubble(int a[], int n)

{

    for (int i = 0; i < n - 1; i++)

    {

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

        {

            if (a[j] > a[j + 1])

            {

                swap(a[j], a[j + 1]);

            }

        }

    }

}

int main()

{

    int n;

    cout << " N = ";

    cin >> n;

    int a[n];

    for (int i = 0; i < n; i++)

    {

        a[i] = i % 2;

    }

    bubble(a, n);

    cout << " Ko'payish tartipida " << endl;

    for (int i = 0; i < n; i++)

    {

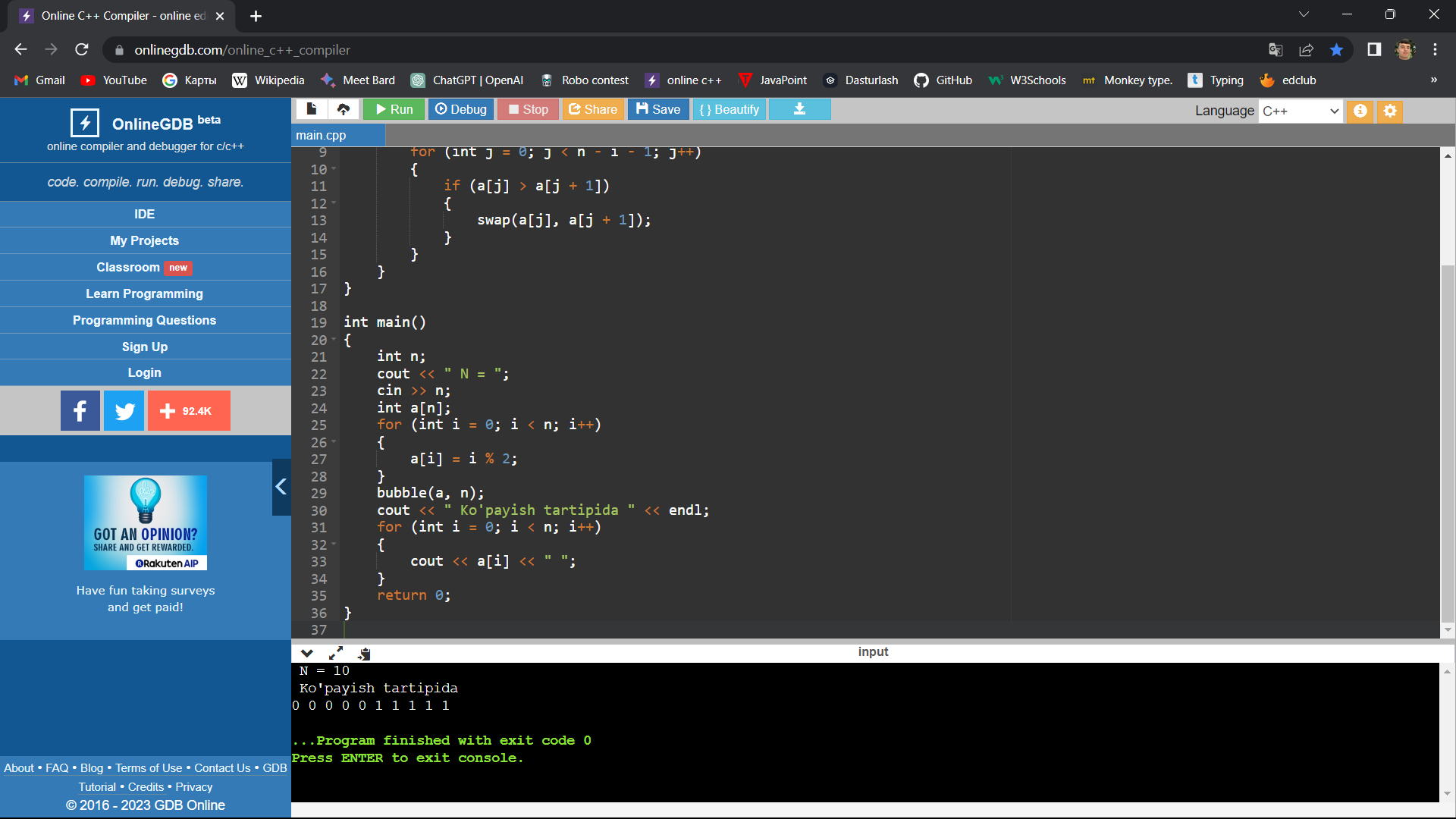
        cout << a[i] << " ";

    }

    return 0;

}

//NATIJA:



**26. Tanlash usuli orqali saralash algoritmining dasturini tuzing va murakkabligini baholang**

#include <iostream>

using namespace std;

void selectionSort(int arr[], int n)

{

    int minIndex;

    for (int i = 0; i < n - 1; i++)

    {

        minIndex = i;

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

        {

            if (arr[j] < arr[minIndex])

            {

                minIndex = j;

            }

        }

        swap(arr[i], arr[minIndex]);

    }

}

int main()

{

    int n = 10;

    int arr[n] = {12, 7, 3, 9, 5, 4, 8, 0, 6, 2};

    selectionSort(arr, n);

    for (int i = 0; i < n; i++)

    {

        cout << arr[i] << " ";

