МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КОЛЕДЖ ЧЕРНІВЕЦЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

ІМЕНІ ЮРІЯ ФЕДЬКОВИЧА

**ЗВІТ**

про навчальну практику

студентки II курсу, спеціальності

«Комп`ютерні науки»

204 групи

*(Ватрич Юлія Юріївна)*

Період практики з “\_\_\_\_\_\_\_\_\_\_” 20\_\_ р. по “\_\_\_\_\_\_\_\_\_” 20\_\_ р.

Керівник практики від коледжу:

викл. Ковдриш В.В. \_\_\_\_\_\_\_\_\_\_

/підпис/

Оцінка:\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_

/підпис/

ЧЕРНІВЦІ, 2020

ТАБЛИЦЯ ОЦІНЮВАННЯ

|  |  |  |
| --- | --- | --- |
| Завдання | Оцінка за завдання | Підпис керівника практики |
| Завдання 1. |  |  |
| Завдання 2. |  |  |
| Завдання 3. |  |  |
| Завдання 4. |  |  |
| Завдання 5. |  |  |
| Завдання 6. |  |  |
| Завдання 7. |  |  |
| Завдання 8. |  |  |
| Завдання 9. |  |  |
| Завдання 10. |  |  |
| Завдання 11. |  |  |
| Завдання 12. |  |  |
| Завдання 13. |  |  |
| Завдання 14. |  |  |
| Оформлення звіту | |  |
| Захист звіту | |  |
| Підсумкова оцінка за навчальну практику | |  |

**Зміст**

[Завдання 1 7](#_heading=h.30j0zll)

[ПОСТАНОВКА ЗАДАЧІ 8](#_heading=h.1fob9te)

[Опис завдання 8](#_heading=h.3znysh7)

[ТЕОРЕТИЧНІ ВІДОМОСТІ 8](#_heading=h.2et92p0)

[СХЕМИ АЛГОРИТМІВ ОСНОВНИХ 8](#_heading=h.tyjcwt)

[ФУНКЦІЙ 8](#_heading=h.3dy6vkm)

[Висновок 8](#_heading=h.1t3h5sf)

[Додаток(а) 9](#_heading=h.4d34og8)

[Додаток(б) 9](#_heading=h.2s8eyo1)

[Завдання 2 10](#_heading=h.17dp8vu)

[ПОСТАНОВКА ЗАДАЧІ 11](#_heading=h.3rdcrjn)

[Опис завдання 11](#_heading=h.26in1rg)

[ТЕОРЕТИЧНІ ВІДОМОСТІ 11](#_heading=h.lnxbz9)

[СХЕМИ АЛГОРИТМІВ ОСНОВНИХ ФУНКЦІЙ 11](#_heading=h.35nkun2)

[Висновок 12](#_heading=h.1ksv4uv)

[Додаток(а) 12](#_heading=h.44sinio)

[Додаток(б) 14](#_heading=h.2jxsxqh)

[Завдання 3 15](#_heading=h.z337ya)

[ПОСТАНОВКА ЗАДАЧІ 16](#_heading=h.3j2qqm3)

[Опис завдання 16](#_heading=h.1y810tw)

[ТЕОРЕТИЧНІ ВІДОМОСТІ 16](#_heading=h.4i7ojhp)

[СХЕМИ АЛГОРИТМІВ ОСНОВНИХ ФУНКЦІЙ 16](#_heading=h.2xcytpi)

[Висновок 17](#_heading=h.1ci93xb)

[Додаток(а) 17](#_heading=h.3whwml4)

[Додаток(б) 19](#_heading=h.2bn6wsx)

[Завдання 4 20](#_heading=h.qsh70q)

[ПОСТАНОВКА ЗАДАЧІ 21](#_heading=h.3nqndbk)

[Опис завдання 21](#_heading=h.22vxnjd)

[ТЕОРЕТИЧНІ ВІДОМОСТІ 21](#_heading=h.i17xr6)

[СХЕМИ АЛГОРИТМІВ ОСНОВНИХ 21](#_heading=h.320vgez)

[ФУНКЦІЙ 21](#_heading=h.1h65qms)

[Висновок 22](#_heading=h.415t9al)

[Додаток(а) 22](#_heading=h.2gb3jie)

[Додаток(б) 24](#_heading=h.vgdtq7)

[Завдання 5 26](#_heading=h.1gf8i83)

[ПОСТАНОВКА ЗАДАЧІ 27](#_heading=h.3fg1ce0)

[Опис завдання 27](#_heading=h.1ulbmlt)

[ТЕОРЕТИЧНІ ВІДОМОСТІ 27](#_heading=h.4ekz59m)

[СХЕМИ АЛГОРИТМІВ ОСНОВНИХ 27](#_heading=h.2tq9fhf)

[ФУНКЦІЙ 27](#_heading=h.18vjpp8)

[Висновок 27](#_heading=h.3sv78d1)

[Додаток(а) 28](#_heading=h.280hiku)

[Додаток(б) 29](#_heading=h.3s49zyc)

[Завдання 6 30](#_heading=h.n5rssn)

[ПОСТАНОВКА ЗАДАЧІ 31](#_heading=h.375fbgg)

[Опис завдання 31](#_heading=h.1maplo9)

[ТЕОРЕТИЧНІ ВІДОМОСТІ 31](#_heading=h.46ad4c2)

[СХЕМИ АЛГОРИТМІВ ОСНОВНИХ ФУНКЦІЙ 32](#_heading=h.2lfnejv)

[Висновок 32](#_heading=h.10kxoro)

[Додаток(а) 32](#_heading=h.3kkl7fh)

[Додаток(б) 33](#_heading=h.1zpvhna)

[Завдання 7 34](#_heading=h.4jpj0b3)

[ПОСТАНОВКА ЗАДАЧІ 35](#_heading=h.2yutaiw)

[Опис завдання 35](#_heading=h.1e03kqp)

[ТЕОРЕТИЧНІ ВІДОМОСТІ 35](#_heading=h.3xzr3ei)

[СХЕМИ АЛГОРИТМІВ ОСНОВНИХ ФУНКЦІЙ 36](#_heading=h.2d51dmb)

[Висновок 36](#_heading=h.sabnu4)

[Додаток(а) 36](#_heading=h.3c9z6hx)

[Додаток(б) 39](#_heading=h.1qoc8b1)

[Завдання 8 40](#_heading=h.1rf9gpq)

[ПОСТАНОВКА ЗАДАЧІ 41](#_heading=h.4bewzdj)

[Опис завдання 41](#_heading=h.2qk79lc)

[ТЕОРЕТИЧНІ ВІДОМОСТІ 41](#_heading=h.15phjt5)

[СХЕМИ АЛГОРИТМІВ ОСНОВНИХ ФУНКЦІЙ 41](#_heading=h.3pp52gy)

[Висновок 41](#_heading=h.24ufcor)

[Додаток(а) 42](#_heading=h.jzpmwk)

[Додаток(б) 43](#_heading=h.42ddq1a)

[Завдання 9 44](#_heading=h.33zd5kd)

[ПОСТАНОВКА ЗАДАЧІ 45](#_heading=h.1j4nfs6)

[Опис завдання 45](#_heading=h.434ayfz)

[ТЕОРЕТИЧНІ ВІДОМОСТІ 45](#_heading=h.2i9l8ns)

[СХЕМИ АЛГОРИТМІВ ОСНОВНИХ ФУНКЦІЙ 45](#_heading=h.xevivl)

[Висновок 45](#_heading=h.3hej1je)

[Додаток(а) 46](#_heading=h.1wjtbr7)

[Додаток(б) 47](#_heading=h.4gjguf0)

[Завдання 10 48](#_heading=h.2vor4mt)

[ПОСТАНОВКА ЗАДАЧІ 49](#_heading=h.1au1eum)

[Опис завдання 49](#_heading=h.3utoxif)

[ТЕОРЕТИЧНІ ВІДОМОСТІ 49](#_heading=h.29yz7q8)

[СХЕМИ АЛГОРИТМІВ ОСНОВНИХ ФУНКЦІЙ 49](#_heading=h.p49hy1)

[Висновок 50](#_heading=h.393x0lu)

[Додаток(а) 50](#_heading=h.1o97atn)

[Додаток(б) 51](#_heading=h.488uthg)

[Завдання 11 52](#_heading=h.2ne53p9)

[ПОСТАНОВКА ЗАДАЧІ 53](#_heading=h.12jfdx2)

[Опис завдання 53](#_heading=h.3mj2wkv)

[ТЕОРЕТИЧНІ ВІДОМОСТІ 53](#_heading=h.21od6so)

[СХЕМИ АЛГОРИТМІВ ОСНОВНИХ ФУНКЦІЙ 53](#_heading=h.gtnh0h)

[Висновок 54](#_heading=h.30tazoa)

[Додаток(а) 54](#_heading=h.1fyl9w3)

[Додаток(б) 58](#_heading=h.3zy8sjw)

[Завдання 12 60](#_heading=h.2f3j2rp)

[ПОСТАНОВКА ЗАДАЧІ 61](#_heading=h.u8tczi)

[Опис завдання 61](#_heading=h.3e8gvnb)

[ТЕОРЕТИЧНІ ВІДОМОСТІ 61](#_heading=h.1tdr5v4)

[СХЕМИ АЛГОРИТМІВ ОСНОВНИХ ФУНКЦІЙ 61](#_heading=h.4ddeoix)

[Висновок 62](#_heading=h.2sioyqq)

[Додаток(а) 62](#_heading=h.17nz8yj)

[Додаток(б) 64](#_heading=h.3rnmrmc)

[Завдання 13 65](#_heading=h.26sx1u5)

[ПОСТАНОВКА ЗАДАЧІ 66](#_heading=h.ly7c1y)

[Опис завдання 66](#_heading=h.35xuupr)

[ТЕОРЕТИЧНІ ВІДОМОСТІ 66](#_heading=h.1l354xk)

[СХЕМИ АЛГОРИТМІВ ОСНОВНИХ ФУНКЦІЙ 66](#_heading=h.452snld)

[Висновок 67](#_heading=h.2k82xt6)

[Додаток(а) 67](#_heading=h.zdd80z)

[Додаток(б) 69](#_heading=h.1c1lvlb)

[Завдання 14 70](#_heading=h.3jd0qos)

[ПОСТАНОВКА ЗАДАЧІ 71](#_heading=h.1yib0wl)

[Опис завдання 71](#_heading=h.4ihyjke)

[ТЕОРЕТИЧНІ ВІДОМОСТІ 71](#_heading=h.2xn8ts7)

[СХЕМИ АЛГОРИТМІВ ОСНОВНИХ ФУНКЦІЙ 71](#_heading=h.1csj400)

[Висновок 71](#_heading=h.3ws6mnt)

[Додаток(а) 71](#_heading=h.2bxgwvm)

[Додаток(б) 71](#_heading=h.r2r73f)

Посилання на Google сайт:

<https://sites.google.com/stud.chnu.edu.ua/digitalage/%D0%B3%D0%BE%D0%BB%D0%BE%D0%B2%D0%BD%D0%B0-%D1%81%D1%82%D0%BE%D1%80%D1%96%D0%BD%D0%BA%D0%B0>

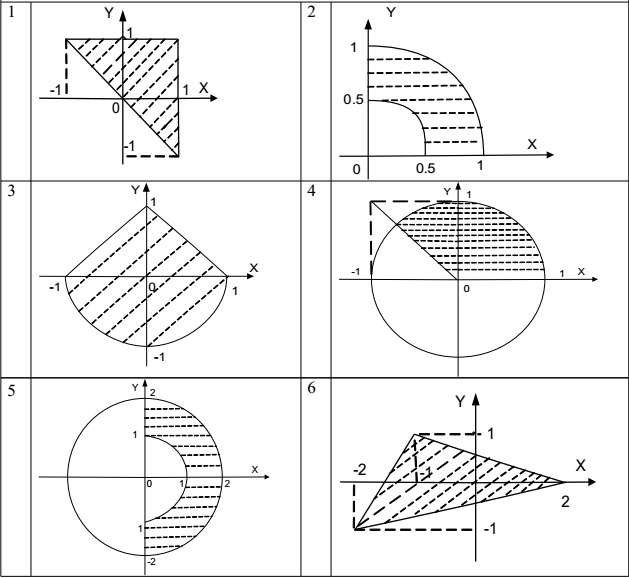
Посилання на GitHub:

<https://github.com/Juliaa532/praktuka.git>

1. **Завдання 1**
   1. **ПОСТАНОВКА ЗАДАЧІ**
      1. **Опис завдання**

Варіант-5

1.1Створити алгоритм та написати програму для розвязку наступної задачі: перевірку попадання точки ( яка буде вводитись вручну із клавіатури) в заштриховану область



* 1. **ТЕОРЕТИЧНІ ВІДОМОСТІ**

2.1 При розв’язуванні наступної задачі були використані наступні засоби:

* Використання логічного оператора if

* 1. **СХЕМИ АЛГОРИТМІВ ОСНОВНИХ**
  2. **ФУНКЦІЙ**

Список підключених бібліотек:

**#include <stdio.h>**

**#include <stdlib.h>**

* 1. **Висновок**

1.1

В результаті написання даної програми, ми отримали консульну програму, працюючого характеру. За якою, користувач матиме змогу ввести дані з клавіатури, щоб виконати перевірку попадання точки в дану заштриховану область.

1.2

При роботі з даною програмою, було розроблено зручний інтерфейс для користувача, та змогу перевірити справність програми. Було використано чимало джерел інформації, а також використовування логічних операторів, такі як if і т.д.

* 1. **Додаток(а)**

#include <stdio.h>

#include <stdlib.h>

int main()

{

float x,y;

printf("vvedit x y");

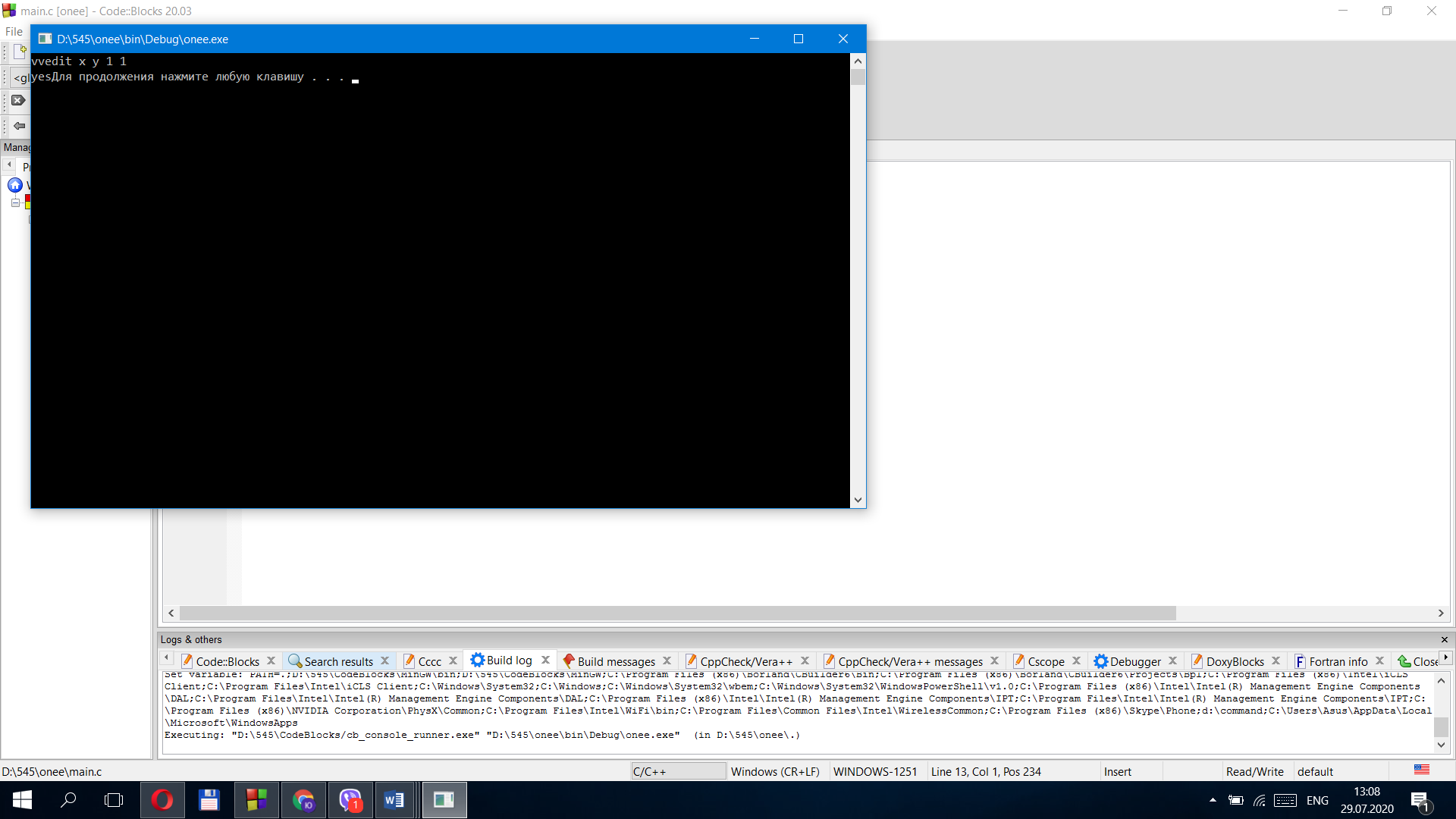
scanf("%f", &x); scanf("%f", &y);

if((x\*x+y\*y<4)&&(x\*x+y\*y>1)){printf("yes");} else {printf("no");}

system("pause");

}

* 1. **Додаток(б)**



1. **Завдання 2**
   1. **ПОСТАНОВКА ЗАДАЧІ**
      1. **Опис завдання**

Варіант-5

1. Обчислити та вивести середнє арифметичне елементiв розмiщених на непарних мiсцях в масивi R(9).

2. Використовуючи задані масиви Т(6) та Н(6), створити масив Р(6), елементи якого обчислюються за формулою Р(і)=Т(і)-Н(і), де і=1,2,…6.

Відсортувати елементи масиву Р за зростанням. Вивести на екран всі масиви.

* 1. **ТЕОРЕТИЧНІ ВІДОМОСТІ**

1. 2.1 При розв’язуванні наступної задачі були використані наступні засоби:

* Визначення середнього арифметичного за існуючою формулою, на непарних місцях масиву
* Виведення нового масиву за допомогою оператора - if
* Робота з одновимірними масивами

2.2 Оголошення одновимірного масиву зі 100 цілих чисел (тип int).

* Масиви організовані у вигляді об’єктів
* Доступ до елементів масиву за допомогою індексів
  1. **СХЕМИ АЛГОРИТМІВ ОСНОВНИХ ФУНКЦІЙ**

1. Список підключених бібліотек:

* #include <time.h>
* #include <stdio.h>
* #include <stdlib.h>
* #include <string.h>

  2. **Висновок**

1. 1.В результаті написання програми, існує зручність та простота організації однотипних наборів даних;
2. зручність обробки даних (сортування, пошук, проведення обчислень) з допомогою операторів циклу мови C#.

2.При опрацюванні наступної задачі, можна зробити висновок, що мова програмування C# строго дотримується допустимих меж масиву, які задаються при його оголошенні. Якщо при доступі до елементів масиву, значення індексу виходить за межі масиву, то виникає виключна ситуація типу IndexOutOfRangeException. У цьому випадку програма передчасно завершиться.

* 1. **Додаток(а)**

1. #include <stdio.h>

#include <stdlib.h>

int main()

{

   int R[9], d=0,i,c=0;

   float seredne;

   printf("Masiv yvestu");

   for (i=0; i<9; i++){

 scanf("%d", &R[i]);

 if ((R[i]%2)!=0){c++; d=d+R[i];}; ( якшо умова викон, то на непарних)

}

   seredne=d/c;

   printf("seredne = %.2f",seredne);

}

2. #include <time.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

/\* run this program using the console pauser or add your own getch, system("pause") or input loop \*/

int main(void) {

 int a[100],b[100],c[100];

 int i,k;

printf("massiv a");

for(i=0;i<6;i++){

 scanf("%d", &a[i]);

}

printf("massiv b");

for(i=0;i<6;i++){

 scanf("%d", &b[i]);

}

printf("massiv c \n");

for(i=0;i<6;i++){

 c[i]=a[i]-b[i];

}

int j;

for(j=0;j>15;j++){

for(i=0;i<5;i++){

 if(c[i]>c[i+1]){k=c[i];c[i]=c[i+1]; c[i+1]=k;};

}}

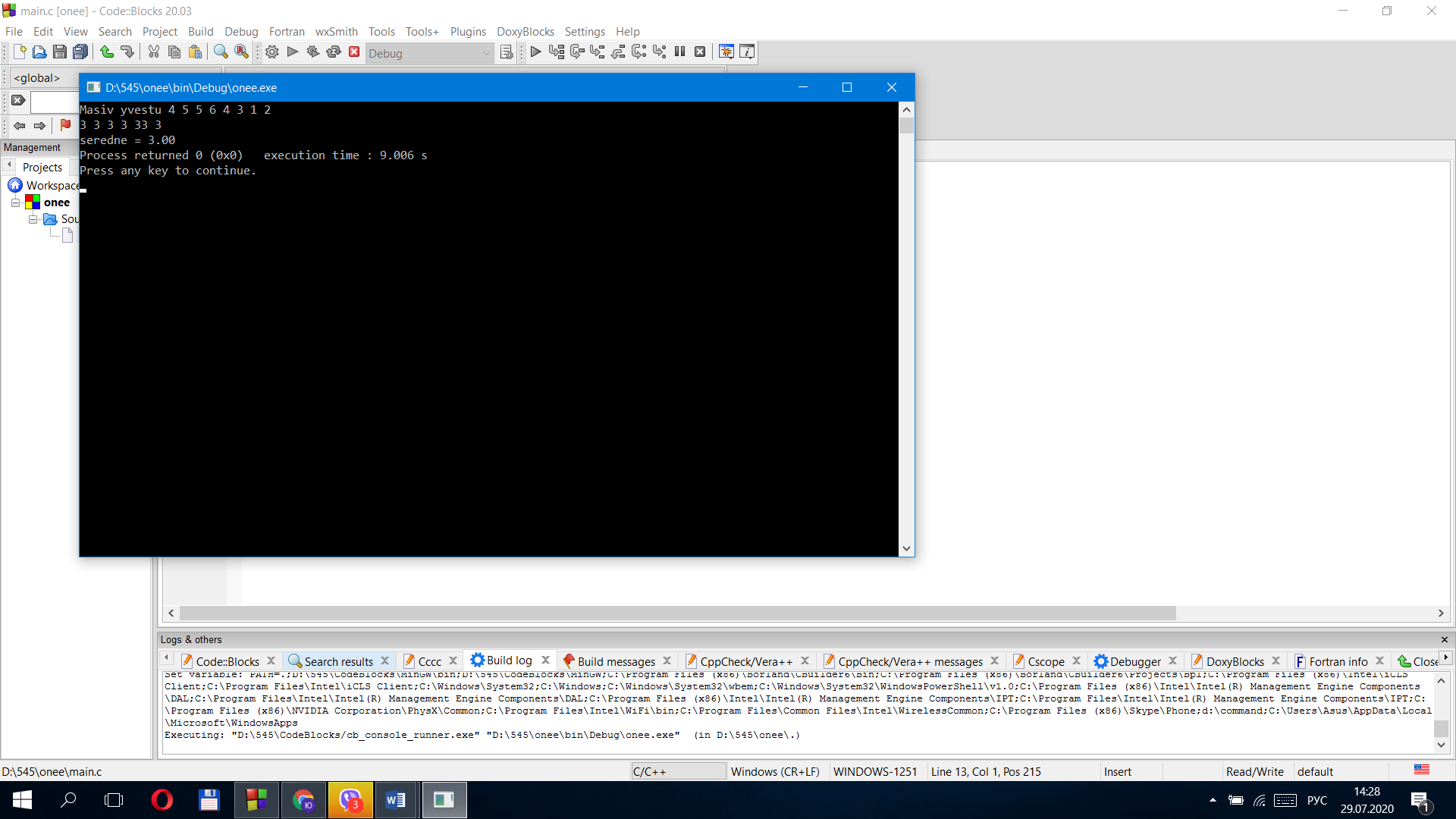
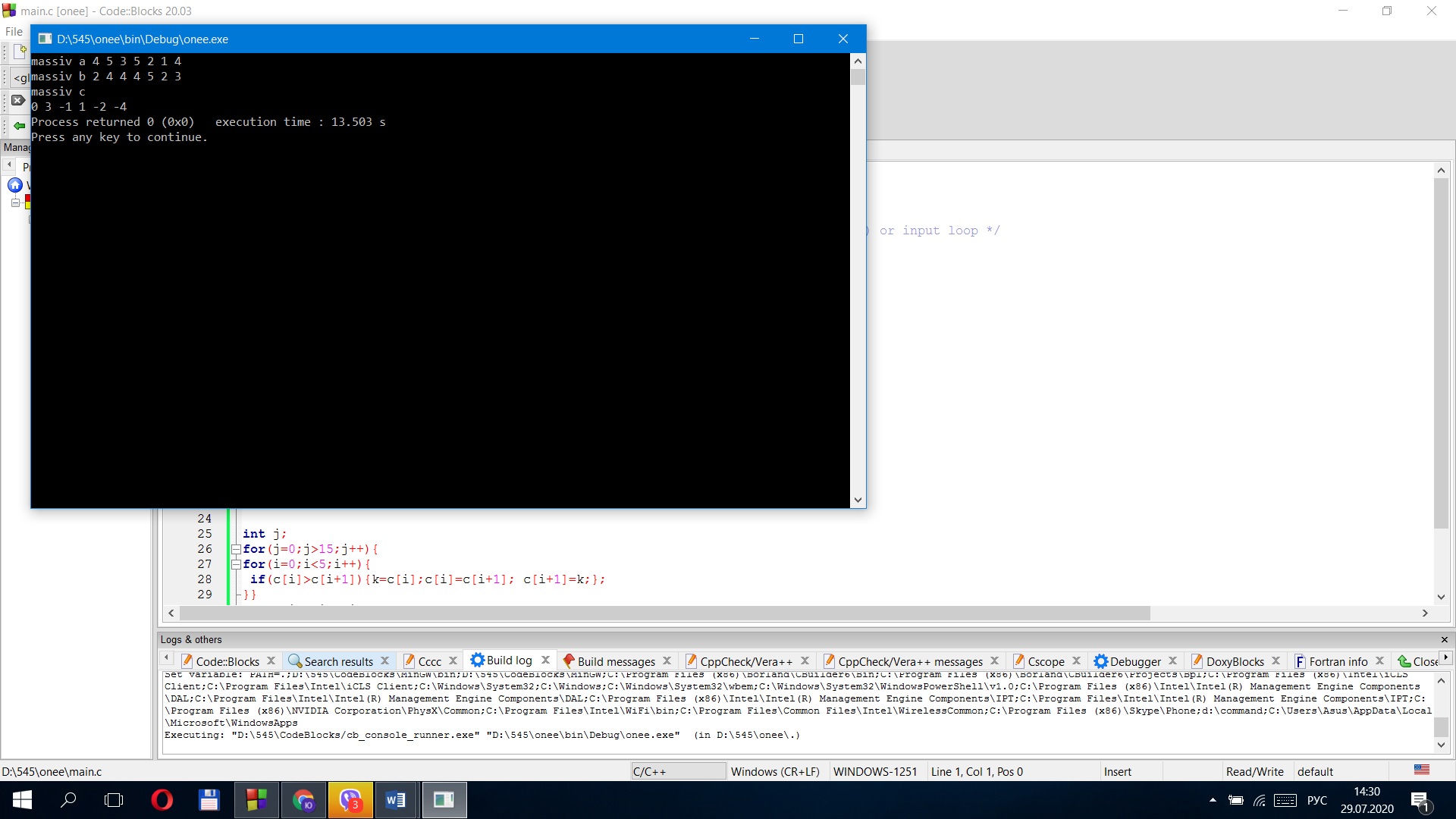
  for(i=0;i<6;i++){

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

}

}

* 1. **Додаток(б)**

1. 
2. 

1. **Завдання 3**
   1. **ПОСТАНОВКА ЗАДАЧІ**
      1. **Опис завдання**

Варіант-5

1.В заданiй матрицi Q(4,5) знайти суму елементiв менших заданого числа D. Вивести вихiдну матрицю та обчислену суму.

2.Задано матрицю В(4,4). Сформувати одновимірний масив з мiнiмальних елементiв стовпчикiв. У цьому масиві обчислити суму та кiлькiсть додатних елементiв. Вивести вихiдну матрицю і сформований масив, суму та кiлькiсть додатних елементiв.

* 1. **ТЕОРЕТИЧНІ ВІДОМОСТІ**

1.1Директива #**define** визначає ідентифікатор і послідовність символів, якій буде заміщатися даний ідентифікатор при його виявленні в тексті програми. Ідентифікатор також називається ім'ям макросу, а процес заміщення називається підстановкою макросу. Стандартний вид директиви наступний:

#define імя\_макроса последовательность\_символів

2.1Звернемо увагу, що в даному операторі відсутня крапка з комою. Міжідентифікатором і послідовністю символів може бути будь-яке число прогалин. Макрос завершується тільки переходом на новий рядок. Також вce #define краще поміщати на початку файлу або взагалі в окремий заголовки.

* 1. **СХЕМИ АЛГОРИТМІВ ОСНОВНИХ ФУНКЦІЙ**

1. Список підключених бібліотек та директив:
2. .#include <stdio.h>
3. #define M 5
4. #define N 4
5. #include <time.h>
6. 1. **Висновок**

3.1 Тобто, двовимірний масив добре підходить для зберігання інформації на шашковому полі. Також двовимірний масив можна легко використовувати для зберігання інформації про будь-який інший грі – шахи, хрестики нулики, сапер і т. д. Щоб отримати доступ до будь-якого елементу такого масиву, потрібно скористатися двома значеннями – індексами, перший індекс – це номер рядка, а другий – номер стовпчика. Все вище сказане відноситься і до n-мірним масивам. Хоча, вже 4-х мірні масиви важкувато візуалізувати.

3.2 Пам'ять для масиву була виділена тільки для десяти елементів, (індекси від 0 до 9), тому елемента з індексом 10 просто не існує. В такому випадку, запис в оперативній пам'яті може призвести до непередбачуваних наслідків – наприклад, ви можете в кінцевому підсумку зіпсувати роботу паралельно запущеної програми.

* 1. **Додаток(а)**

1.#include <stdio.h>

1. #include <time.h>

#define M 5

#define N 4

main() {

    int a[N][M],d, i, j, sum=0;

Printf(“vvedit chislo”);

Scanf(“%d”,d);

    for (i=0; i<N; i++) {

        for (j=0; j<M; j++) {

            scanf("%d", &a[i][j]);

If(a[i][j]<d){

            sum += a[i][j];}

        }

    }

    for (i=0; i<N; i++) {printf("\n");

        for (j=0; j<M; j++){

            printf("%d ", a[i][j]);

       }

    }

Printf(“ suma = %d”, sum);

}

2..#include <stdio.h>

#define m 4

#define n 4

main() {

    int a[n][m],b[100];

    int i, j, s, sum=0, k=0,min=999;

    for (i=0; i<n; i++) {

        for (j=0; j<m; j++) {

            a[i][j] = rand() % 100-20;

            printf("%d ", a[i][j]);

        }

        printf("\n");

    }

    printf("\n new massiv \n");

    for (j=0; j<n; j++) {

   for (i=0; i<m; i++) {

            if(a[i][j]<min){min=a[i][j];b[j]=min;}

        }

        printf("%d ",b[j]);

        min=999;

    }

    for (j=0; j<n; j++) {

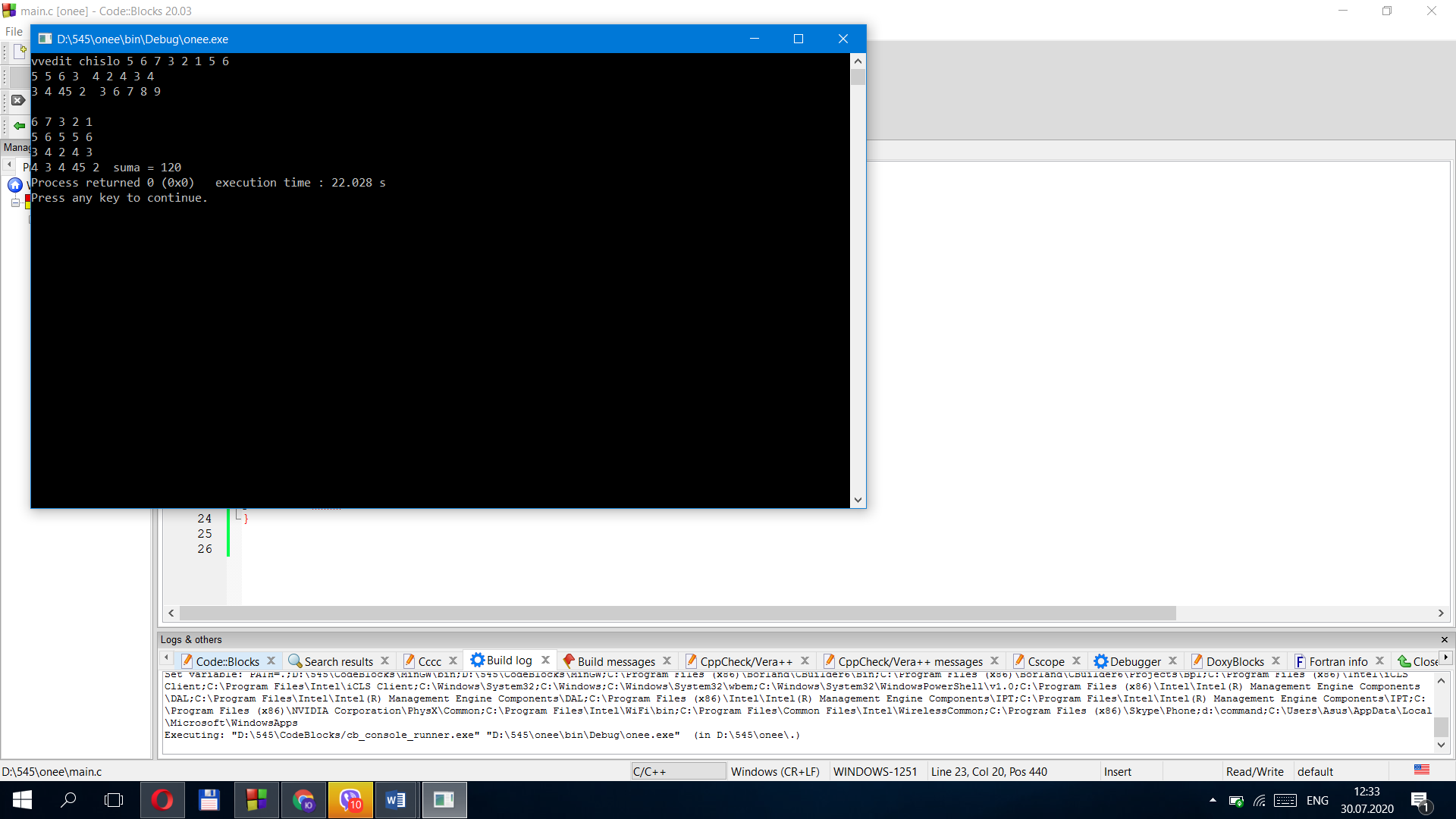
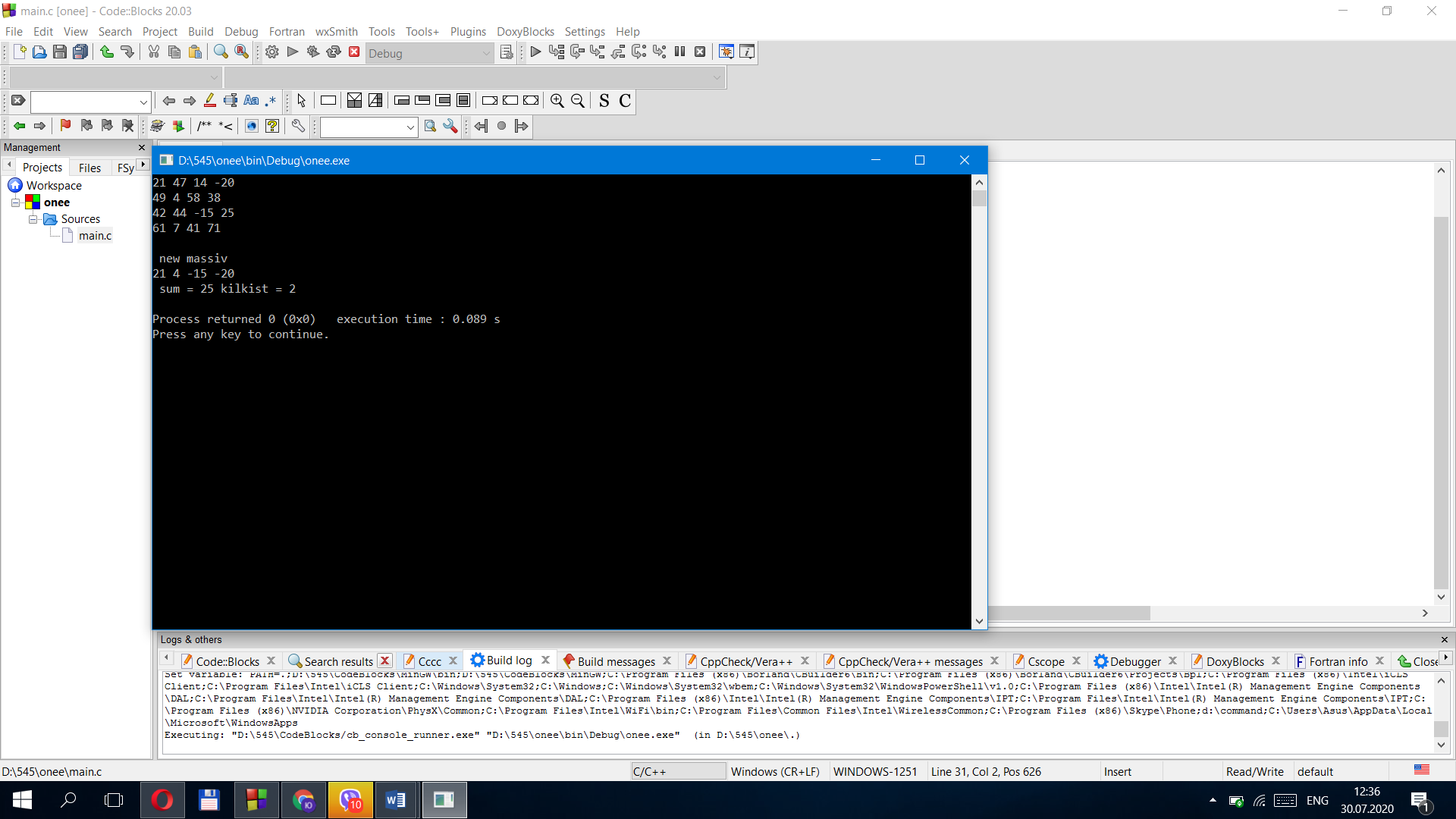
    if(b[j]>0){sum+=b[j]; k++;}

    }

    printf("\n sum = %d kilkist = %d\n", sum,k);

}

* 1. **Додаток(б)**

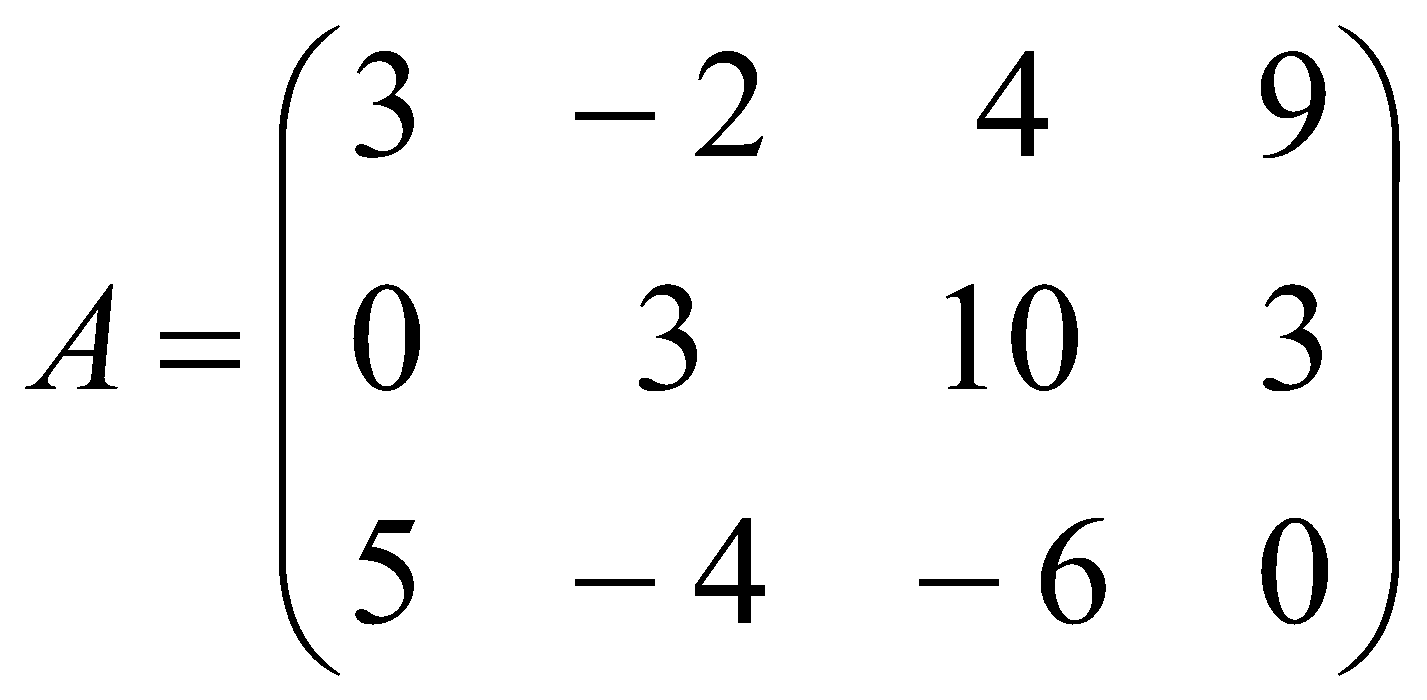
1. 
2. 

1. **Завдання 4**
   1. **ПОСТАНОВКА ЗАДАЧІ**
      1. **Опис завдання**

Варіант-5

1.Задано масив Х(5,5). Всi нульовi елементи головної дiагоналi замiнити сумою додатних елементiв масиву. Вивести вихiдну матрицю, знайдену суму та перетворений масив.

2.Дано дійсну матрицю А розмірності nxm. Заповнити її випадковими цілими числами з відрізка [-10; 10].  Знайти середнє арифметичне максимального та мінімального елементів матриці.

Вхідні дані: n = 3; m = 4; .

* 1. **ТЕОРЕТИЧНІ ВІДОМОСТІ**

1.1Функція srand () використовується для установки початку послідовності, що генерується функцією **rand ()** (функція rand () повертає псевдовипадкові числа).

Функція srand () дозволяє запускати програму кілька разів з різними послідовностями псевдовипадкових чисел.

2.1 Випадкові числа в  мові програмування С#  можуть бути згенеровані функцією  зі стандартної бібліотеки С#. Функція   генерує числа в діапазоні від 0 до…

* 1. **СХЕМИ АЛГОРИТМІВ ОСНОВНИХ**
  2. **ФУНКЦІЙ**

Бібліотеки та директиви, які використовувалися в даних задачах:

**#include <stdio.h>**

**#define N 10**

* 1. **Висновок**

*Суть в тому, що функція  rand() один раз генерує випадкові числа, а при наступних запусках програми всього лише відображає згенеровані перший раз числа. Така особливість функції  rand() потрібна для того, щоб можна було правильно налагодити розроблювану програму. При налагодженні програми, внісши якісь зміни, необхідно упевнитися, що програма спрацьовує правильно, а це можливо, якщо вхідні дані залишилися ті ж, тобто згенеровані числа. Коли програма успішно налагоджена, потрібно, щоб при кожному виконанні програми генерувалися випадкові числа. Для цього потрібно скористатися функцією  srand() зі стандартної бібліотеки С#. Функція  srand() отримавши цілий позитивний аргумент типу  unsigned або unsigned int (без знакове ціле) виконує рандомізацію, таким чином, щоб при кожному запуску програми функція  srand() генерувала випадкові числа. Програма, що використовує функцію  srand() для рандомізації генератора випадкових чисел  rand()*

* 1. **Додаток(а)**

1.#include <stdio.h>

#define N 10

main() {

    int a[N][N], i, j,sum=0;

    srand(time(NULL));

    for (i=0; i<N; i++) {

        for (j=0; j<N; j++) {

            a[i][j] = rand()%100 - 50;

            printf("%d", a[i][j]);

        }

        printf("\n");

    }

    printf("\n");

    for (i=0; i<N; i++){

for(j=0;j<n;j++){

        if (a[i][i] > 0) {

            sum=sum+ a[i][i]

        }}}

   for (i=0; i<N; i++){

for(j=0;j<n;j++){

If((i=j)&&(a[i][j]==0)){a[i][j]=sum;}}}

for (i=0; i<N; i++){printf(“\n”)

For(j=0;j<n;j++){

Printf(“%d ”,a[i][j]);}}

}

2.#include <stdio.h>

#include <stdlib.h>

#include <math.h>

int main()

{

int n,m,j,i, max=-999, min=999;

int a[100][100];

float serarifm;

 printf("vvedit n");

scanf("%d", &n);

printf("vvedit m");

scanf("%d", &m);

  for (j=0; j<n; j++) {

     for (i=0; i<m; i++) {

    a[j][i] = rand() % 20-10;

    printf("%d ", a[j][i]);

          }

          printf("\n");

      }

      for (j=0; j<n; j++) {

     for (i=0; i<m; i++) {

              if(a[j][i]<min){min=a[j][i];}

          }} for (j=0; j<n; j++) {

     for (i=0; i<m; i++) {

              if(a[j][i]>max){max=a[j][i];}}}

  serarifm=((max+min)/2)\*1.0;

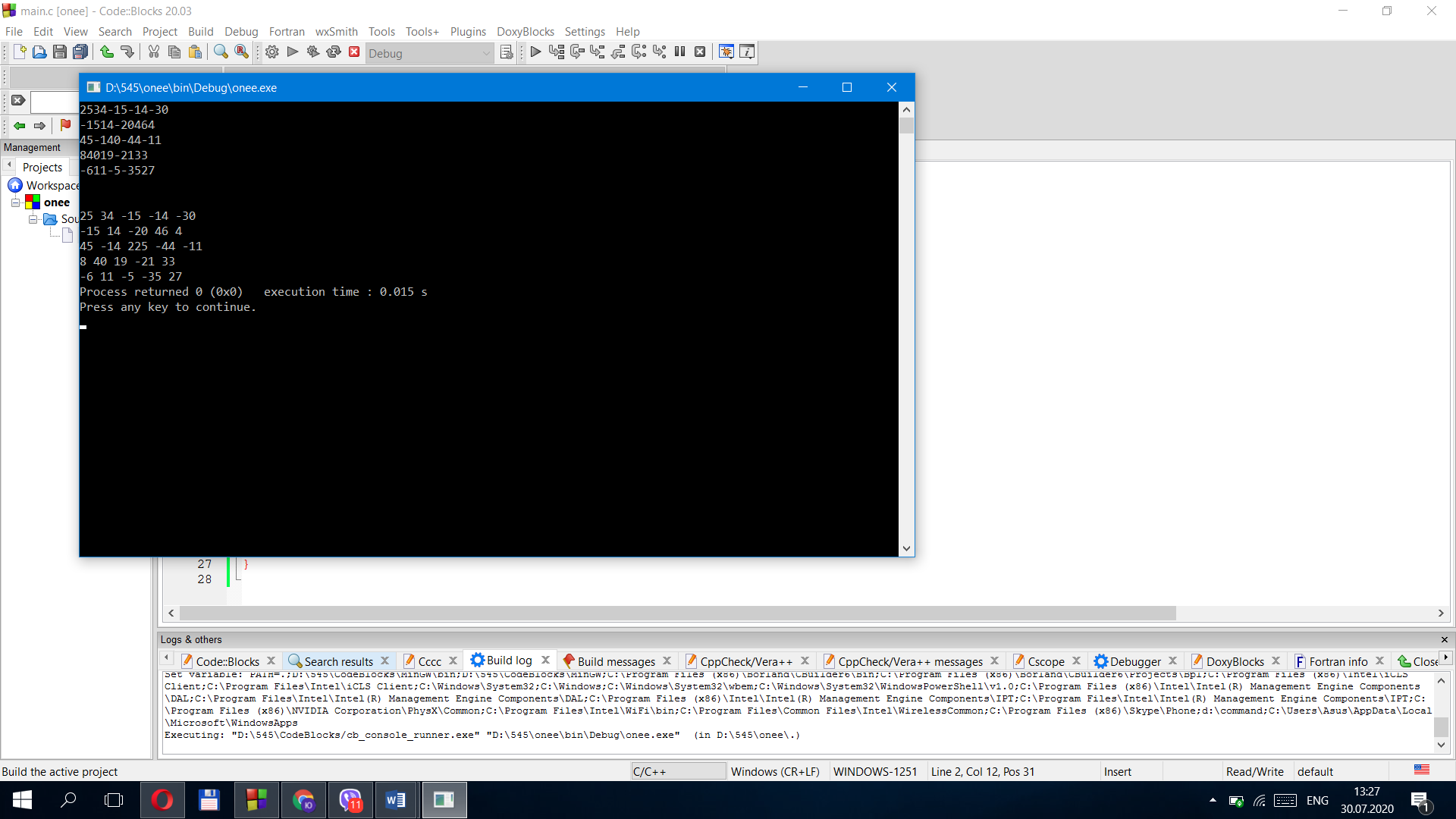
      printf("\n serarifm =%.2f",serarifm);

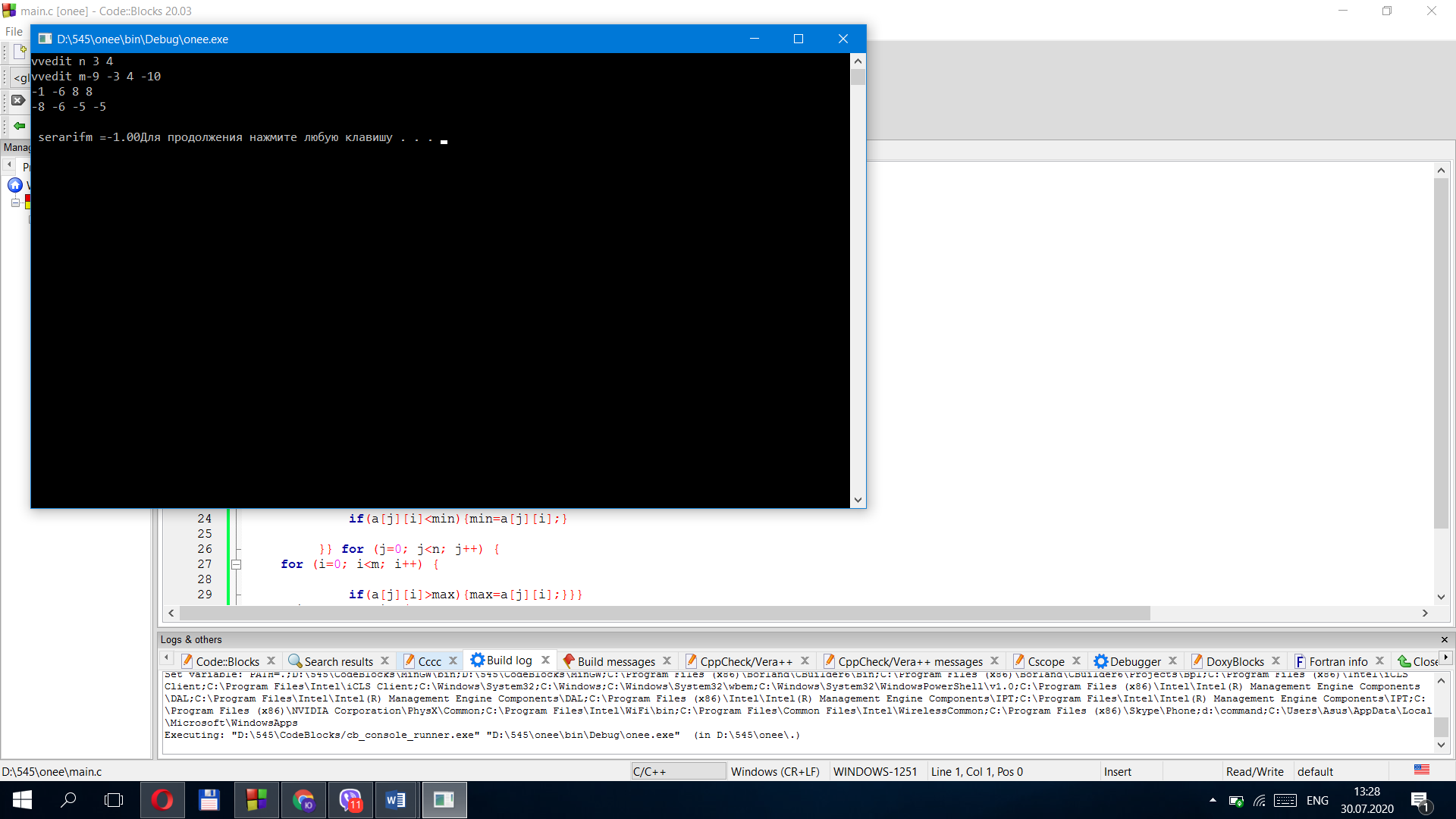
      system("pause");

     }

* 2. **Додаток(б)**

1.



2.

1. **Завдання 5**

**ПОСТАНОВКА ЗАДАЧІ**

* + 1. **Опис завдання**

Варіант-5

1.Дано масив цілих чисел *a*[*i*], *і* = 1, … , *n*. Використовуючи вказівники підрахувати кількість ділянок масиву, де елементи змінюють знак.

*Вхідні дані:* *n* = 5; *a*[*i*] = {3, -7, -12, 13, -7}.

*Вихідні дані:* *Кількість ділянок = 3*

**ТЕОРЕТИЧНІ ВІДОМОСТІ**

* Оголошення **char \* argv []** - це масив (невизначеного розміру) покажчиків на **char** , іншими словами масив рядків. І всі масиви розпадаються на покажчики, і тому ви можете використовувати масив як вказівник (так само, як ви можете використовувати вказівник як масив).

* 1. **СХЕМИ АЛГОРИТМІВ ОСНОВНИХ**
  2. **ФУНКЦІЙ**

Бібліотеки та параметри, які використовувалися в даних задачах:

* #include <stdio.h>
* #include <stdlib.h>
* int main(int argc, char \*argv[])

* 1. **Висновок**
* char \*argv[] - це параметри командного рядка, які можуть бути передані Вашому додатку. Перший аргумент передається з тієї причини, що вбудовані масиви в С# не зберігають розмір і передача розміру - один із способів про це дізнатися (можна наприклад домовитися, що останній елемент буде нульовим, але це накладає певні обмеження).

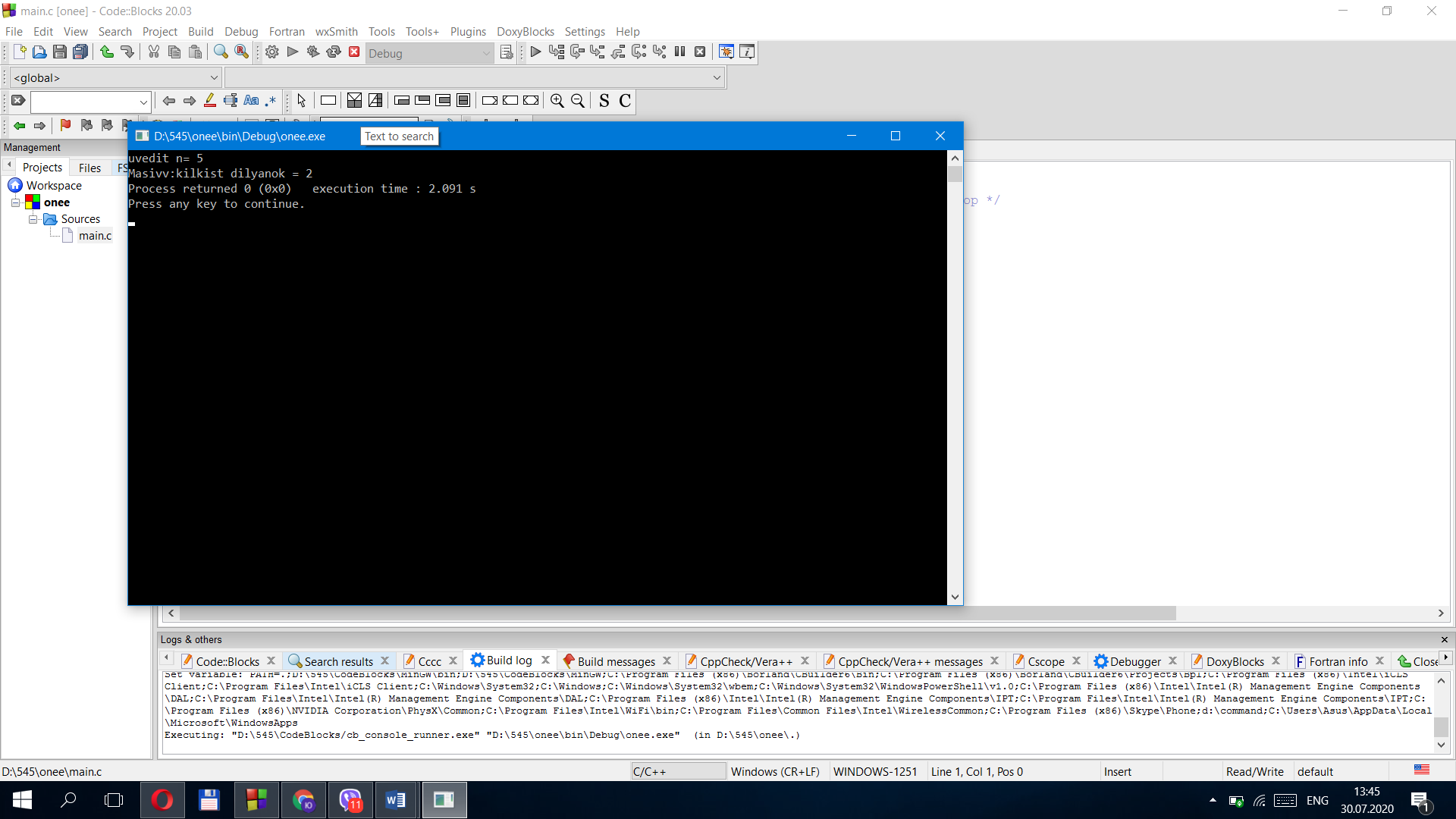
1. пам'ять виділяється якось дивно без **new**
2. пам'ять виділяється в іншому місці, а в функцію приходить тільки покажчик. Як саме вона там виділяється - в даний момент Вас турбувати не повинно. Головне пам'ятати правило - "хто пам'ять виділяє, той зазвичай її і видаляє". В даному випадку компілятор вставить весь необхідний код, який виділить пам'ять і звільнить. Хоча там ще й система трохи може виділити.

* 1. **Додаток(а)**

1.

1. #include <stdio.h>
2. #include <stdlib.h>
3. /\* run this program using the console pauser or add your own getch, system("pause") or input loop \*/
4. int main(int argc, char \*argv[]) {
5. int i,n;
6. int k=0;
7. printf("uvedit n=");
8. scanf("%d",&n);
9. int a[n],\*p;
10. p=a;
11. printf("Masivv:");
12. for(i=0;i<n;i++){
13. \*(a+i)=rand()%20-10;
15. }   for(i=0;i<n;i++){
16. if(i%2==0){
17. if (\*(a+i)<0){k++;}   }}
18. printf("\kilkist dilyanok = %d",k);
19. return 0; }

    2. **Додаток(б)**



1. **Завдання 6**
   1. **ПОСТАНОВКА ЗАДАЧІ**
      1. **Опис завдання**

Варіант-5

Задача 1. Двовимірні масиви. Задача про вибори.

Нехай шість населених пунктів позначені номерами від 1 до 6 (величина *k),* а п’ять кандидатів – номерами від 1 до 5 (величина *п).* Кількість голосів, набраних кандидатами у кожному пункті визначається формулою *akn*=random(10і+50), де *і -* номер варіанта. (Функція random(*п*) описана у модулі stdlib.h. Перед використанням функції random(*п*) треба записати на початку програми функцію randomize() щоб під час виконання програми кожного разу отримувати різні випадкові числа.) Вивести на екран таблицю результатів голосування, де у рядках є дані з населених пунктів, а у стовпцях - дані щодо конкретних кандидатів. Визначити і вивести значення величин з додаткового завдання. Створити одновимірний масив з шуканими даними.

1. Ввести прізвище, ім’я та по батькові як одне дане типу рядок. Визначити довжину рядка і кількість букв «а» у ньому. Виконати додатково завдання свого варіанта двома способами: а) розглядаючи рядок як масив символів;

Вивести ім’я та кількість букв у прізвищі.

* 1. **ТЕОРЕТИЧНІ ВІДОМОСТІ**
* char \*argv[] - це параметри командного рядка, які можуть бути передані Вашому додатку. Перший аргумент передається з тієї причини, що вбудовані масиви в С# не зберігають розмір і передача розміру - один із способів про це дізнатися (можна наприклад домовитися, що останній елемент буде нульовим, але це накладає певні обмеження).

1. пам'ять виділяється якось дивно без **new**

пам'ять виділяється в іншому місці, а в функцію приходить тільки покажчик.

* Функція gets () зчитує символи з stdin і поміщає їх в масив символів, на який вказує str.
  1. **СХЕМИ АЛГОРИТМІВ ОСНОВНИХ ФУНКЦІЙ**

Список використаних бібліотек:

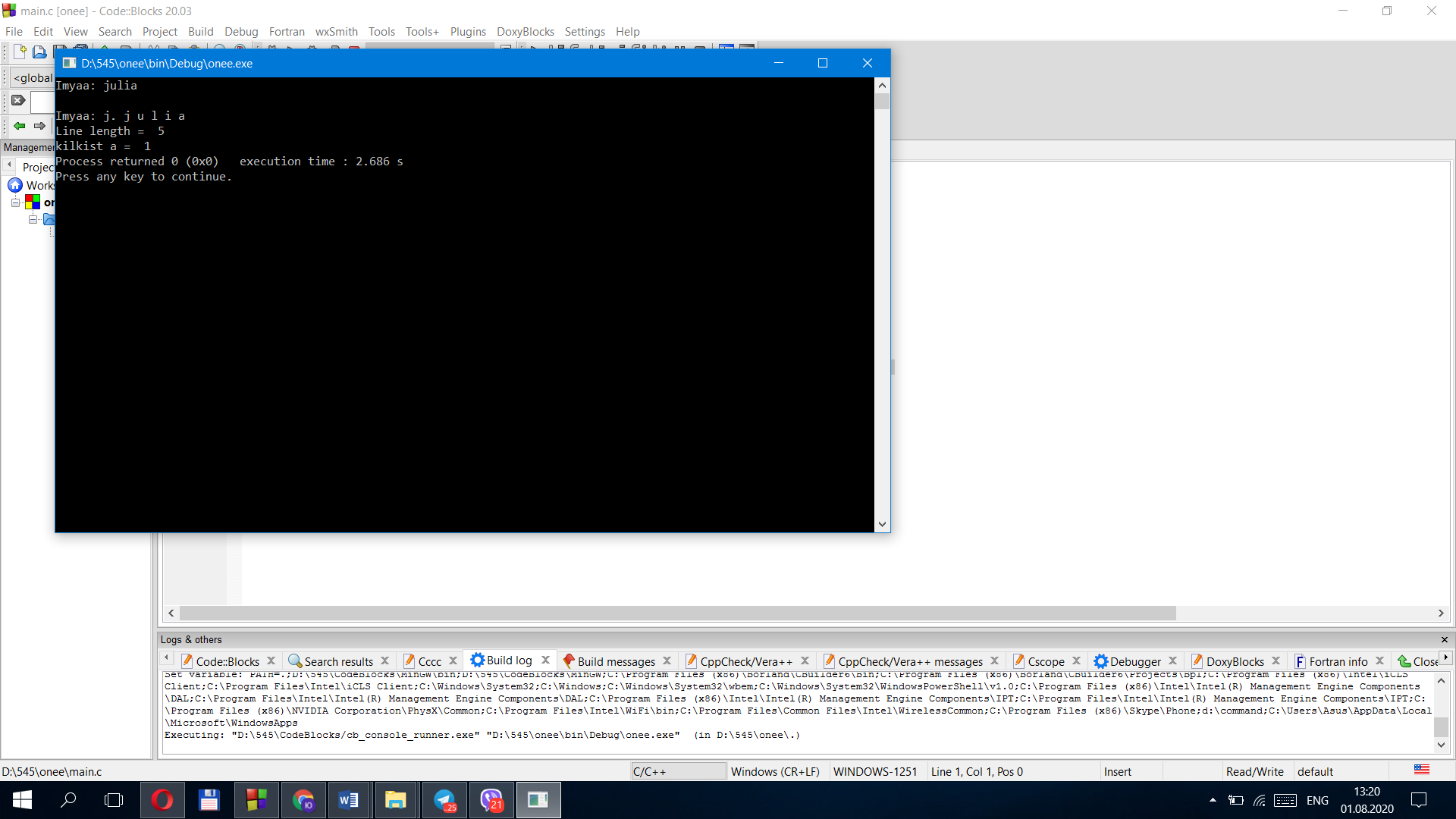
1. #include <stdio.h>
2. #include <stdlib.h>

* 1. **Висновок**

Довжина Сі-рядка визначається по досягненню нульового символу – нуль термінатор. Функція ***strlen*** бачить початок Сі-рядки і починає спочатку рахувати кількість символів (байтів, відводяться під кожен символ), цей процес виконується доти, поки не буде досягнутий завершальний нульовий символ. Зверніть увагу на те, що завершальний нульовий символ не входить в довжину рядка. Він є службовим символом, для позначення завершення Сі-рядки.

* 1. **Додаток(а)**

1. #include <stdio.h>
2. #include <stdlib.h>
3. int main(int argc, char \*argv[]) {
4. char s[100];
5. int i,k=0,n;
6. printf("Imyaa: ");
7. gets(s);
8. n=strlen(s);
9. printf("\nImyaa: %c. ",s[0]);
10. for( i=0; i<strlen(s); i++ ){if( s[i]!="") { printf("%c ",s[i]); } else{break;} }
11. for(i=0; i<strlen(s); i++){
12. if(s[i]=='a') {k++;} else
13. if(s[i]==' ') { printf("%c. ",s[i+1]);}
14. }
15. printf("\nLine length = %d",n);
16. printf("\nkilkist a = %d", k);
17. return 0;
18. }
    1. **Додаток(б)**



1. **Завдання 7**
   1. **ПОСТАНОВКА ЗАДАЧІ**
      1. **Опис завдання**

Варіант-5

Задача 1. Функції користувача.

Утворити і вивести масив y з елементами yk=fi+1(k), де k=1,12, і- номер варіанта. Для розрахунку yk створити функцію користувача. Скласти ще одну функцію користувача для пошуку даних у цьому масиві. Критерії пошуку взяти відповідно свого варіанту. Шукані дані вивести в головній функції.

Скільки було від’ємних значень? Визначити максимальне значення.

Задача 2. . Функції користувача.

Утворити і вивести масив А, елементи якого описані формулою ,  m,n=1…4, i – номер варіанта. Скласти функцію користувача для пошуку даних у цьому масиві. Критерії пошуку взяти відповідно свого варіанту, якщо критерії пошуку не відповідають вашим даним вибрати свої значення для пошуку. Шукані дані вивести в головній функції.

Обчислити суму квадратів елементів, значення яких більші, ніж 1 та кількість таких елементів.

* 1. **ТЕОРЕТИЧНІ ВІДОМОСТІ**

Покажчик типу void (або «загальний покажчик») - це спеціальний тип покажчика, який може вказувати на об'єкти будь-якого типу даних! Оголошується він як звичайний покажчик, тільки замість типу даних використовується ключове слово void

double \*A- посилання у вигляді ярлика

Суть в тому, що функція  rand() один раз генерує випадкові числа, а при наступних запусках програми всього лише відображає згенеровані перший раз числа. Така особливість функції  rand() потрібна для того, щоб можна було правильно налагодити розроблювану програму

* 1. **СХЕМИ АЛГОРИТМІВ ОСНОВНИХ ФУНКЦІЙ**

Список бібліотек:

1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <time.h>

* 1. **Висновок**

Функція - це іменована частина програми, яка може бути багаторазово викликана з іншої ділянки програми (в якому ця функція видна). Функція може приймати фіксоване або змінне число аргументів, а може не мати аргументів. Функція може як повертати значення, так і бути порожньою (void) і нічого не повертати.

Ми вже знайомі з багатьма функціями і знаємо, як їх викликати - це функції бібліотек stdio, stdlib, string, conio тощо. Більш того, main - це теж функція. Вона відрізняється від інших тільки тим, що є точкою входу під час запуску програми.

* 1. **Додаток(а)**

1. 1.
2. #include <stdio.h>
3. #include <stdlib.h>
4. #include <time.h>
5. double \*A(int n) {
6. return (double \*)malloc(n\*sizeof(double));
7. }
8. void am(int \*a, int n) {
9. int i;
10. for(i = 0; i < n; i++){
11. \*(a + i) = rand()%30\*1.12;}}
12. void ap(int \*a, int n) {
13. int i;
14. for (i = 0; i < n; i++){
15. printf("%d ", \*(a + i));}}
16. void m(int \*a, int n) {
17. int i, j,ch,max=-999;
18. int k=0;
19. for (i = 0; i < n; i++){ch=\*(a + i);
20. if(ch<0){
21. k++;}
22. if(ch>max){max=ch;}
23. }
24. printf("\nkilkist =%d \n max=%d",k,max);}
25. int main(){
26. int \*a;
27. int n;
28. printf("vvedit n");
29. scanf("%d", &n);
30. a = A(n);
31. am(a, n);
32. ap(a, n);
33. m(a, n);
34. return 0;}

**2.**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

double \*A(int n) {

return (double \*)malloc(n\*sizeof(double));

}

void am(int \*a, int n) {

int i;

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

\*(a + i) = rand()%30\*1.12;}}

void ap(int \*a, int n) {

int i;

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

printf("%d ", \*(a + i));}}

void m(int \*a, int n) {

int i, j,ch;

int d=0,k=0;

for (i = 0; i < n; i++){ch=\*(a + i);

if(ch>1){

k++;d=d+ch\*ch;}}

printf("\nkilkist =%d \n sum=%d",k,d);}

int main(){

int \*a;

int n;

printf("vvedit n");

scanf("%d", &n);

a = A(n);

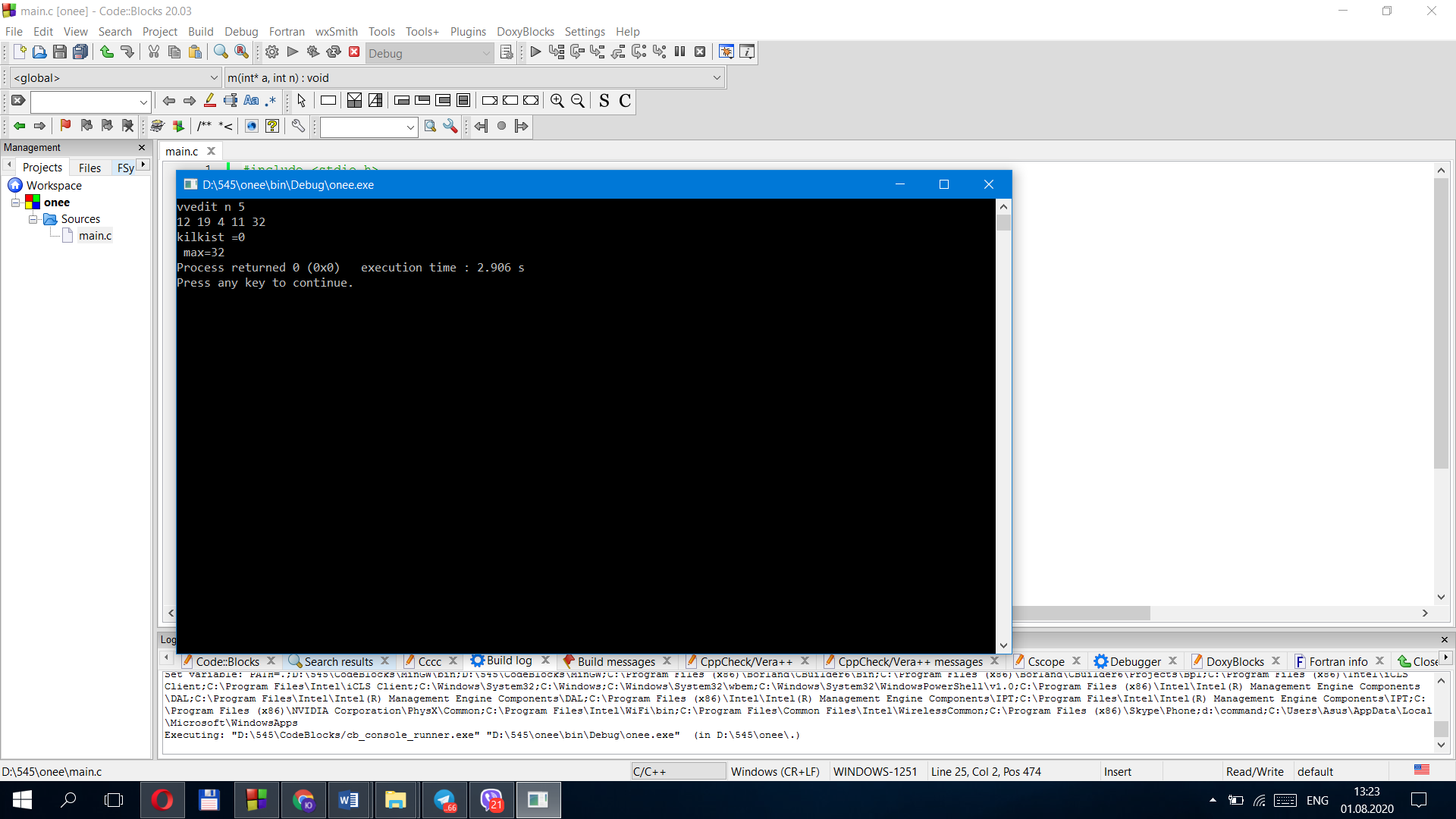
am(a, n);

ap(a, n);

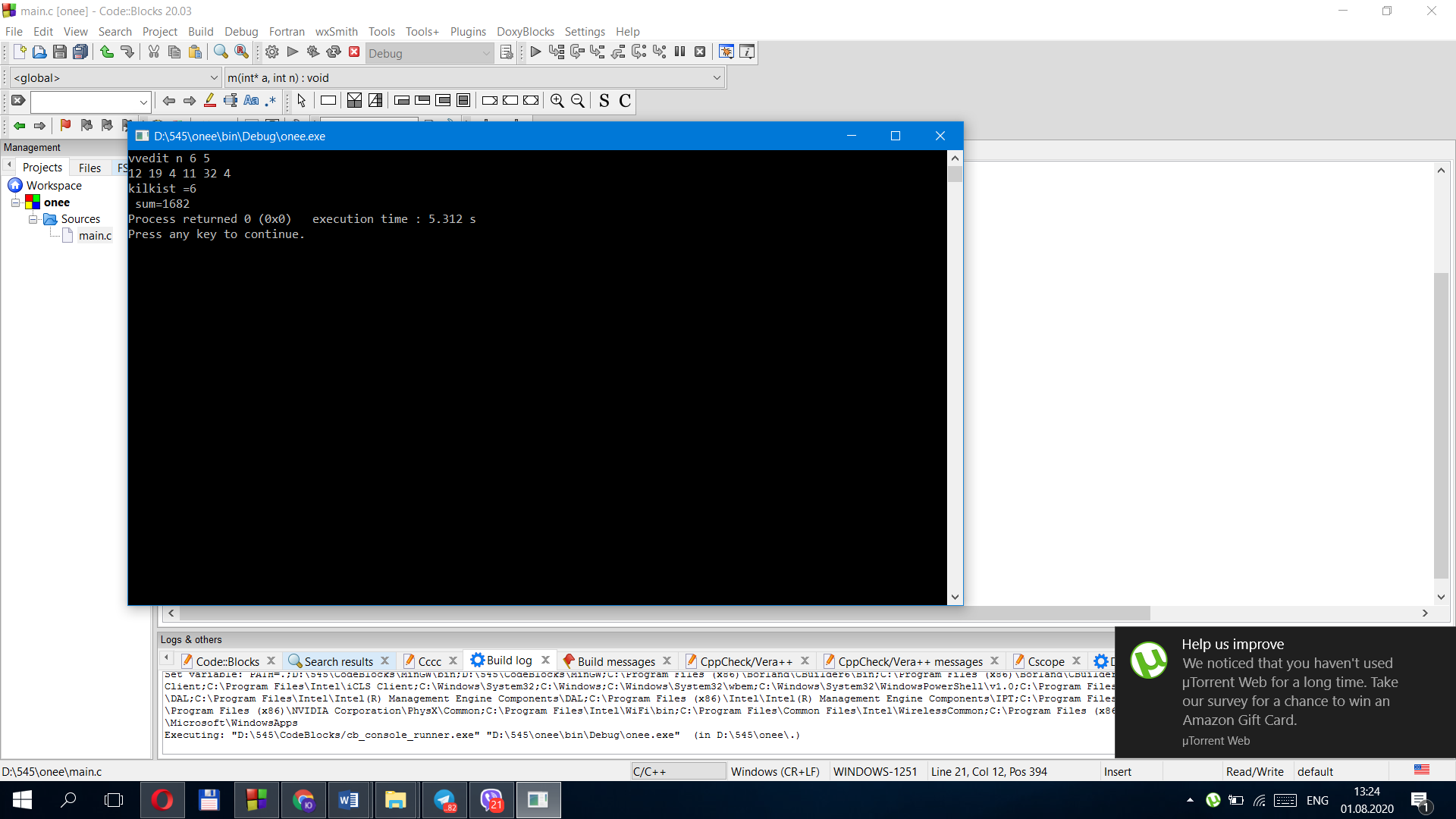
m(a, n);

return 0;}

* 1. **Додаток(б)**

1. 

2.



1. **Завдання 8**
   1. **ПОСТАНОВКА ЗАДАЧІ**
      1. **Опис завдання**

Варіант-5

1. Написати функцію Even логічного типу, що повертає True, якщо цілий параметр k є парним, і False в протилежному випадку. За її допомогою знайти кількість парних чисел серед послідовності з 10 цілих чисел.

Вхідні дані: -2, 13, 3, 22, -6, 32, -4, -5, 19, 16.

Вихідні дані: Кількість парних чисел = 6.

* 1. **ТЕОРЕТИЧНІ ВІДОМОСТІ**

1. double \*A- посилання у вигляді ярлика
2. Функція malloc виділяє блок пам'яті, розміром sizemem байт, і повертає покажчик на початок блоку.
3. Зміст виділеного блоку пам'яті не ініцеліарезується, воно залишається з невизначеними значеннями.

* 1. **СХЕМИ АЛГОРИТМІВ ОСНОВНИХ ФУНКЦІЙ**

Підключені бібліотеки:

1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <time.h>

* 1. **Висновок**

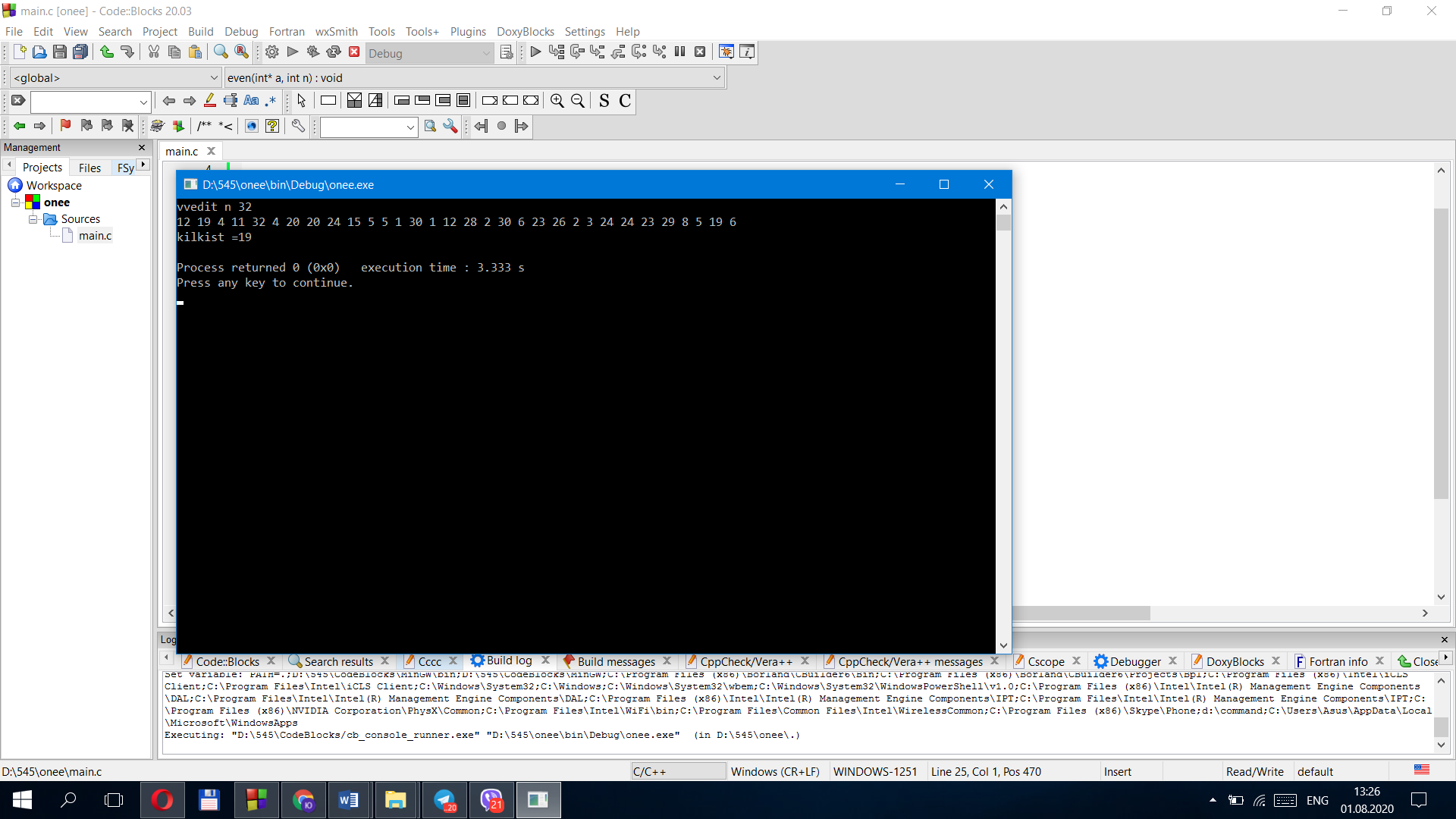
Покажчик на виділений блок пам'яті. Тип даних на який посилається вказівник завжди void\*, тому це тип даних може бути приведений до бажаного типу даних.  
Якщо функції не вдалося виділити необхідний блок пам'яті, повертається нульовий покажчик.

Ця програма створює рядок довжиною вказаний користувачем і заповнює її символами алфавіту. Можлива довжина цього рядка обмежена тільки кількістю вільної пам'яті в системі, яку malloc може виділити.

* 1. **Додаток(а)**

1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <time.h>
4. double \*A(int n) {
5. return (double \*)malloc(n\*sizeof(double));
6. }
7. void am(int \*a, int n) {
8. int i;
9. for(i = 0; i < n; i++){
10. \*(a + i) = rand()%30\*1.12;}}
11. void ap(int \*a, int n) {
12. int i;
13. for (i = 0; i < n; i++){
14. printf("%d ", \*(a + i));}}
15. void even(int \*a, int n) {
16. int i, j,ch;
17. int k=0;
18. for (i = 0; i < n; i++){ch=\*(a + i);
19. if(ch%2==0){
20. k++;}
21. }
22. printf("\nkilkist =%d \n ",k);}
23. int main(){
24. int \*a;
25. int n;
26. printf("vvedit n");
27. scanf("%d", &n);
28. a = A(n);
29. am(a, n);
30. ap(a, n);
31. even(a, n);
32. return 0;}

* 1. **Додаток(б)**



1. **Завдання 9**
   1. **ПОСТАНОВКА ЗАДАЧІ**
      1. **Опис завдання**

Варіант-5

1. Дано масив дійсних чисел *a*[*i*], *і* = 1, … , *n*. Виділіть для нього необхідний обсяг динамічної пам'яті. Створити динамічний масив *b*[*j*], *j* = 1, … , *m,* в який записати лише парні елементи масиву *a*[*i*], що знаходяться на непарних місцях. Вивільніть динамічну пам'ять.

*Вхідні дані:* *n* = 6, *a*[*i*] = {31, -6, 2, 0, 33, 6}.

*Вихідні дані:* *b*[*j*] = {-6, 0, 6}.

* 1. **ТЕОРЕТИЧНІ ВІДОМОСТІ**
  2. **СХЕМИ АЛГОРИТМІВ ОСНОВНИХ ФУНКЦІЙ**

Підключені бібліотеки:

1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <time.h>

* 1. **Висновок**

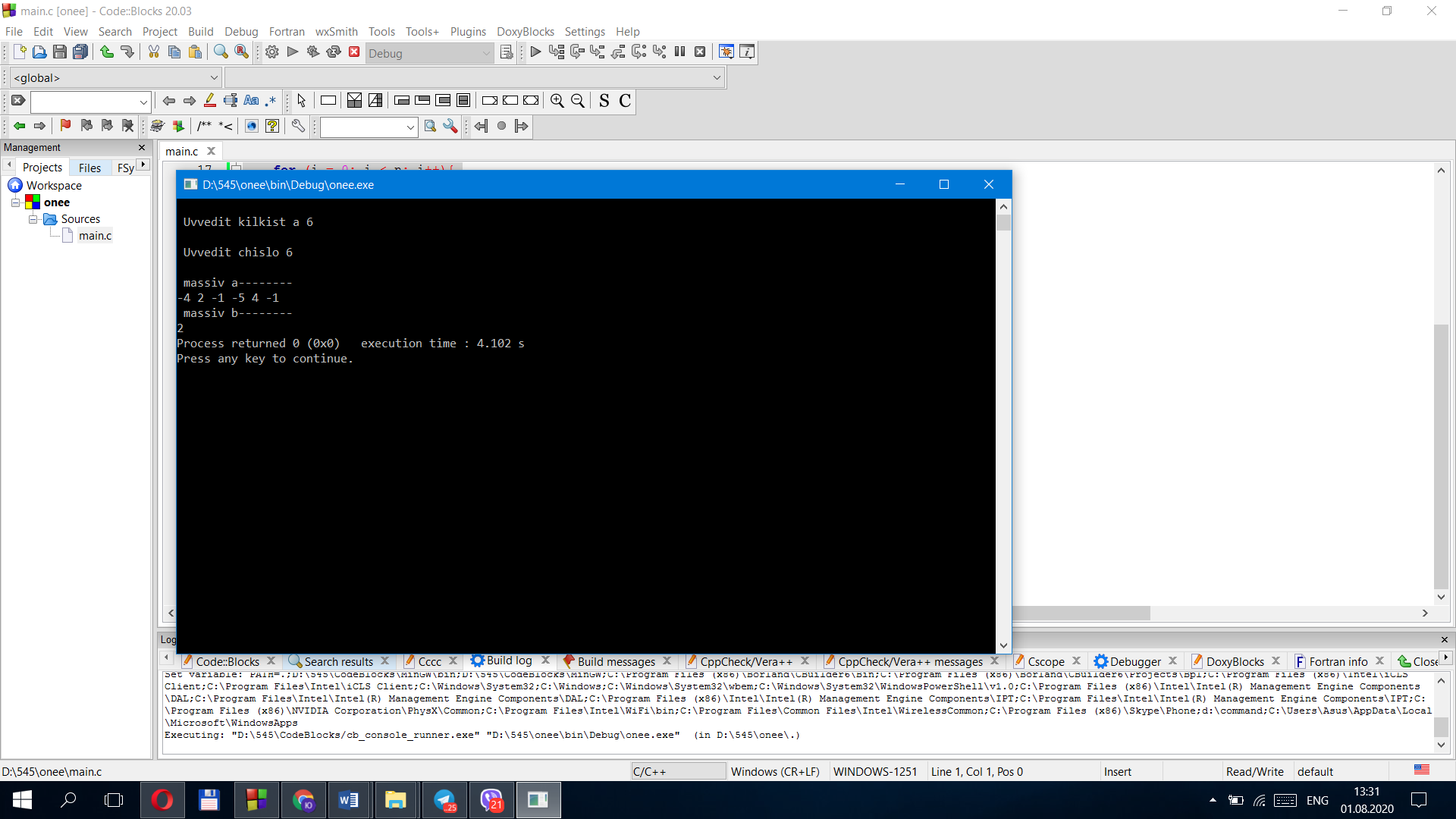
Функція free () повертає пам'ять, на яку вказує параметр ptr, тому в купу. В результаті ця пам'ять може виділятися знову.

Обов'язковою умовою використання функції free () є те, що звільняється пам'ять повинна була бути попередньо виділена з використанням однієї з наступних функцій: malloc () або calloc (). Використання неправильного покажчика при виклику цієї функції зазвичай веде до руйнування механізму управління пам'яттю і краху системи.

* 1. **Додаток(а)**

1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <time.h>
4. double \*A(int n) {
5. return (double \*)malloc(n\*n\*sizeof(double));}
6. double \*B(int m) {
7. return (double \*)malloc(m\*m\*sizeof(double));}
8. void am(int \*a, int n) {
9. int i;
10. for(i = 0; i < n; i++){
11. a[i] = rand()%10-5;}}
12. void ap(int \*a, int n) {
13. int i;
14. for (i = 0; i < n; i++){
15. printf("%d ", a[i]);}}
16. void s(int \*a, int \*b,int n) {
17. int i,j=0;
18. for (i = 0; i < n; i++){
19. if(i%2!=0){if (a[i]%2==0){b[j]=a[i]; printf("%d ", b[j]); j++;}}}}
20. int main(){
21. int \*a,\*b;
22. int n;
23. printf("\n Uvvedit kilkist a ");
24. scanf("%d", &n);
25. printf("\n Uvvedit chislo ");
26. scanf("%d", &n);
27. a = A(n);
28. b = B (n);
29. printf("\n massiv a-------- \n");
30. am(a, n);
31. ap(a, n);
32. printf("\n massiv b-------- \n");
33. s(a,b,n); free(a); free(b);
34. return 0;}

* 1. **Додаток(б)**



1. **Завдання 10**
   1. **ПОСТАНОВКА ЗАДАЧІ**
      1. **Опис завдання**

Варіант-5

1. У шкільній адміністрації є така інформація про кожного випускника 11 класів:

– прізвище, ім'я, по батькові;

– дата народження (рік, місяць, число);

– домашня адреса (вулиця, будинок, квартира);

– оцінки атестата про середню освіту.

Визначити: а) чи є в цьому списку випускник із вказаним прізвищем та ім’ям; б) кількість випускників, які старші на даний момент за 18 років

* 1. **ТЕОРЕТИЧНІ ВІДОМОСТІ**

Між struct і typedef struct відмінність є, і воно тонке. Справа в тому, що struct визначає новий тип. На відміну від цього typedef struct ніякого типу не визначає.

- infinity -1. # INF випадкові цифри

Gets - функція, що входить в Стандартну бібліотеку мови Сі, оголошується в заголовки stdio.h, яка зчитує рядок стандартного вводу і поміщає її в буфер, створений викликає функцією.

* 1. **СХЕМИ АЛГОРИТМІВ ОСНОВНИХ ФУНКЦІЙ**

Підключені бібліотеки:

1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <time.h>

* 1. **Висновок**

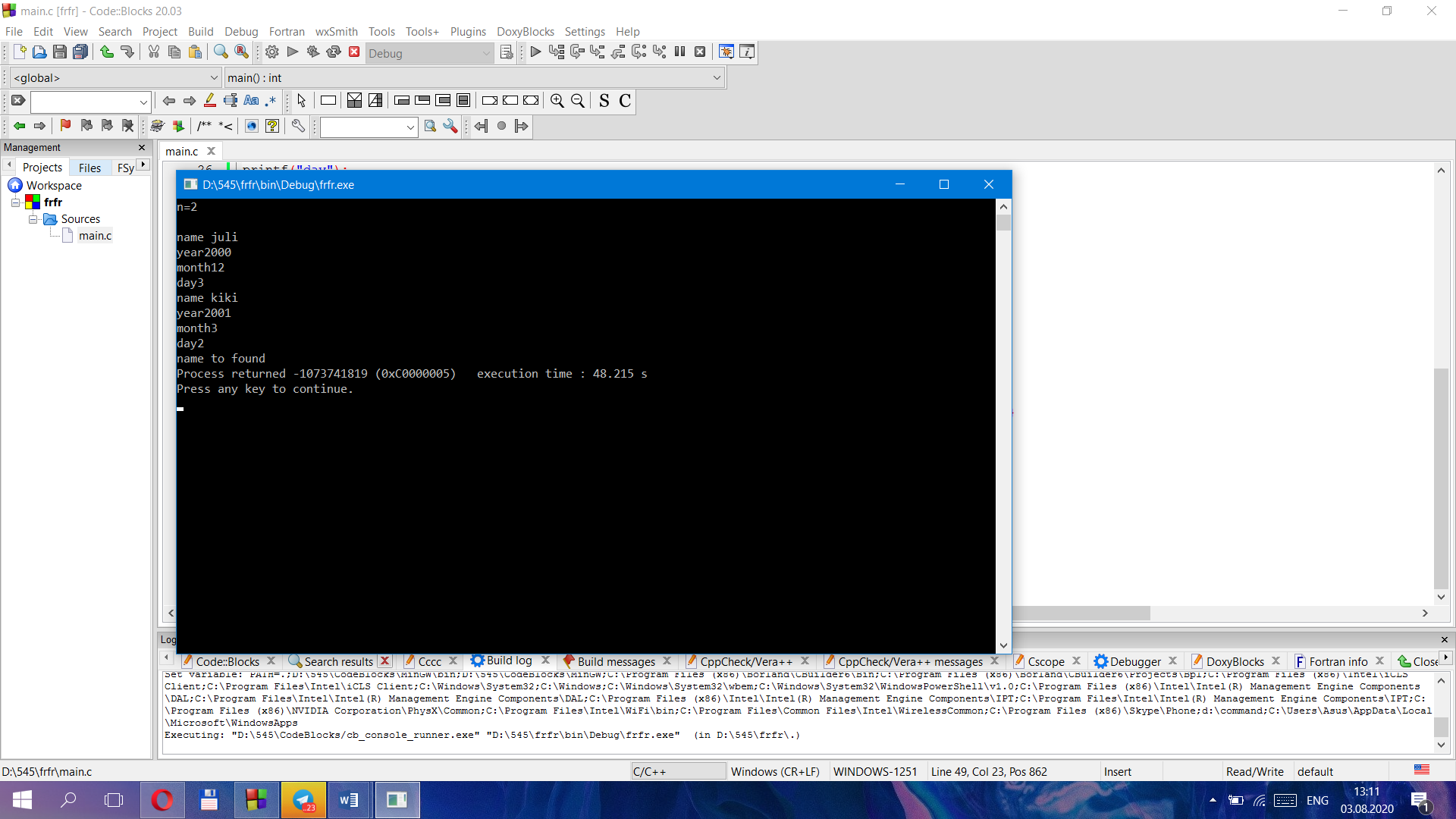
sizeof не може використовуватися з типами функцій, неповними типами або значеннями бітового поля. При застосуванні до еталонного типу результатом є розмір посилального типу.

Якщо застосовано до типу класу, результатом є розмір об'єкта цього класу плюс будь-яка додаткова підкладка, необхідна для розміщення такого об’єкта в масиві. Результат sizeof завжди є ненульовим, навіть якщо він застосовується до порожнього типу класу.

* 1. **Додаток(а)**

1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <time.h>
4. typedef struct {
5. char nm[64];
6. int d, m,y;
7. }g;
8. int main(){char c[100];
9. int i,n,k=0,l=0;
10. g \*inf;
11. printf("n=");
12. scanf("%d",&n);
13. inf = (g \*) calloc (n,sizeof(g));
14. if(inf == 0){exit(1);}
15. printf("\n");
16. for (i=0; i<n; i++){
17. printf("name ");
18. fflush(stdin);
19. gets(inf[i].nm);
20. printf("year");
21. scanf("%d",&inf[i].y);
22. printf("month");
23. scanf("%d",&inf[i].m);
24. printf("day");
25. scanf("%d",&inf[i].d);
26. }
27. printf("name to found ");
28. gets(c);
29. for (i=0; i<n;i++){
30. if (strcmp(inf[i].m, c)==0){i++;}
31. if(inf[i].y<2001){k++;}else {
32. if((inf[i].y==2002)&&(inf[i].m<8)){k++;} else {if((inf[i].y==2002)&&(inf[i].m==8)&&(inf[i].d<3)){k++;}
33. }
34. }
35. }
36. printf("\nkilkist 18+=%d",k);
37. if(l==0){printf("name not found");}else{printf("name found");}
38. system("pause");
39. return 0;
40. }

* 1. **Додаток(б)**



1. **Завдання 11**
   1. **ПОСТАНОВКА ЗАДАЧІ**
      1. **Опис завдання**

Варіант-5

1. Відомі дані про ріст 15 юнаків класу. Впорядкувати дані за спаданням. Жодна пара учнів не має однаковий зріст. На початку навчального року в клас з’явився новий учень (відомо, що його ріст не збігається з ростом жодного з учнів класу, не перевищує ріст найнижчого учня та менший росту найвищого).
   1. Вивести прізвища всіх учнів, ріст яких менше росту новенького.
   2. Визначити прізвище учня, після якого варто записати прізвище новенького, щоб упорядкованість не порушилася.
   3. Визначити прізвище учня, ріст якого найменше відрізняється від росту новенького.
   4. Одержати новий список прізвищ учнів (з урахуванням прізвища новенького), у якому прізвища розташовані в порядку спадання росту.

ПРОСТІ ЗАДАЧІ

1. Відомі дані про вартість кожного з 20 найменувань товарів: число гривень та число копійок. Скласти програму, що порівнює вартість двох будь-яких найменувань товарів (визначальну, який з товарів коштує дорожче).

* 1. **ТЕОРЕТИЧНІ ВІДОМОСТІ**

1. Між struct і typedef struct відмінність є, і воно тонке. Справа в тому, що struct визначає новий тип. На відміну від цього typedef struct ніякого типу не визначає.
2. Gets - функція, що входить в Стандартну бібліотеку мови Сі, оголошується в заголовки stdio.h, яка зчитує рядок стандартного вводу і поміщає її в буфер, створений викликає функцією.
3. 1. **СХЕМИ АЛГОРИТМІВ ОСНОВНИХ ФУНКЦІЙ**

Список підключених бібліотек:

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

typedef struct {

* 1. **Висновок**

Якщо потік пов'язаний з файлом, відкритим для запису, то випустіть fflush (), приводячи до фізичних записів, що містяться у файлі. Якщо ж потік вказує на введений файл, то очищається вхідний буфер. У обох випадках файл залишається відкритим.

Возвращает 0 значить успіх, а возвращает ненулевой величини вказує на наявні помилки за записами. Очистка всіх буферів виробляється автоматично при нормальних завершених програмах або при заповненні буферів. Закрийте файл також приводячи в очищувальні буфери.

* 1. **Додаток(а)**

1.

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

typedef struct {

char c[100];

int h;

}g;

void p(g infa){

printf("last name %s \t height %d \n", infa.c,infa.h);};

int main(){

char cc[100],cn[100];

int hh,hn, v=300;

int i,n,j=0,t,x;

g \*inf;

printf("n=");

scanf("%d",&n);

inf = (g \*) calloc (n,sizeof(g));

if(inf == 0){exit(1);}

printf("\n");

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

printf("last name ");

fflush(stdin);

gets(inf[i].c);

printf("height ");

scanf("%d",&inf[i].h);}

printf("last name new one ");

fflush(stdin);

gets(cn);

printf("height ");

scanf("%d",&hn);

printf("massiv------ \n");

while(v>0){

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

t=inf[i].h;

if(t==v){

p(inf[i]); v--;

}

}v--;}

/\* 1 \*/

printf("\n menshe \n");

for (i=0; i<n;i++){t=inf[i].h;

if(t<hn){p(inf[i]);x=i;}}

/\* 1 \*/

/\* 2 \*/

int max=-999;

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

if((inf[i].h<hn)&&(inf[i].h>max)){

max=inf[i].h; x=i;

}

}

printf("\n new after = ");p(inf[x]);

/\* 2 \*/

/\* 3 \*/

int u,min=999;

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

if((inf[i].h>hn)&&(inf[i].h<min)){

min=inf[i].h; u=i;

}

}

max=hn-max;

min=min-hn;

if(max>min){printf("\n naymenshe = ");p(inf[u]);}else{printf("\n naymenshe = ");p(inf[x]);}

/\* 3 \*/

/\* 4 \*/

printf("new------ \n");

v=300;

while(v>0){

if(v==hn){

printf("last name %s \t height %d \n", cn,hn);

}

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

t=inf[i].h;

if(t==v){

p(inf[i]);

}

}v--;}

system("pause");

return 0;

}

2.

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

typedef struct {

char c[100];

int h,k;

}g;

void p(g infa){

printf("name %s \t grn %d kopiyok %d \n", infa.c,infa.h, infa.k);};

int main(){

int i,n,a,b,hh,kk;

g \*inf;

printf("n=");

scanf("%d",&n);

printf("vvedit nomer a");

scanf("%d",&a);

printf("vvedit nomer b");

scanf("%d",&b);

inf = (g \*) calloc (n,sizeof(g));

if(inf == 0){exit(1);}

printf("\n");

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

printf("name");

fflush(stdin);

gets(inf[i].c);

printf("grn");

scanf("%d",&inf[i].h);

printf("kopiyok");

scanf("%d",&inf[i].k);

}

hh=inf[a].h\*100+inf[a].k;

kk=inf[b].h\*100+inf[b].k;

If(hh>kk{printf(“\n bilshe” ); p(inf[a]); printf(“\n menshe” ); p(inf[b]);

} else {printf(“\n menshe” ); p(inf[a]); printf(“\n bilshe” ); p(inf[b]);

}

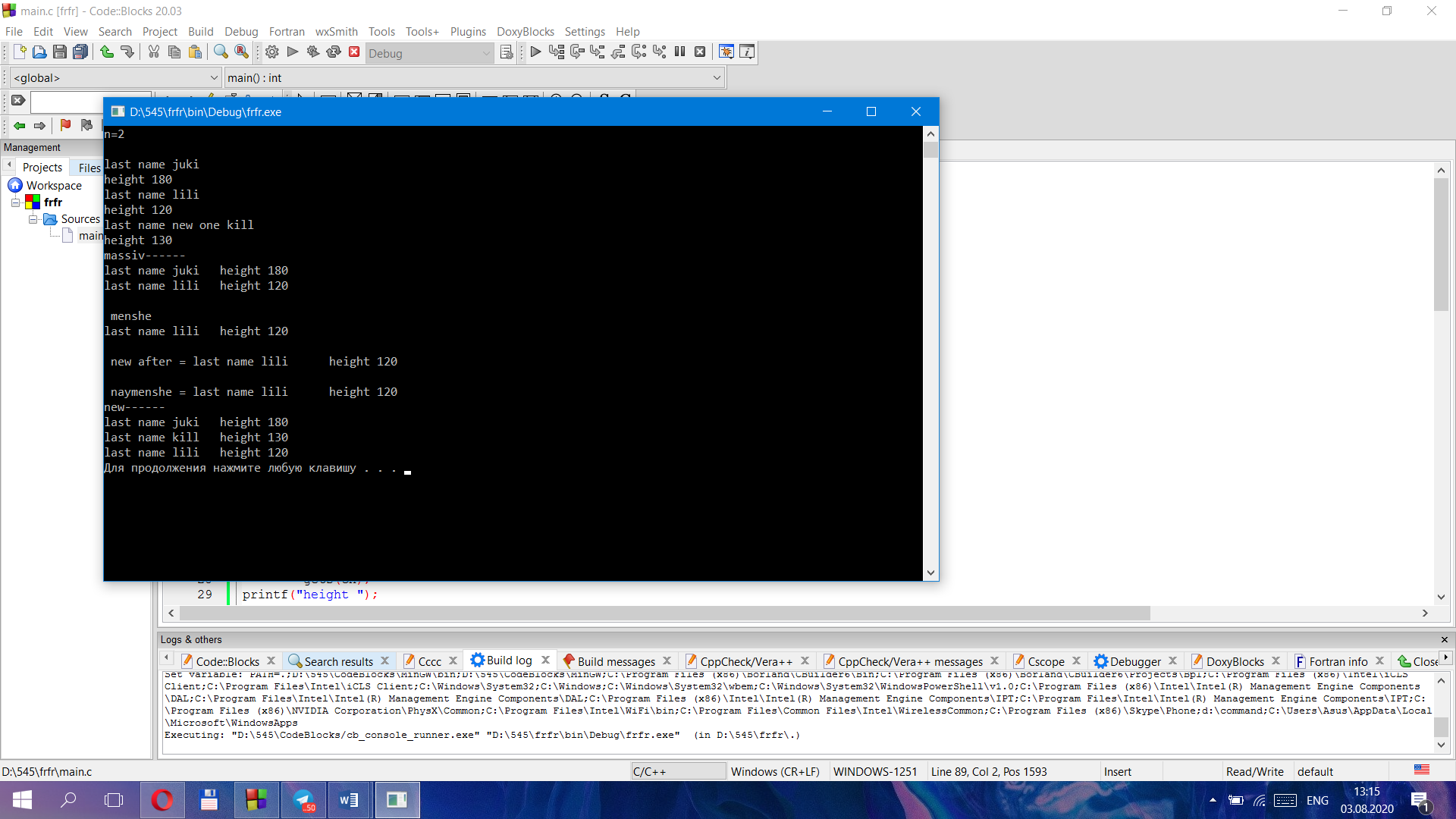
system("pause");

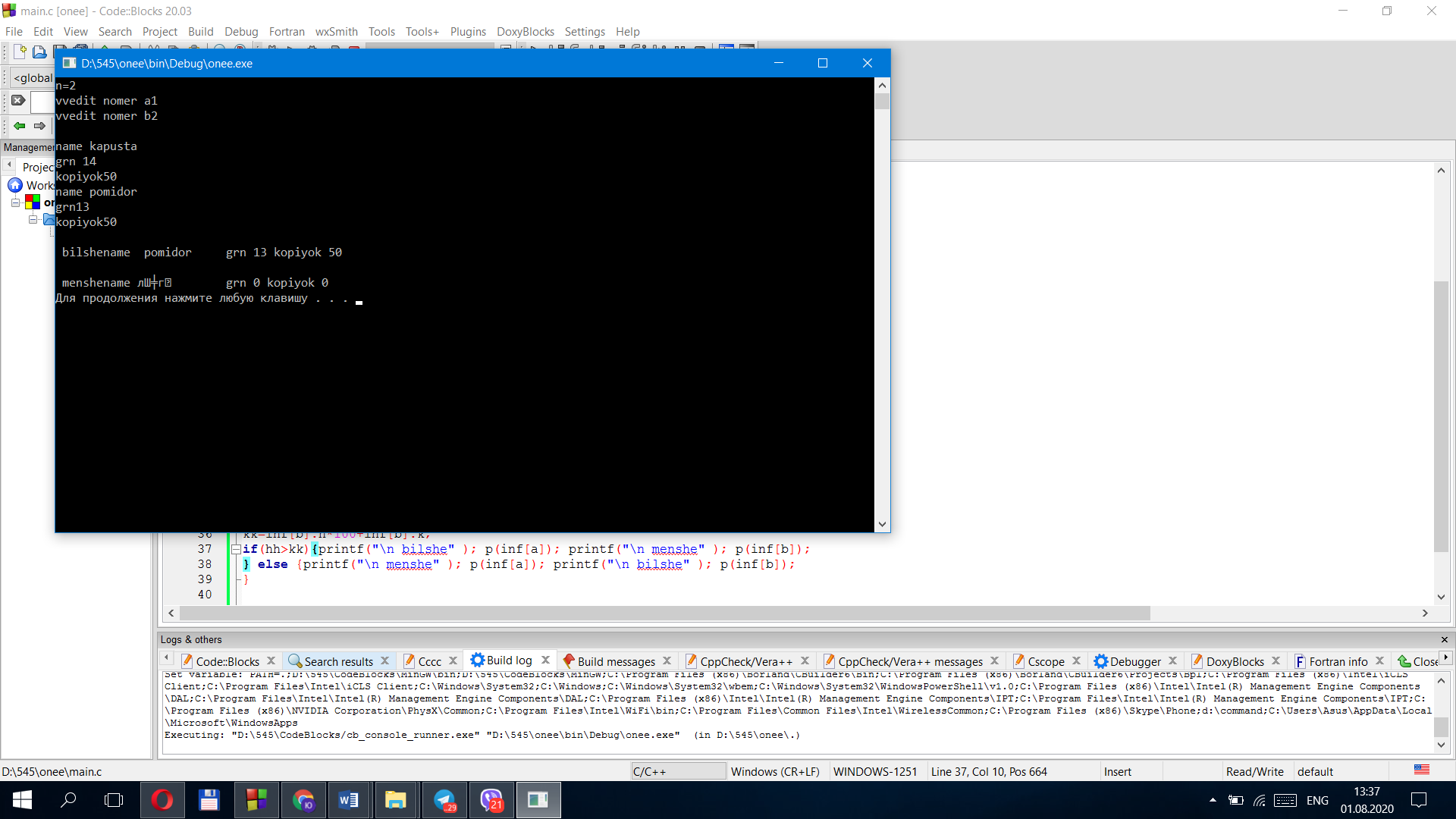
return 0;

}

* 1. **Додаток(б)**

1.



2.

1. **Завдання 12**
   1. **ПОСТАНОВКА ЗАДАЧІ**
      1. **Опис завдання**

Варіант-5

1. Створити однозв’язний лінійний список з 5-ти елементів, які мають одне інформаційне поле цілого типу (як додатні, так і від’ємні значення). На основі створеного списку створити ще 2 списки: в одному повинні міститися додатні елементи початкового списку, в іншому – від’ємні. Вивільнити динамічну пам’ять.

* 1. **ТЕОРЕТИЧНІ ВІДОМОСТІ**

1. Директива #**define** визначає ідентифікатор і послідовність символів, якій буде заміщатися даний ідентифікатор при його виявленні в тексті програми. Ідентифікатор також називається ім'ям макросу, а процес заміщення називається підстановкою макросу. Стандартний вид директиви наступний:
2. #define імя\_макроса последовательность\_символів

Функція ставить puts() виводить строку, вказану стрічку, на стандартному пристрої виводу. Нуль у конце строки перелічується в новій строці.

Функція ставить puts () возвращает символ «нової строки» у випадку успіху та EOF - при невдачі.

* 1. **СХЕМИ АЛГОРИТМІВ ОСНОВНИХ ФУНКЦІЙ**

Бібліотеки та директиви:

#include <stdio.h>

#include <stdlib.h>

#define NUMELEM 5

* 1. **Висновок**

Операції для роботи зі стеками:

ініціалізація стека init (s), де s - стек

зміщення елемента в стек push (s, i), де s - стек, i - елемент;

удалення елементів із стеки i = pop (s);

ввід верхнього елементу стека без його удалення i = stkTop (и), де s - стек

визначення, пуст лі стек isempty (s) возвращает 1, якщо стек пустий і 0 в противному випадку.

вивід элементов стека stkPrint (s), де s - стек

* 1. **Додаток(а)**

#include <stdio.h>

#include <stdlib.h>

#define NUMELEM 5

typedef struct mylist

{ int val;

struct mylist \*next;

}mylist;

mylist \*new\_node(int v)

{

mylist \*new;

if((new=(mylist\*)malloc(sizeof(mylist)))==NULL)return NULL;

new->val=v;

new->next=NULL;

return new;

}

mylist \*add\_front(mylist \*p, mylist \*n)

{

n->next=p;

return n;

}

void print\_mylist(mylist \*l)

{

for(; l!=NULL; l=l->next)

{

printf("%d ", l->val);

}

}

void free\_mylist(mylist \*l)

{

mylist \* n;

for(; l!=NULL; l=n)

{

n=l->next;

free(l);

}

}

int main(void)

{

mylist\* a=NULL, \*b=NULL, \*f=NULL;

int e=0, c=0;

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

{

scanf("%d", &e);

a=add\_front(a, new\_node(e));

if(e<0)b=add\_front(b, new\_node(e));

if(e>0)f=add\_front(f, new\_node(e));

}

print\_mylist(a);

puts("");

if(b!=NULL)print\_mylist(b);

puts("");

if(f!=NULL)print\_mylist(f);

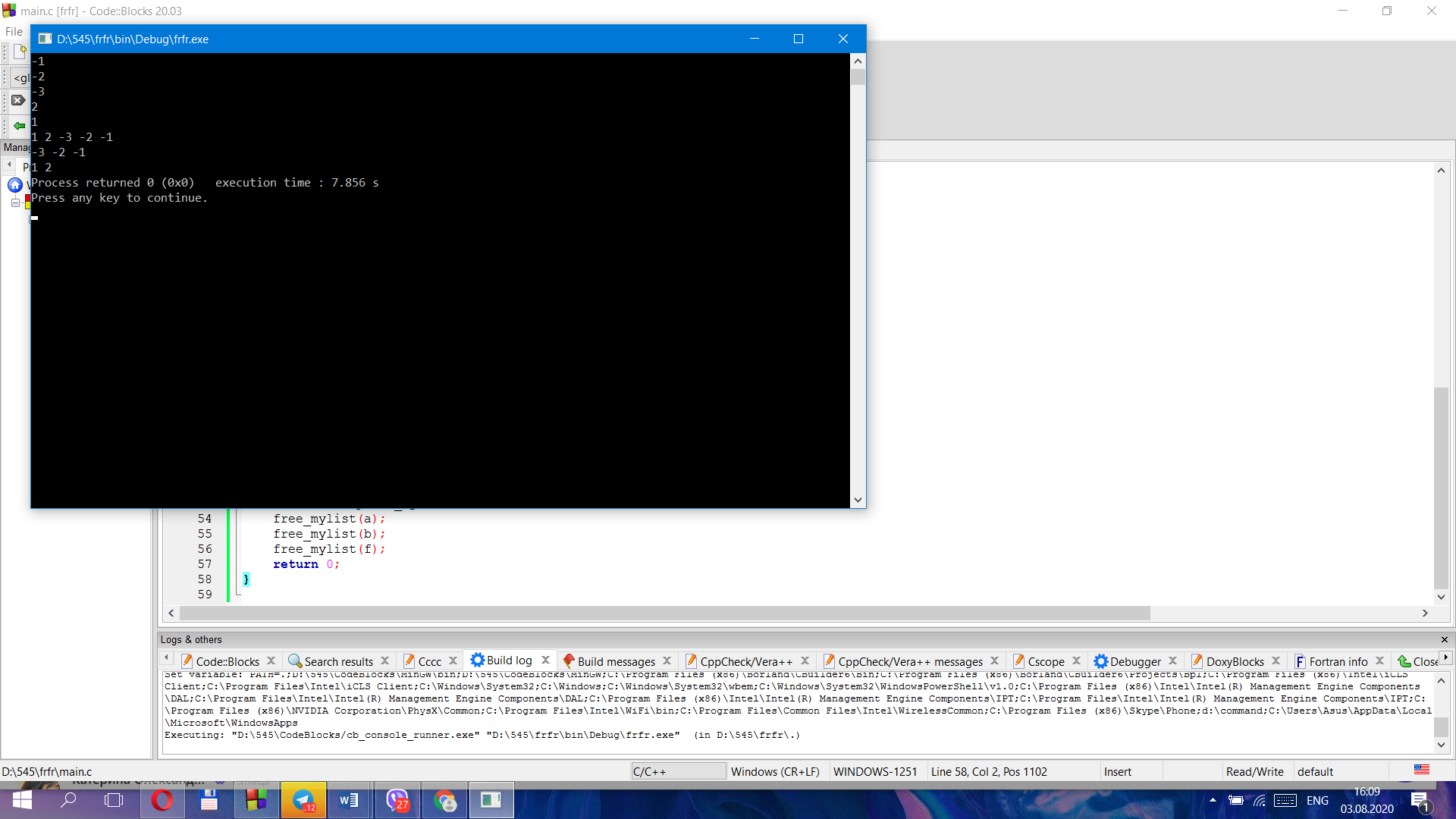
free\_mylist(a);

free\_mylist(b);

free\_mylist(f);

return 0;}

* 1. **Додаток(б)**





1. **Завдання 13**
   1. **ПОСТАНОВКА ЗАДАЧІ**
      1. **Опис завдання**

Варіант-5

Створіть файл дійсних чисел. Підрахуйте кількість елементів файлу, більших за *n* (кількість елементів файлу, *n* та самі елементи вводяться з клавіатури).

1. . Задано текст, слова в якому розділені пробілами і розділовими знаками. Розробити програму, яка вилучає в кожному слові цього тексту всі наступні входження першої літери.

* 1. **ТЕОРЕТИЧНІ ВІДОМОСТІ**

char імя [довжина];

Об'єднання строкі в S має то синтаксис, що і розміщення одномерного символічного массива. Длина строки повинна бути однаково цільовим значенням (у стандартах C89 - константа, в стандартах C99 може бути вироблена). Длина строкі показує з участю одного символу на зберігання завершеного нуля, тому максимальне число значущих символів в строці на одиницю меншої її довжини. Наприклад, строка може містити максимально двадцять символів, якщо розміщується наступна система:

char str [21]; Ініціалізація строкі в S виконується у своїй рекламі, використовуючи наступний синтаксис:

char str [довжина] = строковий літерал;

* 1. **СХЕМИ АЛГОРИТМІВ ОСНОВНИХ ФУНКЦІЙ**

Використані бібліотеки:

#include <string.h>

#include <stdio.h>

* 1. **Висновок**

Неможливі функції сканування при введених строгих даних містяться в тому, що символи розбиваються на даній функції, використовуючи:

перевод строкі,

табуляція;

пробел.

Таким чином, використовуючи дану функцію, невозможно ввести строгу, містити кілька слов, розділених пробелами або табуляції. Наприклад, якщо в попередній програмі користувач ввів строгу: "Сообщение из нескольких слов", то на екрані буде виведено тільки "Сообщение".

Для введення та виходу напруги в бібліотеку stdio.h вміщено спеціалізовані функції отримує та ставить.

* 1. **Додаток(а)**

1.

#include <string.h>

#include <stdio.h>

int main (void)

{

int a[100],i,kk=0, n,k;

printf ("vvedit kilkist");

scanf("%d" ,&k);

printf ("vvedit massiv");

for(i=0;i<k;i++){ scanf("%d" ,&a[i]);}

printf ("vvedit chislo");

scanf("%d" ,&n);

for(i=0;i<k;i++){

if(a[i]>n){kk++;}}

printf("kilkist = %d",kk);

return 0;}

2.

#include <string.h>

#include <stdio.h>

int main (void)

{

char str[50],str2[100];

int k=0, j=0,i=0,kk;

printf ("print ");

gets(str);

kk=str[0];

for (j=0;j<strlen(str);j++){

if(str[j]!=kk){k++; str2[i]=str[j];i++;}}

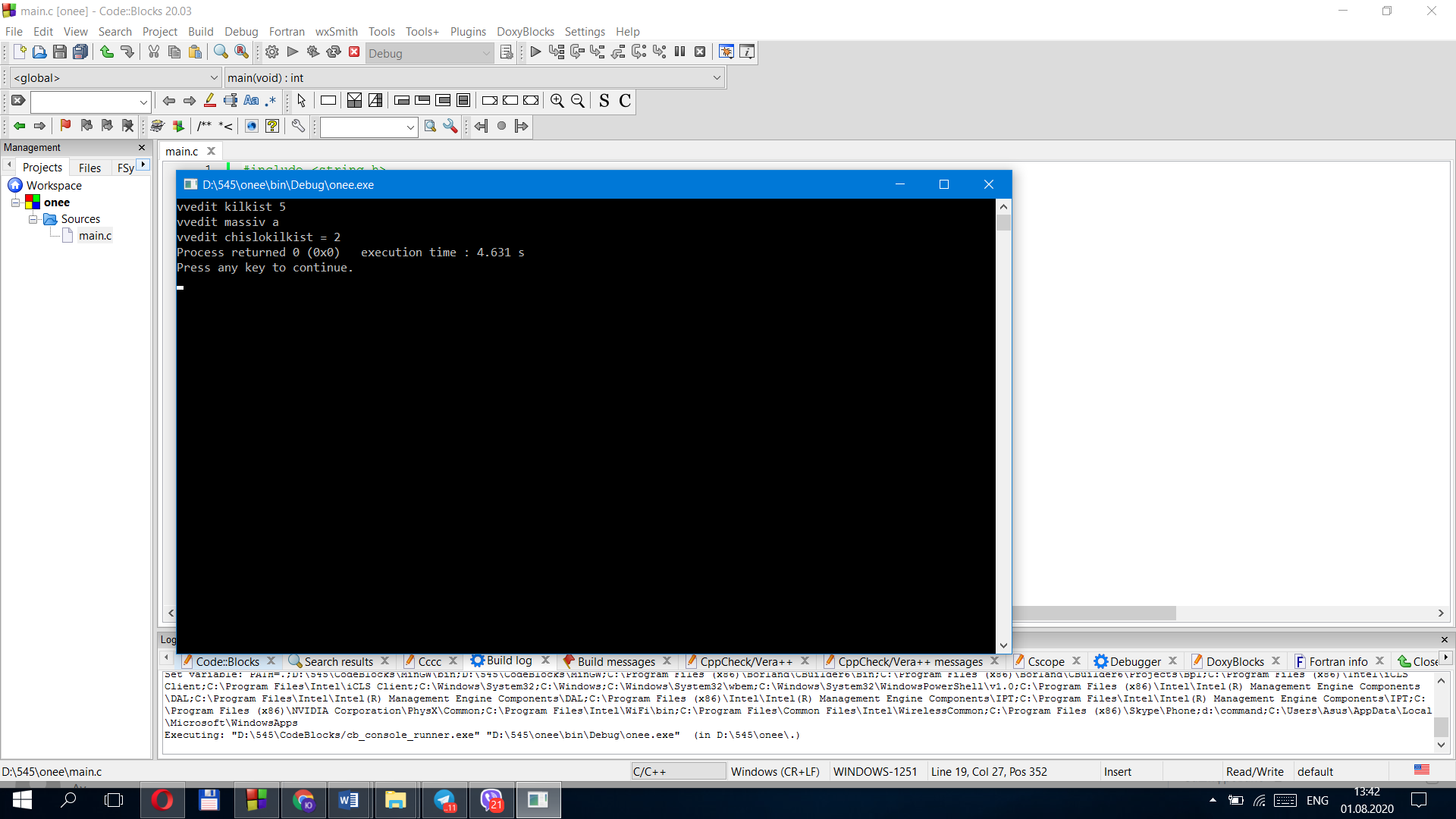
for(i=0;i<k;i++){

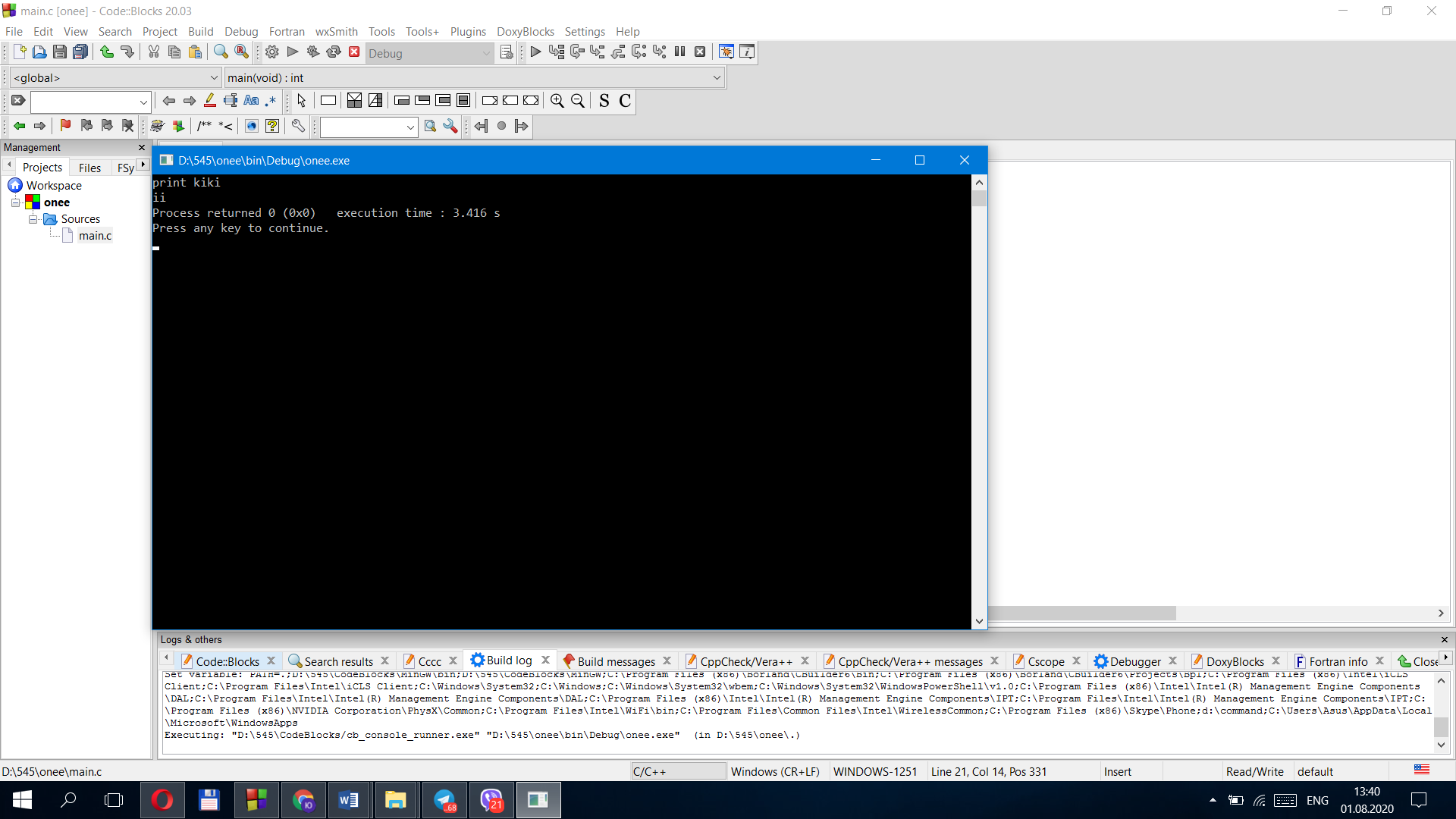
printf("%c",str2[i]);}

return 0;}

* 1. **Додаток(б)**

1.



2.

1. **Завдання 14**
   1. **ПОСТАНОВКА ЗАДАЧІ**
      1. **Опис завдання**

Варіант-5

1. Дано рядок *S*. Перевірити баланс дужок в ньому. Вивести повідомлення про виконання.

*Вхідні дані:* *S* =*"2\*X1+(3\*X2/(X2+3\*X1)/(X2+25\*X"*.

*Вихідні дані:* "*В даному рядку більше символів "(" на 2 одиниці*")

2. Задано символьні рядки. Рядок складається з декількох слів (наборів символів), які розділяються не менше ніж одним символом пробілу (' '). Слова у рядку складаються з букв латинського алфавіту. Для кожного рядка знайти:

* 1. Найдовше слово в рядку;

3. Задано символьні рядки. Рядок, який складається із декількох послідовностей символів (наборів символів), які розділяються не менше ніж одним символом пробілу (' '). Для кожного рядка знайти послідовності символів у яких є:

* 1. ідентифікатори;

* 1. **ТЕОРЕТИЧНІ ВІДОМОСТІ**

Якщо файл C містить прагму hdrstop при компіляції з / YX або / Yc, компілятор зберігає стан компіляції до місця розташування прагми. Складений стан будь-якого коду, який слідує за прагмою, не зберігається.

Кількість функцій count не має параметрів.

   id типа short;

* 1. **СХЕМИ АЛГОРИТМІВ ОСНОВНИХ ФУНКЦІЙ**

Використані бібліотеки та прагма:

#include <stdio.h>

#include <string.h>

#pragma hdrstop

* 1. **Висновок**

1. У разі успішного читання рядка, повертається покажчик на масив в який поміщені лічені дані (повертається аргумент s).  
     
   Якщо досягнуто кінця файлу, а дані не були прочитані, то повертається NULL.  
     
   Якщо при читанні дані сталася помилка, то повертається NULL, а в змінну errno записується код помилки. При цьому стан масиву, в який повинна була зберегтися зчитана рядок, невизначено.
2. **опис:**
3. Функція gets зчитує рядок з стандартного потоку введення (stdin) і поміщає її в масив вказаний аргументом s. Читання рядка проводиться поки не буде зустрінутий символ «перехід на новий рядок», або не буде досягнутий кінець файлу.  
     
   Якщо читання рядка завершилося з зчитування символу «перехід на новий рядок», то символ «перехід на новий рядок" не записується в масив, а замінюється символом «кінець рядка»  
     
   Застосування цієї функції не рекомендується, так як розмір зчитується рядки не обмежується розміром масиву, в який повинна бути записана прочитується рядок. В результаті може виникнути переповнення масиву і несанкціонована запис в пам'ять, що призведе до помилок в роботі програми або її зависання.  
     
   Рекомендується використовувати функцію fgets замість функції gets.

* 1. **Додаток(а)**

1.

#include <stdio.h>

#include <string.h>

#pragma hdrstop

//---------------------------------------------------------------------------

#pragma argsused

int main(int argc, char\* argv[])

{

char s[100];

int i,l,p;

l=0;p=0;

gets(s);

for(i=0;i<strlen(s);i++){

if(s[i]=='('){

l++;}

if(s[i]==')'){

p++;}};

if (l>p){printf("netu %d )", l-p);};

if (l<p){printf("netu %d (", p-l);};

if (l==p){printf("odnakovo");};

return 0;

}

2.

#include <stdio.h>

#include <string.h>

main() {

char s[100];

int i, id, max, count, len;

gets(s);

len = strlen(s);

max = 0;

id = 0;

count = 0;

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

if (s[i] != ' ') count += 1;

else {

if (count > max) {

max = count;

id = i - count;

}

count = 0;

}

if (count > max) {

max = count;

id = i - count;

}

max += id;

for (i=id; i<max; i++)

putchar(s[i]);

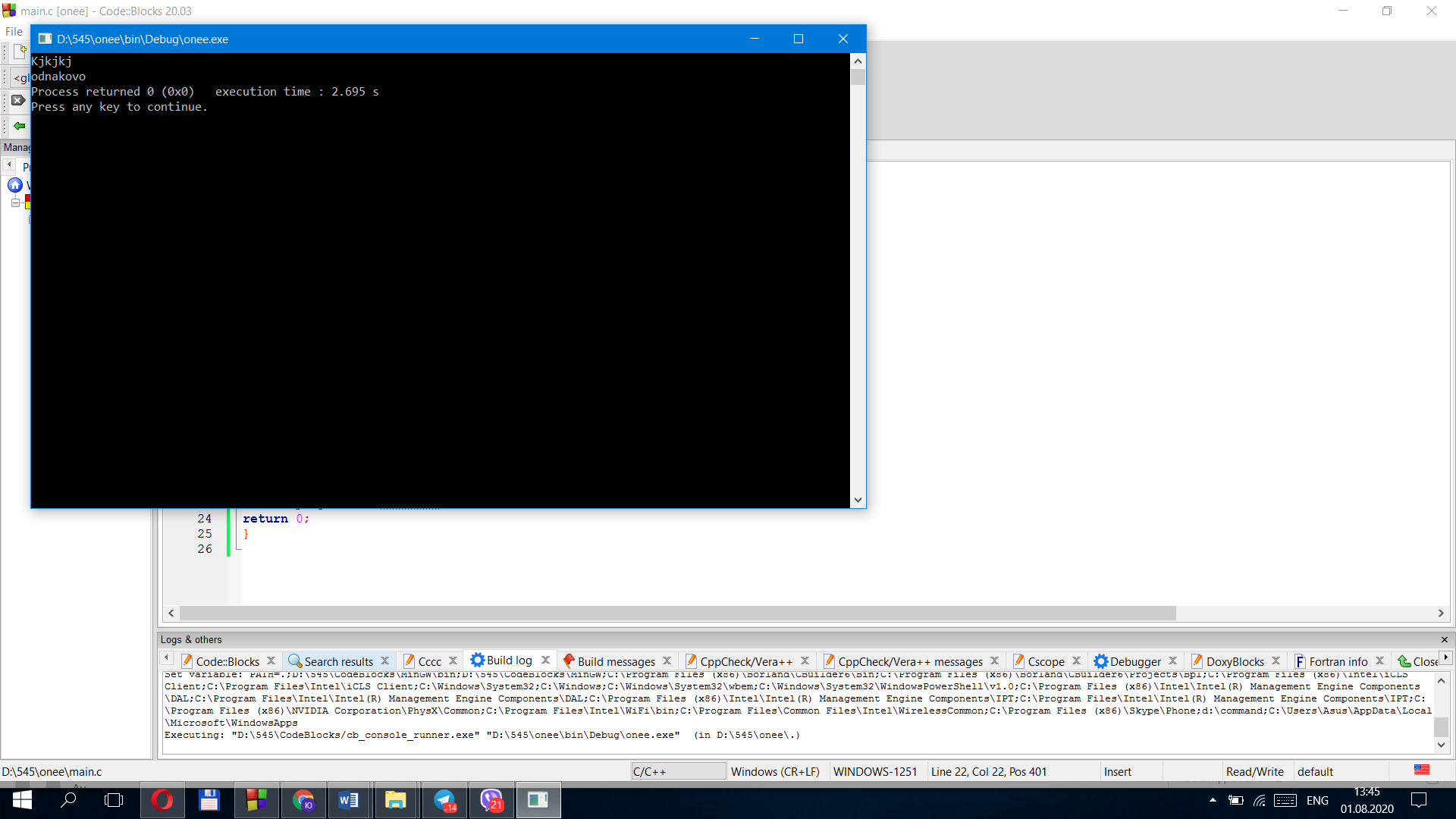
printf("\n");

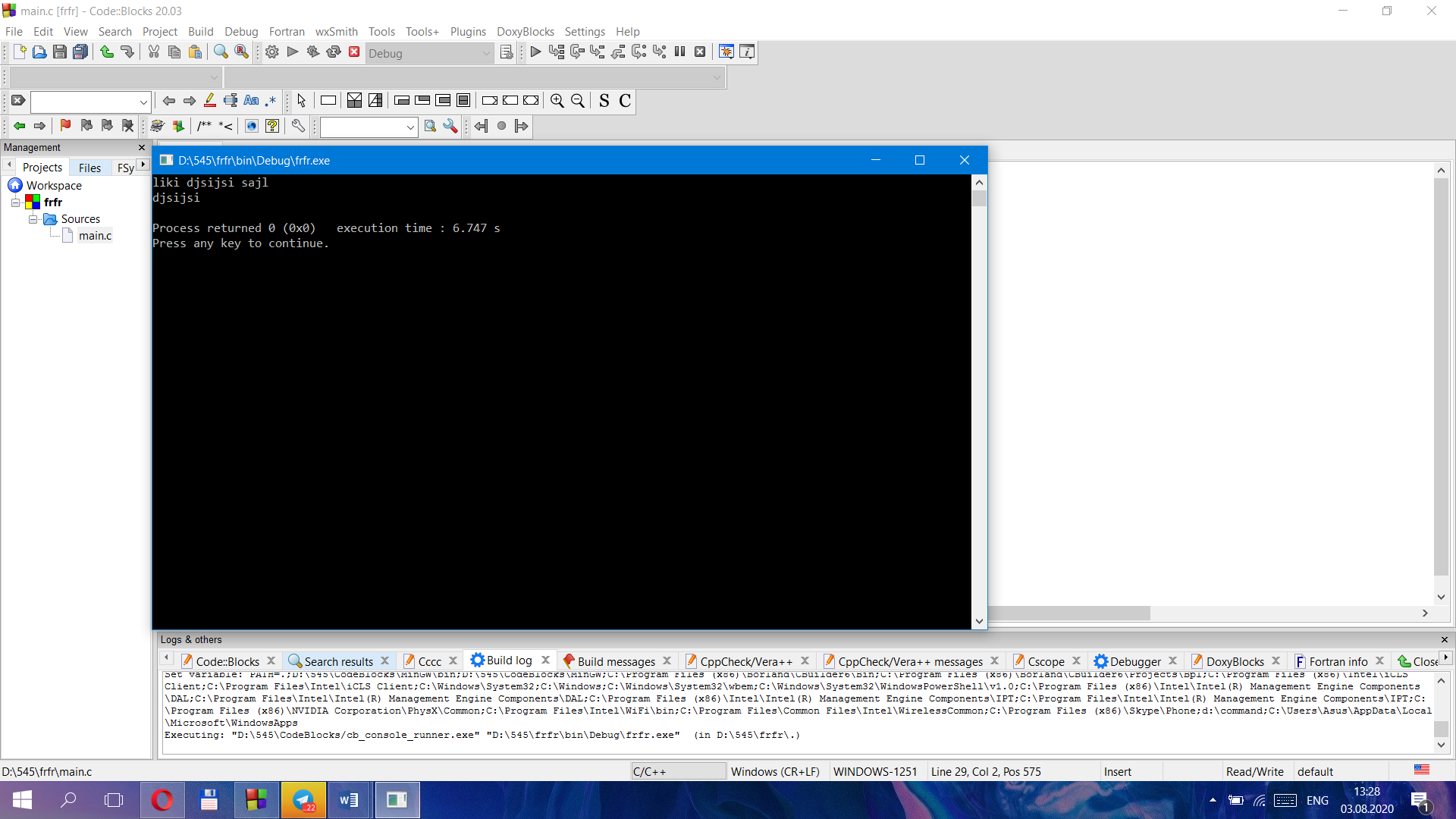
}

3.

#include <stdio.h>  
#include <string.h>  
main() {  
    char s[100];  
    short i, f;  
    scanf("%s", s);  
    f = 1;  
   
    for (i=0; i < strlen(s); i++)  
        if (!(s[i] <= 'z' && s[i] >= 'a' ||  
              s[i] <= 'Z' && s[i] >= 'A' ||  
              s[i] == '\_' ||  
              s[i] <= '9' && s[i] >= '0') ) {  
                printf("No\n");  
                f = 0;  
                break;  
        }  
   
    if (f == 1)  
        if (s[0] <= '9' && s[0] >= '0')  
            printf("No\n");  
        else  
            printf("Yes\n");      
}

* 1. **Додаток(б)**

1. 

2. 

3.