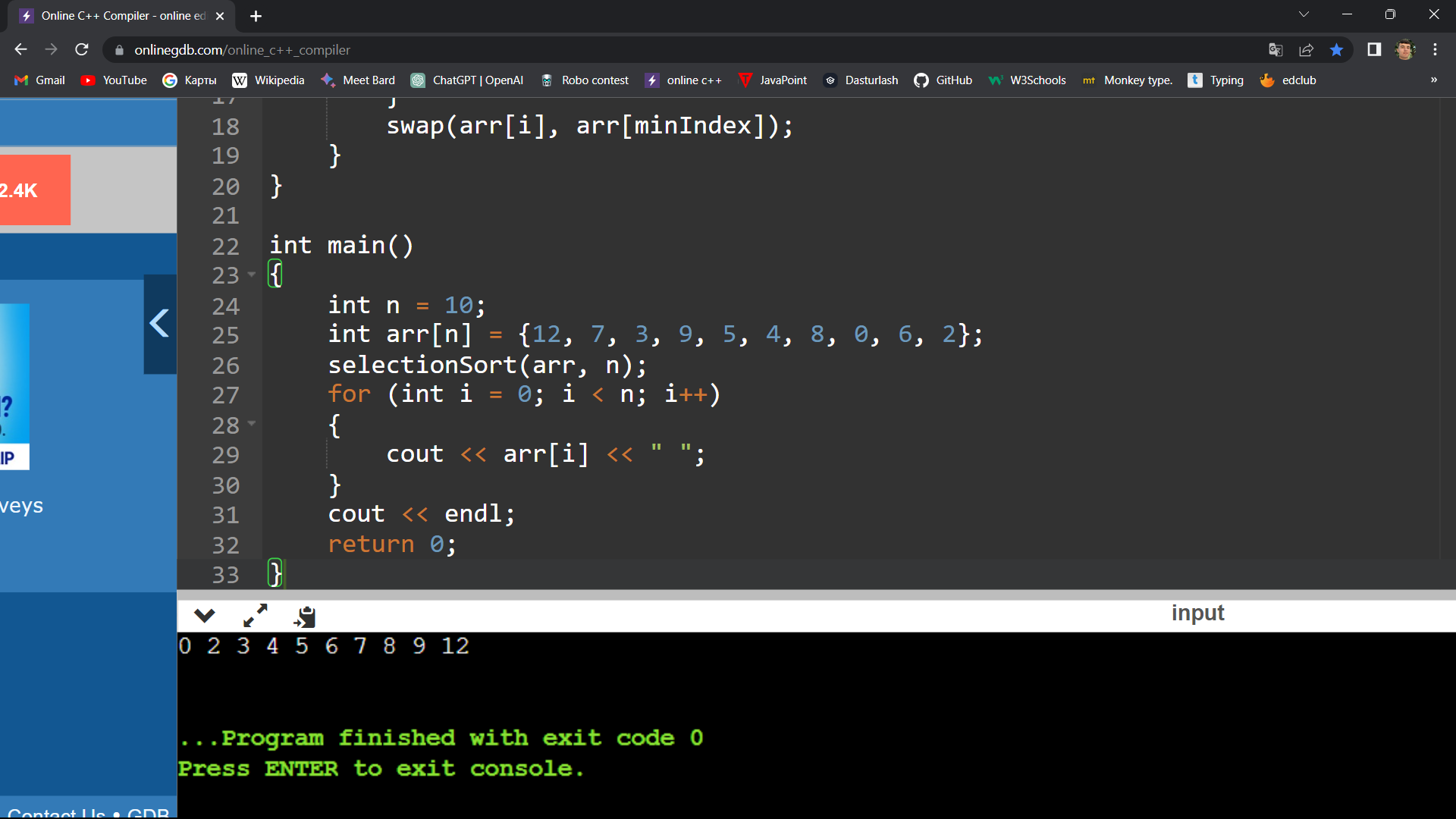
    }

    cout << endl;

    return 0;

}

//NATIJA:



**28. Tanlash saralash algoritmining murakkabligini quyidagi holatlar bo’yicha baholang. Eng yaxshi baho, o’rtacha baho, eng yomon baho.**

Selection sortning murakkabligi O(n^2)ga tengdir. Bu, ortiqcha muvaffaqiyatsiz taqqoslashlar soni va elementlarni almashishning qo'shma o'lchamli nusxalari yuzasidan kelib chiqadi. Murakkabligi esa quyidagi formulaga ko'ra hisoblanadi:

T(n) = (n-1) + (n-2) + ... + 2 + 1 Shu sababli, har bir yangi elementni tartibga solish uchun n ta taqqoslash amalga oshiriladi va har bir taqqoslash n elementni tekshirish va almashtirishni talab qiladi. Buna ko'ra murakkablik O(n^2)ga tengdir.

**30. 20 ta elemantdan iborat massiv hosil qiling. Ularni ixtiyoriy sonlar generatori yordamida juft sonlar bilan to’ldiring va massivni kamayish tartibida saralang. (Selection sort usuli yordamida)**

#include <bits/stdc++.h>

using namespace std;

void selection(int arr[], int n)

{

    int minIndex;

    for (int i = 0; i < n - 1; i++)

    {

        minIndex = i;

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

        {

            if (arr[j] > arr[minIndex])

            {

                minIndex = j;

            }

        }

        swap(arr[i], arr[minIndex]);

    }

}

int main()

{

    int n = 20;

    int arr[n];

    for (int i = 0; i < n; i++)

    {

        arr[i] = rand() % 500 \* 2;

    }

    selection(arr, n);

    for (int i = 0; i < n; i++)

    {

        cout << arr[i] << " ";

    }

    cout << endl;

    return 0;

}

//NATIJA:

