МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КОЛЕДЖ ЧЕРНІВЕЦЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

ІМЕНІ ЮРІЯ ФЕДЬКОВИЧА

**ЗВІТ**

про навчальну практику

студента II курсу, спеціальності

«Комп`ютерні науки»

214 групи

*Ляшенко Андрій Романович*

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](#_Toc47182033)

[**ПОСТАНОВКА ЗАДАЧІ** 7](#_Toc47182034)

[**ТЕОРЕТИЧНІ ВІДОМОСТІ** 8](#_Toc47182035)

[**СХЕМИ АЛГОРИТМІВ ОСНОВНИХ ФУНКЦІЙ** 8](#_Toc47182036)

[**Висновок** 8](#_Toc47182037)

[**Додаток(а)** 9](#_Toc47182038)

[**Додаток(б)** 9](#_Toc47182039)

[**Завдання 2** 10](#_Toc47182040)

[**ПОСТАНОВКА ЗАДАЧІ** 10](#_Toc47182041)

[**ТЕОРЕТИЧНІ ВІДОМОСТІ** 10](#_Toc47182042)

[**СХЕМИ АЛГОРИТМІВ ОСНОВНИХ ФУНКЦІЙ** 11](#_Toc47182043)

[**Висновок** 11](#_Toc47182044)

[**Додаток(а)(1)** 11](#_Toc47182045)

[**Додаток(б)(1)** 12](#_Toc47182046)

[**Додаток(а)(2)** 12](#_Toc47182047)

[**Додаток(б)(2)** 14](#_Toc47182048)

[**Завдання 3** 15](#_Toc47182049)

[**ПОСТАНОВКА ЗАДАЧІ** 15](#_Toc47182050)

[**СХЕМИ АЛГОРИТМІВ ОСНОВНИХ ФУНКЦІЙ** 16](#_Toc47182051)

[**Висновок** 16](#_Toc47182052)

[**Додаток(а)(1)** 17](#_Toc47182053)

[**Додаток(б)(1)** 17](#_Toc47182054)

[**Додаток(а)(2)** 18](#_Toc47182055)

[**Додаток(б)(2)** 19](#_Toc47182056)

[**Додаток(а)(3)** 19](#_Toc47182057)

[**Додаток(б)(3)** 20](#_Toc47182058)

[**Завдання 4** 22](#_Toc47182059)

[**ПОСТАНОВКА ЗАДАЧІ** 22](#_Toc47182060)

[**СХЕМИ АЛГОРИТМІВ ОСНОВНИХ ФУНКЦІЙ** 22](#_Toc47182061)

[**Висновок** 23](#_Toc47182062)

[**Додаток(а)(2)** 23](#_Toc47182063)

[**Додаток(б)(2)** 24](#_Toc47182064)

[**Завдання 5** 25](#_Toc47182065)

[**ПОСТАНОВКА ЗАДАЧІ** 25](#_Toc47182066)

[**ТЕОРЕТИЧНІ ВІДОМОСТІ** 26](#_Toc47182067)

[**СХЕМИ АЛГОРИТМІВ ОСНОВНИХ ФУНКЦІЙ** 26](#_Toc47182068)

[**Висновок** 27](#_Toc47182069)

[**Додаток(а)** 27](#_Toc47182070)

[**Додаток(б)** 28](#_Toc47182071)

[**Завдання 6** 29](#_Toc47182072)

[**ПОСТАНОВКА ЗАДАЧІ** 29](#_Toc47182073)

[**ТЕОРЕТИЧНІ ВІДОМОСТІ** 30](#_Toc47182074)

[**СХЕМИ АЛГОРИТМІВ ОСНОВНИХ ФУНКЦІЙ** 30](#_Toc47182075)

[**Висновок** 30](#_Toc47182076)

[**Додаток(а)(1)** 31](#_Toc47182077)

[**Додаток(б)(1)** 32](#_Toc47182078)

[**Додаток(а)(2)** 32](#_Toc47182079)

[**Додаток(б)(2)** 34](#_Toc47182080)

[**Завдання 7** 35](#_Toc47182081)

[**ПОСТАНОВКА ЗАДАЧІ** 35](#_Toc47182082)

[**ТЕОРЕТИЧНІ ВІДОМОСТІ** 36](#_Toc47182083)

[**СХЕМИ АЛГОРИТМІВ ОСНОВНИХ ФУНКЦІЙ** 36](#_Toc47182084)

[**Висновок** 36](#_Toc47182085)

[**Додаток(а)(1)** 37](#_Toc47182086)

[**Додаток(б)(1)** 38](#_Toc47182087)

[**Додаток(а)(2)** 38](#_Toc47182088)

[**Додаток(б)(2)** 39](#_Toc47182089)

[**Завдання 8** 40](#_Toc47182090)

[**ПОСТАНОВКА ЗАДАЧІ** 41](#_Toc47182091)

[**ТЕОРЕТИЧНІ ВІДОМОСТІ** 41](#_Toc47182092)

[**СХЕМИ АЛГОРИТМІВ ОСНОВНИХ ФУНКЦІЙ** 42](#_Toc47182093)

[**Висновок** 42](#_Toc47182094)

[**Додаток(а)** 42](#_Toc47182095)

[**Додаток(б)** 43](#_Toc47182096)

[**Завдання 9** 43](#_Toc47182097)

[**ПОСТАНОВКА ЗАДАЧІ** 44](#_Toc47182098)

[**ТЕОРЕТИЧНІ ВІДОМОСТІ** 45](#_Toc47182099)

[**СХЕМИ АЛГОРИТМІВ ОСНОВНИХ ФУНКЦІЙ** 45](#_Toc47182100)

[**Висновок** 45](#_Toc47182101)

[**Додаток(а)** 45](#_Toc47182102)

[**Додаток(б)** 46](#_Toc47182103)

[**Завдання 10** 47](#_Toc47182104)

[**ПОСТАНОВКА ЗАДАЧІ** 48](#_Toc47182105)

[**ТЕОРЕТИЧНІ ВІДОМОСТІ** 48](#_Toc47182106)

[**СХЕМИ АЛГОРИТМІВ ОСНОВНИХ ФУНКЦІЙ** 49](#_Toc47182107)

[**Висновок** 49](#_Toc47182108)

[**Додаток(а)** 50](#_Toc47182109)

[**Додаток(б)** 52](#_Toc47182110)

[**Завдання 11** 54](#_Toc47182111)

[**ПОСТАНОВКА ЗАДАЧІ** 54](#_Toc47182112)

[**ТЕОРЕТИЧНІ ВІДОМОСТІ** 55](#_Toc47182113)

[**СХЕМИ АЛГОРИТМІВ ОСНОВНИХ ФУНКЦІЙ** 56](#_Toc47182114)

[**Висновок** 57](#_Toc47182115)

[**Додаток(а)(1)** 58](#_Toc47182116)

[**Додаток(б)(1)** 65](#_Toc47182117)

[**Додаток(а)(2)** 67](#_Toc47182118)

[**Додаток(б)(2)** 69](#_Toc47182119)

[**Завдання 12** 71](#_Toc47182120)

[**ПОСТАНОВКА ЗАДАЧІ** 71](#_Toc47182121)

[**СХЕМИ АЛГОРИТМІВ ОСНОВНИХ ФУНКЦІЙ** 72](#_Toc47182122)

[**Висновок** 72](#_Toc47182123)

[**Додаток(а)** 73](#_Toc47182124)

[**Додаток(б)** 75](#_Toc47182125)

[**Завдання 13** 76](#_Toc47182126)

[**ПОСТАНОВКА ЗАДАЧІ** 76](#_Toc47182127)

[**ТЕОРЕТИЧНІ ВІДОМОСТІ** 77](#_Toc47182128)

[**СХЕМИ АЛГОРИТМІВ ОСНОВНИХ ФУНКЦІЙ** 78](#_Toc47182129)

[**Висновок** 78](#_Toc47182130)

[**Додаток(а)(1)** 78](#_Toc47182131)

[**Додаток(б)(1)** 79](#_Toc47182132)

[**Додаток(а)(2)** 80](#_Toc47182133)

[**Додаток(б)(2)** 81](#_Toc47182134)

[**Завдання 14** 82](#_Toc47182135)

[**ПОСТАНОВКА ЗАДАЧІ** 82](#_Toc47182136)

[**ТЕОРЕТИЧНІ ВІДОМОСТІ** 83](#_Toc47182137)

[**СХЕМИ АЛГОРИТМІВ ОСНОВНИХ ФУНКЦІЙ** 84](#_Toc47182138)

[**Висновок** 84](#_Toc47182139)

[**Додаток(а)(1)** 84](#_Toc47182140)

[**Додаток(б)(1)** 85](#_Toc47182141)

[**Додаток(а)(2)** 86](#_Toc47182142)

[**Додаток(б)(2)** 88](#_Toc47182143)

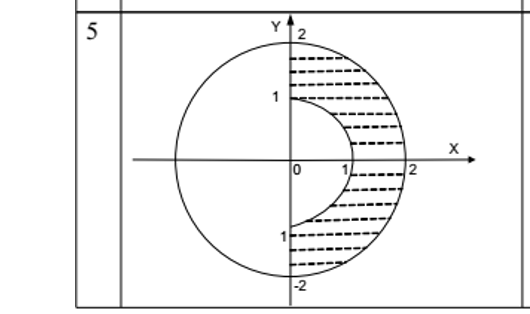
[**Додаток(а)(3)** 88](#_Toc47182144)

[**Додаток(б)(3)** 90](#_Toc47182145)

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

Варіант 5

Створити алгоритм та написати програму для розв'язку наступної задачі: перевірку попадання точки ( яка буде вводитись вручну із клавіатури) в заштриховану область.

* + 2. ****
    3. **ТЕОРЕТИЧНІ ВІДОМОСТІ**

Під час виконання завдання потрібно було знати рівняння кола з центром в початку координат x^2 + y^2 = r^2. Виходячи з цього за допомогою системи з 2 нерівностей окреслюємо площу в формі “бублика”, попередньо відкинувши варіанти 2 та 3 чвертей.

* 1. **СХЕМИ АЛГОРИТМІВ ОСНОВНИХ ФУНКЦІЙ**

Список підключених бібліотек:

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

* 1. **Висновок**

Під час написання цієї роботи було розроблено математичну модель. У результаті виконання отримано працюючу програму. У якій ми можемо визначати чи належать точка заданій умові у постановці задачі за введеними координатами точки.

* 2. **Додаток(а)**

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

int main (){

float x,y;

printf("Enter :\n");

printf("x : ");

scanf("%f",&x);

printf("\ny : ");

scanf("%f",&y);

if( (y>0) && (x<0))

printf("NO\n");

else

if((pow(x,2)+pow(y,2)<=4) && (pow(x,2)+pow(y,2)>=1))

printf("YES\n");

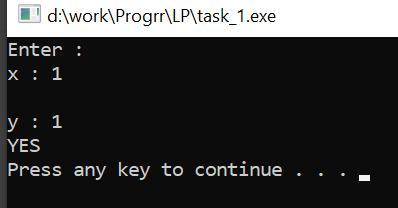
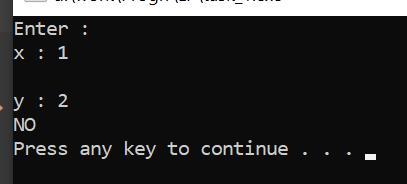
else

printf("NO\n");

system("pause");

return 0;

}

* 1. **Додаток(б)**
  2. 

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

Варіант 5

1) Обчислити та вивести середн∙ 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**

У даних програмах використано генератор випадкових чисел

* 1. **СХЕМИ АЛГОРИТМІВ ОСНОВНИХ ФУНКЦІЙ**
  2. Список підключених бібліотек:
  3. #include <stdio.h>
  4. #include <stdlib.h>
  5. #include <time.h>
  6. Для завдання №2 було створено функції:
  7. void Array\_create(int \*t, int \*h, int \*p) – для заповнення масиву p за формулою, заданою завданням.
  8. void SortArr(int \*p) – для сортування масиву p.
  10. **Висновок**
  11. Дані завдання допомогли розвинути вміння роботи над масивами, а також відточило вміння роботи з функціями користувача.
  12. **Додаток(а)(1)**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

/\*5. Обчислити та вивести середн∙ арифметичне елементiв розмiщених на непарних

мiсцях в масивi R(9).

\*/

int main (){

int r[9];

const n = 9;

int s =0;

double sz;

printf("array:\n");

srand(time(NULL));

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

{

r[i] = rand()%10;

s+=r[i];

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

}

sz=(double)s/9;

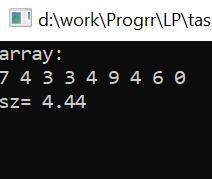
printf("\nsz= %.2lf\n",sz);

system("pause");

return 0;

}

* 1. **Додаток(б)(1)**



* 1. **Додаток(а)(2)**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

/\* 5. 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.

\*/

void ArrayP\_create(int \*t,int \*h,int \*p)

{

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

{

p[i]=t[i]-h[i];

}

}

void SortArr(int \*p)

{ int \*head;

head = p;

for (int i = 6; i > 0 ; i--)

{ p = head;

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

{

if(\*p>\*(p+1))

{

int tmp;

tmp = \*p;

\*p = \*(p+1);

\*(p+1) = tmp;

}

p++;

}

}

}

int main (){

const n = 6;

int t[n], h[n],p[n];

srand(time(NULL));

printf("array T:\n");

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

{

t[i]=rand()%10;

h[i]=rand()%10;

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

}

printf("\narray H:\n");

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

{

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

}

printf("\narray P:\n");

ArrayP\_create(t,h,&p);

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

{

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

}

printf("\nSorted array P:\n");

SortArr(&p);

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

{

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

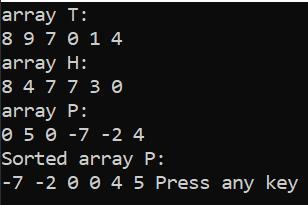
}

system("pause");

return 0;

}

* 1. **Додаток(б)(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.
3. Задано масив Х(5,5). Всi нульовi елементи головної дiагоналi замiнити сумою додатних елементiв масиву. Вивести вихiдну матрицю, знайдену суму та перетворений масив.
   1. **СХЕМИ АЛГОРИТМІВ ОСНОВНИХ ФУНКЦІЙ**

Список підключених бібліотек:

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

* 1. **Висновок**

У результаті ми отримали програму 1, яка в матриці знаходить суму елементів, які менші за випадкове число, програму 2, яка формує вектор з мінімальних чисел згенерованої матриці, а також у сформованому векторі рахує суму елементів та кількість додатних, і програму 3, яка всі нульові елементи головної діагоналі замінює сумою додатних елементiв двовимірного масиву.

* 1. **Додаток(а)(1)**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

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

int main (){

int q[4][5];

const col = 5 , row = 4;

srand(time(NULL));

int d,s =0;

d = rand()%9+1;

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

printf("matrix q:\n");

for (int i = 0; i < row; i++,printf("\n"))

{

for (int j = 0; j < col; j++)

{

q[i][j] = rand()%10;

if(q[i][j] < d) s+=q[i][j];

printf("%d ",q[i][j]);

}

}

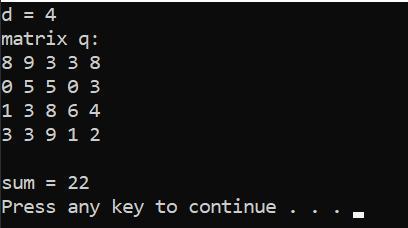
printf("\nsum = %d\n",s);

system("pause");

return 0;

}

* 1. **Додаток(б)(1)**

****

* 1. **Додаток(а)(2)**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

/\*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.\*/

int main (){

int b[4][4];

const int n = 4;

int a[4],min;

srand(time(NULL));

printf("matrix b:\n");

for (int i = 0; i < n; i++,printf("\n"))

{

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

{

b[i][j] = rand()%20-10;

printf("%d ",b[i][j]);

}

}int k=0;

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

{

min = b[0][j];

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

{

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

}

a[k++] = min;

}

printf("array A:\n");

int s=0,positive\_counter = 0;

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

{

s+=a[i];

if(a[i]>0) positive\_counter++;

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

}

printf("\nsum = %d ", s);

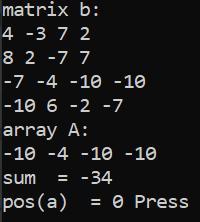
printf("\npos(a) = %d ", positive\_counter);

system("pause");

return 0;

}

* 1. **Додаток(б)(2)**

****

* 1. **Додаток(а)(3)**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

/\*Задано масив Х(5,5). Всi нульовi елементи головної дiагоналi замiнити сумою додатних елементiв масиву.

Вивести вихiдну матрицю, знайдену суму та перетворений масив.\*/

int main (){

int x[5][5];

const int n = 5;

int s =0;

printf("matrix X:\n");

srand(time(NULL));

for (int i = 0; i < n; i++,printf("\n"))

{

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

{

x[i][j] = rand()%10;

printf("%d ",x[i][j]);

s+= x[i][j];

}

}

printf("new matrix\n");

for (int i = 0; i < n; i++,printf("\n"))

{

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

{

if((i == j) && (x[i][j] == 0)) x[i][j] = s;

printf("%d ",x[i][j]);

}

}

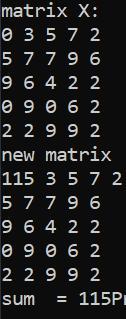
printf("sum = %d",s);

system("pause");

return 0;

}

* 1. **Додаток(б)(3)**

****

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

Варіант 5

1. Завдання 1 ідентичне до завдання 3 (лабораторна робота №3)
2. Дано дійсну матрицю *А* розмірності *n*x*m*. Заповнити її випадковими цілими числами з відрізка [-10; 10]. Знайти середнє арифметичне максимального та мінімального елементів матриці.

*Вхідні дані:* *n =* 3; *m =* 4; .*Вихідні дані:* *avg* = 2.

* 1. **СХЕМИ АЛГОРИТМІВ ОСНОВНИХ ФУНКЦІЙ**

Список підключених бібліотек:

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

* 1. **Висновок**

У результаті отримано програму, яка знаходить середнє арифметичне між мінімальним та максимальним елементами матриці. Також під час роботи було удосконалено вміння роботи з двовимірними масивами.

* 1. **Додаток(а)(2)**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

/\*5. Дано дійсну матрицю А розмірності nxm. Заповнити її випадковими цілими числами з відрізка [-10; 10].

Знайти середнє арифметичне максимального та мінімального елементів матриці.\*/

int main (){

const int n = 3, m = 4;

int a[n][m];

printf("matrix:\n");

srand(time(NULL));

for (int i = 0; i < n; i++,printf("\n"))

{

for (int j = 0; j < m; j++)

{

a[i][j] = rand()%20-10;

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

}

}

int min, max;

min = a[0][0];

max = a[0][0];

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

{

for (int j = 0; j < m; j++)

{

if(min>a[i][j]) min = a[i][j];

if(max<a[i][j]) max = a[i][j];

}

}

float avg;

avg =(float)(min+max)/2;

printf("min = %d, max = %d\n", min,max);

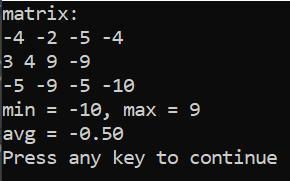
printf("avg = %.2f\n", avg);

system("pause");

return 0;

}

* 1. **Додаток(б)(2)**

****

1. **Завдання 5**
   1. **ПОСТАНОВКА ЗАДАЧІ**
      1. **Опис завдання**

Варіант 5

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.

* 1. **ТЕОРЕТИЧНІ ВІДОМОСТІ**

Під час обробки масиву використовується прапорець, якій змінює значення після того як в масиві змінюється знак. Кількість таких змін і буде нашою шуканою величиною.

* 1. **СХЕМИ АЛГОРИТМІВ ОСНОВНИХ ФУНКЦІЙ**

Список підключених бібліотек:

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

* 1. **Висновок**

Під час роботи було розроблено програму, яка підраховує кількість ділянок в масиві на яких змінюється знак.

* 1. **Додаток(а)**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

/\*5. Дано масив цілих чисел a[i], і = 1, … , n. Використовуючи вказівники підрахувати кількість ділянок масиву,

де елементи змінюють знак.\*/

int main()

{

int n;

printf("Enter n :");

scanf("%d", &n);

printf("array a:\n");

srand(time(NULL));

int a[n], \*p;

p = a;

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

{

a[i] = rand() % 20 - 10;

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

}

int counter = 0;

int f;

if (\*p > 0)

{

f = 1;

}

else

{

f = 0;

}

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

{

if (f == 1)

{

if (\*p < 0)

{

f = 0;

counter++;

}

p++;

}

else

{

if(\*p>0)

{

f = 1;

counter++;

}

p++;

}

}

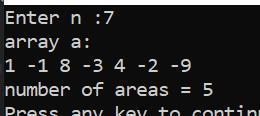
printf("\nnumber of areas = %d\n",counter );

system("pause");

return 0;

}

* + - 1. **Додаток(б)**

****

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

Варіант 5

1) Вивести на екран таблицю результатів голосування, де у рядках є дані з населених пунктів, а у стовпцях - дані щодо конкретних кандидатів В яких населених пунктах другий і четвертий кандидати набрали максимальну кількість голосів?

2) 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; б) застосовуючи до рядка стандартні функції. Вивести ім’я та кількість букв у прізвищі.

* 1. **ТЕОРЕТИЧНІ ВІДОМОСТІ**

Під час написання програми 2 використовувались набір стандартних функцій в бібліотеці string.h. Була використана функція strtok яка ділить рядок на лексеми, щоб вивести ім`я достатньо було ввести лічильник слів і вивести 2 слово. Якщо розглядати як масив символів то під час знаходження першого пробілу з значенням int 32 ми виводимо ім'я посимвольно, довжину прізвища знаходимо віднявши від позиції пробіла 1 символ(власне пробіл).

* 1. **СХЕМИ АЛГОРИТМІВ ОСНОВНИХ ФУНКЦІЙ**

Список підключених бібліотек:

1) #include <stdio.h>

#include <stdlib.h>

#include <time.h>

2) #include <stdio.h>

#include <stdlib.h>

#include <string.h>

* 1. **Висновок**

Під час розробки 2 програми було розвинуто вміння роботи з рядками як масивом символів та за допомогою стандартних функцій.

* 1. **Додаток(а)(1)**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

/\*Вивести на екран таблицю результатів голосування, де у рядках є дані з населених пунктів, а у стовпцях - дані

щодо конкретних кандидатів

5. В яких населених пунктах другий і четвертий кандидати на¬брали максимальну кількість голосів?

\*/

int main()

{

int max\_2 = 0, max\_4 = 0, place\_2, place\_4;

srand(time(NULL));

int s1 = 0, s2 = 0, s3 = 0;

int \*\*p;

p = (int \*)calloc(6, sizeof(int \*));

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

{

p[i] = (int \*)calloc(5, sizeof(int));

for (int j = 0; j < 5; j++)

p[i][j] = rand() % 10 \* i + 50;

}

printf("1 |2 |3 |4 |5 |<-candidates \n--+--+--+--+--+\n");

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

{

for (int j = 0; j < 5; j++)

{

printf("%d|", p[i][j]);

if (j == 1)

{

if (max\_2 < p[i][j])

{

max\_2 = p[i][j];

place\_2 = i+1;

}

}

if (j == 3)

{

if (max\_4 < p[i][j])

{

max\_4 = p[i][j];

place\_4 = i+1;

}

}

}

printf("punkt%d", i + 1);

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

}

printf("\ncandidate 2 in punkt #%d, %d votes",place\_2,max\_2);

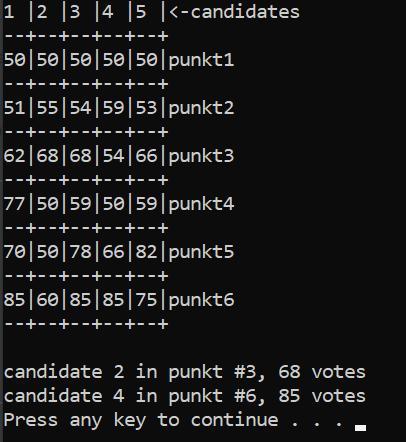
printf("\ncandidate 4 in punkt #%d, %d votes\n",place\_4,max\_4);

system("pause");

return 0;

}

* 1. **Додаток(б)(1)**

****

* 1. **Додаток(а)(2)**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

/\*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; \*/

int main()

{

char s[100];

printf("text:\n");

gets(s);

int a\_counter = 0;

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

{

if ((s[i] == 'a') || (s[i] == 'A'))

{

a\_counter++;

}

}

int NAME\_position;

printf("length = %d, num of a = %d\n", strlen(s), a\_counter);

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

{

if (s[i] == 32)

{

NAME\_position = i + 1;

i++;

while (s[i] != 32)

{

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

i++;

}

break;

}

}

printf("\nLname length = %d\n",NAME\_position-1);

printf("\n2 case\n");

{

char \*p=strtok(s," ");

int i = 0;

while (p!=NULL)

{

if(i == 1)

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

p = strtok (NULL, " ");

i++;

}

}

char lname[30];

strcpy(lname,s-(strlen(s)-NAME\_position+1));

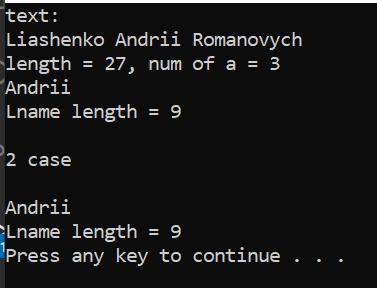
printf("\nLname length = %d\n",strlen(lname));

system("pause");

return 0;

}

* 1. **Додаток(б)(2)**

****

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

Варіант 5

1) 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 мінімальне значення.

2) 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**

В програмі 1 використовується багато функцій користувача для заповнення масиву та інших розрахунків. В програмі 2 використовується функція користувача для розрахунків.

* 1. **СХЕМИ АЛГОРИТМІВ ОСНОВНИХ ФУНКЦІЙ**

Список підключених бібліотек:

1;2) #include <stdio.h>

#include <stdlib.h>

#include <time.h>

Список використаних функцій:

void Fill\_Array(int \*a, int n) - *для заповнення масиву*

int Neg\_In\_Arr(int \*a, int n) - *повертає кількість елементів<0*

int MIN(int \*a, int n) - *повертає мінімальне значення в масиві*

int sqrSum(int \*\*a, int n, int m, int \*counter) -  *повертає значення суми квадратів крім 1, а також записує у counter їх кількість.*

* 1. **Висновок**

Було отримано 2 програми. Перша програма за допомогою функцій підраховує кількість від'ємних елементів та мінімальне значення. Друга програма підраховує суму квадрадратів елементів матриці які не є 1 та виводить кількість таких елементів.

* 1. **Додаток(а)(1)**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

/\*. 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.

5. Скільки було від’ємних значень? Визначити мінімальне значення.\*/

void Fill\_Array(int \*a, int n)

{

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

{

a[i] = rand()%20-10;

}

}

int Neg\_In\_Arr(int \*a, int n)

{int negative\_counter = 0;

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

{

if(a[i]<0) negative\_counter++;

}

return negative\_counter;

}

int MIN(int \*a, int n)

{int min = a[0];

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

{

if(min > a[i]) min = a[i];

}

return min;

}

int main (){

int n;

printf("Enter n ");

scanf("%d", &n);

int a[n];

printf("array\n" );

Fill\_Array(a,n);

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

{

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

}

int min, negative\_counter;

negative\_counter = Neg\_In\_Arr(a,n);

min = MIN(a,n);

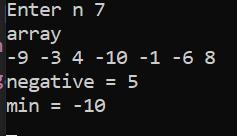
printf("\nnegative = %d\nmin = %d\n",negative\_counter,min);

system("pause");

return 0;

}

* 1. **Додаток(б)(1)**

****

* 1. **Додаток(а)(2)**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

/\*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.\*/

int sqrSum(int \*\*a, int n, int m, int \*counter)

{\*counter = 0;

int s = 0;

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

{

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

{

if (a[i][j] != 1)

{

(\*counter)++;

s += pow((a[i][j]),2);

}

}

}

return s;

}

int main()

{

int \*\*a, n = 4, m = 4;

a = (int \*\*)malloc(m \* sizeof(int \*));

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

{

a[i] = (int \*)malloc(n \* sizeof(int));

}

srand(time(NULL));

printf("matrix:\n");

for (int i = 0; i < m; i++, printf("\n"))

{

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

{

a[i][j] = rand() % 9 + 1;

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

}

}

int sum, counter;

sum = sqrSum(a,n,m,&counter);

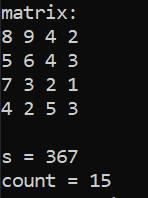
printf("\ns = %d\ncount = %d\n", sum, counter);

system("pause");

return 0;

}

1. **Додаток(б)(2)**

****

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

Варіант 5

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.

* 1. **ТЕОРЕТИЧНІ ВІДОМОСТІ**

Було написано функцію типу bool яка повертає значення true якщо число парне.

* 1. **СХЕМИ АЛГОРИТМІВ ОСНОВНИХ ФУНКЦІЙ**

Список підключених бібліотек:

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#include <time.h>

Список використаних функцій:

bool Even(int n) - *повертає значення true якщо число парне.*

* 1. **Висновок**

У результаті отримано програму яка за допомогою створеної функції рахує кількість парних елементів в масиві.

1. **Додаток(а)**

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#include <time.h>

/\*5. 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\*/

bool Even(int n)

{

return n!=0 && n%2 == 0;

}

int main (){

int a[10];

srand(time(NULL));

int counter = 0;

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

{

a[i] = rand()%10;

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

if(Even(a[i])) counter++;

}

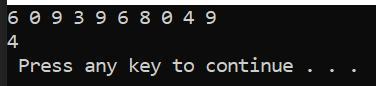
printf("\n%d\n ", counter);

system("pause");

return 0;

}

1. **Додаток(б)**

****

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

Варіант 5

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**

Було використано функцію Even з попередньої лабораторної роботи. Формування масиву b відбулося циклом в якому перевірялося парність елементів на непарних місцях.

* 1. **СХЕМИ АЛГОРИТМІВ ОСНОВНИХ ФУНКЦІЙ**

Список підключених бібліотек:

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#include <time.h>

Список використаних функцій:

bool Even(int n) - *повертає значення true якщо число парне.*

* 1. **Висновок**

Було отримано програму яка формує масив b з масиву a. Було відточено навички роботи з динамічною пам'яттю.

1. **Додаток(а)**

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#include <time.h>

/\*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.\*/

bool Even(int n)

{

return n%2 == 0;

}

int main()

{

int \*a, \*b, n, m;

printf("Enter n: ");

scanf("%d",&n);

a = (int\*)malloc(n\*sizeof(int));

srand(time(NULL));

printf("array a:\n");

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

{

a[i] = rand()%10;

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

}

int k=0;

printf("\narray b:\n");

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

{

if( Even(i) && Even(a[i]))

m++;

}

b = (int\*)malloc(m\*sizeof(int));

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

{

if( Even(i) && Even(a[i]))

b[k++] = a[i];

}

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

{

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

}

free(a);

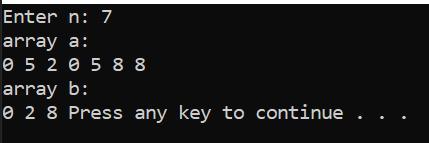
free(b);

system("pause");

return 0;

}

1. **Додаток(б)**

****

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

Варіант 5

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**

В даній програмі використовуються структуровані типи даних. І оскільки відома їх кількість запускаємо цикл for і вводимо дані з клавіатури. Для того щоб не путатися вводимо показник поточного студента і після кожного введеного студента очищується екран за допомогою system(“cls”). Також для визначення віку було розроблено функцію яка витягувала систему дату і рахувала вік.

* 1. **СХЕМИ АЛГОРИТМІВ ОСНОВНИХ ФУНКЦІЙ**

Список підключених бібліотек:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <time.h>

Список використаних функцій:

int AgeCalcul(int dd, int mm, int yy) - *повертає вік(роки)*

Список використаних структур:

typedef struct

{

char name[15], lname[15], father[15];

} PIB;

typedef struct

{

int day, month, year,age;

} date;

typedef struct

{

char street[15], house[15];

int flat;

} adress;

typedef struct

{

PIB p;

date d;

adress a;

int notes[3];

} student;

* 1. **Висновок**

Під час роботи було використано багато даних структурованого типу. Напириклад, структура date має відомості про день, місяць і рік народження, та використовується в структурі student.

1. **Додаток(а)**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <time.h>

/\*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.

\*/

typedef struct

{

char name[15], lname[15], father[15];

} PIB;

typedef struct

{

int day, month, year,age;

} date;

typedef struct

{

char street[15], house[15];

int flat;

} adress;

typedef struct

{

PIB p;

date d;

adress a;

int notes[3];

} student;

int AgeCalcul(int dd, int mm, int yy)

{ time\_t t = time(NULL);

struct tm \*T = localtime(&t);

if ((T->tm\_mon+1) > mm)

return (T->tm\_year+1900) - yy;

else

{

if (mm == (T->tm\_mon+1))

if (T->tm\_mday >= dd)

return (T->tm\_year+1900) - yy;

else

return (T->tm\_year+1900) - yy - 1;

else

return (T->tm\_year+1900) - yy - 1;

}

}

int main()

{

student a[100];

int n,counter = 0;

printf("Enter num of students ");

scanf("%d", &n);

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

{

system("cls");

printf("student #%d", i + 1);

printf("\nName >");

fflush(stdin);

gets(a[i].p.name);

printf("\nLname >");

fflush(stdin);

gets(a[i].p.lname);

printf("\nday of birth >");

scanf("%d", &a[i].d.day);

printf("\nmonth of birth >");

scanf("%d", &a[i].d.month);

printf("\nyear of birth >");

scanf("%d", &a[i].d.year);

printf("\nstreet >");

fflush(stdin);

gets(a[i].a.street);

printf("\nhouse >");

fflush(stdin);

gets(a[i].a.house);

printf("\nflat >");

scanf("%d", &a[i].a.flat);

for (int j = 0; j < 3; j++)

{

printf("\n\tnote >");

scanf("%d", &a[i].notes[j]);

}

a[i].d.age = AgeCalcul(a[i].d.day,a[i].d.month,a[i].d.year);

}

char NAME[15],LNAME[15];

printf("Enter Name :");

scanf("%s",&NAME);

printf("Enter Lname :");

scanf("%s",&LNAME);

int f = 0;

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

{

if((strcmp(NAME,a[i].p.name)==0) && (strcmp(LNAME,a[i].p.lname)==0))

{

printf("\nYES\n");

f = 1;

}

if(a[i].d.age > 18) counter++;

}

if(f == 0) { printf("No student %s %s ", NAME,LNAME);}

printf("\nnum of students older than 18 y.o = %d\n",counter);

system("pause");

return 0;

}

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.

2) 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**

Під час написання 1 програми я використав однозв`язний лінійний список, хоча підозрював що написати його сортування на багато буде важче, ніж масив. Також я використав безкінечний цикл for(;;) для меню, яке також працює на операторі вибору switch (при натисненні клавіші “0” спрацьовував вихід з програми. Під час написання 2 програми, я вивів таблицю в 2 колонки з назвою товару і випадково згенерованою ціною дійного типу. Тут також використовувався безкінечний цикл з виходом (клавіша “0”). Але тут я запрошував id товару який порівнював.

* 1. **СХЕМИ АЛГОРИТМІВ ОСНОВНИХ ФУНКЦІЙ**

Список підключених бібліотек:

1:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

2:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <time.h>

Список використаних функцій:

1:

void SwapForwList\_BEG(Elem \*\*st, Elem \*\*bef) - *для обміну елементів списку на його початку*

void SwapForwList\_END(Elem \*\*before\_before\_End, Elem \*\*bef) - *для обміну елементів в кінці списку*

void SwapForwList\_CUR(Elem \*\*before\_first\_Cur, Elem \*\*bef) - *для обміну елементів списку в інших ділянках*

void push\_menu(char s[]) - *для виводу меню в консоль*

void push\_table(char s1[], char s2[], Elem \*\*st, Elem \*\*pt) - *для виводу в консоль таблиці учнів та їх зростом.*

void AddEnd(Elem \*\*st, Elem \*\*pt) - *для додавання елементів в кінець списку(а також початкового формування списку)*

2:

void push\_table(goods \*a,int n) - *для виводу в консоль таблиці товарів*

Список створених структур:

1:

typedef struct Tst

{

char lname[15];

int high;

struct Tst \*next;

} Elem;

2:

typedef struct

{

double price;

char name[30];

} goods;

* 1. **Висновок**

Було отримано 2 програми. Перша програма сортує список учнів за зростом, визначає після якого студента треба записати новенького для збереження сортування, визначає учня у якого ріст найближчий до новенького. Друга ж програма порівнює 2 товари.

1. **Додаток(а)(1)**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <time.h>

/\*5. 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

\*/

typedef struct Tst

{

char lname[15];

int high;

struct Tst \*next;

} Elem;

void SwapForwList\_BEG(Elem \*\*st, Elem \*\*bef)

{

Elem \*tmp;

tmp = (\*st)->next;

(\*st)->next = tmp->next;

tmp->next = \*st;

\*st = tmp;

\*bef = \*st;

}

void SwapForwList\_END(Elem \*\*before\_before\_End, Elem \*\*bef)

{

if ((\*before\_before\_End)->next->next == NULL)

{

(\*before\_before\_End) = (\*bef);

Elem \*tmp = (\*before\_before\_End)->next;

(\*before\_before\_End)->next->next->next = tmp;

(\*before\_before\_End)->next = tmp->next;

tmp->next = NULL;

}

else

{

Elem \*tmp = (\*before\_before\_End)->next;

(\*before\_before\_End)->next->next->next = tmp;

(\*before\_before\_End)->next = tmp->next;

tmp->next = NULL;

}

}

void SwapForwList\_CUR(Elem \*\*before\_first\_Cur, Elem \*\*bef)

{

(\*before\_first\_Cur) = (\*bef);

Elem \*tmp = (\*before\_first\_Cur)->next;

Elem \*forth\_ = tmp->next->next;

(\*before\_first\_Cur)->next->next->next = tmp;

(\*before\_first\_Cur)->next = tmp->next;

tmp->next = forth\_;

// if ((forth\_->next == NULL))

// \*bef = (\*before\_first\_Cur)->next;

}

void push\_menu(char s[])

{

char s1[] = "add student";

char s2[] = "sort";

char s3[] = "LN before new st";

char s5[] = "LN list < new\_high";

char s4[] = "push to file";

char stop[] = "clear and exit";

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

printf("\n|%-2d|%-18s|", 1, s);

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

printf("\n|%-2d|%-18s|", 2, s1);

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

printf("\n|%-2d|%-18s|", 3, s2);

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

printf("\n|%-2d|%-18s|", 4, s3);

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

printf("\n|%-2d|%-18s|", 5, s4);

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

printf("\n|%-2d|%-18s|", 6, s5);

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

printf("\n|%-2d|%-18s|", 0, stop);

printf("\n+--+------------------+\n"); //23

}

void push\_table(char s1[], char s2[], Elem \*\*st, Elem \*\*pt)

{

if (\*pt == \*st)

{

printf("\n\t+----+-----------------------------+"); //36

printf("\n\t|%-4s|%-29s|", s1, s2);

printf("\n\t+----+-----------------------------+");

}

printf("\n\t|%-4d|%-29s|", (\*pt)->high, (\*pt)->lname);

if ((\*pt)->next == NULL)

printf("\n\t+----+-----------------------------+");

}

void AddEnd(Elem \*\*st, Elem \*\*pt)

{

Elem \*dd = (Elem \*)malloc(sizeof(Elem));

printf("\n\tLname > ");

fflush(stdin);

gets(dd->lname);

printf("\n\tHigh(cm) > ");

scanf("%d", &dd->high);

if (\*st == NULL)

{

dd->next = NULL;

\*st = dd;

\*pt = dd;

}

else

{

dd->next = NULL;

(\*pt)->next = dd;

\*pt = dd;

}

}

int main()

{

int f = 0, new\_high;

FILE \*file = fopen("11\_1out.txt", "w");

int n = 5;

Elem \*st, \*pt;

st = NULL;

pt = NULL;

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

{

printf("#%d", i + 1);

AddEnd(&st, &pt);

}

int k = 0, p = 0;

pt = st;

for (;;)

{

m1:

if (p == 1)

break;

system("cls");

push\_menu("show table");

scanf("%d", &k);

switch (k)

{

case 1:

{

pt = st;

while (pt != NULL)

{

push\_table("high", " Last name ", &st, &pt);

pt = pt->next;

}

getch();

break;

}

case 2:

{

if (f == 0)

{

printf("not sorted! Please sort first");

getch();

break;

}

pt = st;

while (pt != NULL)

{

if (pt->next == NULL)

{

AddEnd(&st, &pt);

new\_high = pt->high;

}

pt = pt->next;

}

f = 0;

n++;

break;

}

case 3:

{

f = 1;

pt = st;

Elem \*bef = NULL;

for (int p = 0; p < 12; p++)

{

for (int i = n - 1; i > 0; i--)

{

pt = st;

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

{

if (pt->high < pt->next->high)

{

if (pt == st)

SwapForwList\_BEG(&st, &bef);

else if (pt->next->next == NULL)

SwapForwList\_END(&pt, &bef);

else

SwapForwList\_CUR(&pt, &bef);

}

else

{

bef = pt;

pt = pt->next;

}

}

}

}

printf("sorted");

getch();

break;

}

case 4:

{

if (f == 0)

{

printf("not sorted! Please sort first");

getch();

break;

}

pt = st;

while (pt != NULL)

{

if (pt->next->high == new\_high)

{

if (abs(pt->high - new\_high) < abs(pt->next->next->high - new\_high))

{

printf("\n3) %s , %d", pt->lname, pt->high);

fprintf(file, "\n3) %s, %d", pt->lname, pt->high);

}

else

{

printf("\n3) %s, %d", pt->next->next->lname, pt->next->next->high);

fprintf(file, "\n3) %s, %d", pt->next->next->lname, pt->next->next->high);

}

printf("\n2) %s", pt->lname);

fprintf(file, "\n2) %s", pt->lname);

getch();

break;

}

pt = pt->next;

}

break;

}

case 5:

{

pt = st;

while (pt != NULL)

{

if (pt == st)

{

fprintf(file, "\n\t+--------+-----------------------------------+"); //44

fprintf(file, "\n\t| number | type |"); //44

fprintf(file, "\n\t+--------+-----------------------------------+"); //44

}

fprintf(file, "\n\t|%-8d|%-35s|", pt->high, pt->lname); //44

if (pt->next == NULL)

fprintf(file, "\n\t+--------+-----------------------------------+"); //44

pt = pt->next;

}

printf("Loaded\n");

fclose(file);

getch();

break;

}

case 6:

{

pt = st;

while (pt != NULL)

{

if (pt->high < new\_high)

{

printf("\n1)\n");

printf("%s, %d",pt->lname, pt->high );

fprintf(file,"\n1)\n");

fprintf(file,"%s, %d",pt->lname, pt->high );

}

pt = pt->next;

}

getch();

break;

}

case 0:

{

p = 1;

system("pause");

break;

}

default:

printf("\nError");

goto m1;

break;

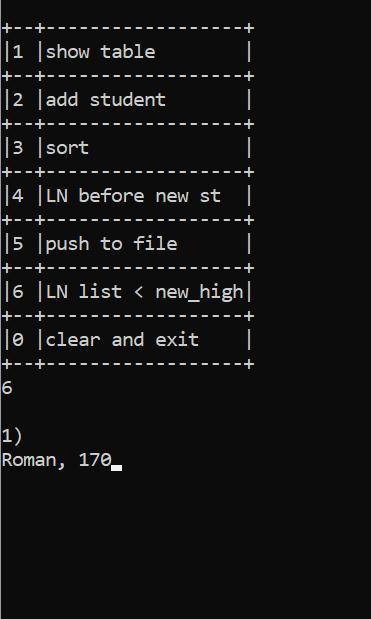
}

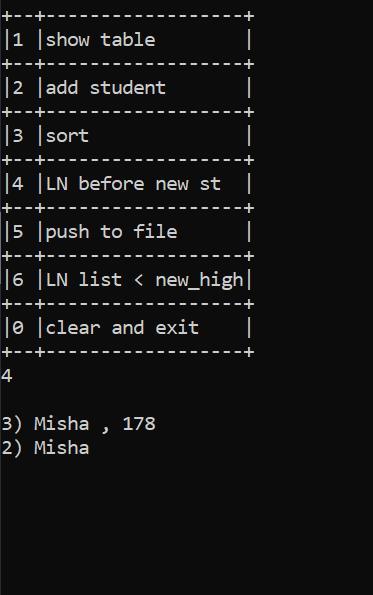
}

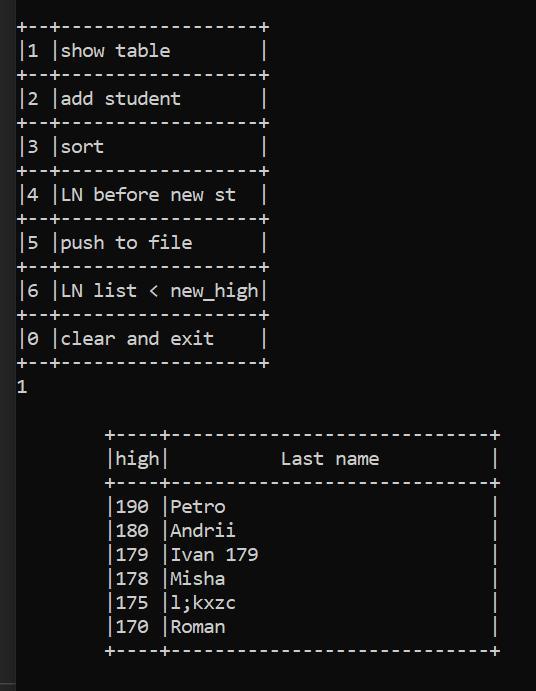
return 0;

}

1. **Додаток(б)(1)**

****

****

****

1. **Додаток(а)(2)**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <time.h>

typedef struct

{

double price;

char name[30];

} goods;

void push\_table(goods \*a,int n)

{

char s1[] = " name ";

char s2[] = "priece";

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

{

if (i == 0)

{

printf("\n\t+--+--------------+-------+ "); printf("+--+--------------+-------+ ");//24

printf("\n\t|# |%-14s|%-7s| ", s1, s2); printf("|# |%-14s|%-7s| ", s1, s2);

printf("\n\t+--+--------------+-------+ "); printf("+--+--------------+-------+ ");//24

}

printf("\n\t|%-2d|%-14s|%.2lf | ", i+1,a[i].name,a[i].price);printf("|%-2d|%-14s|%.2lf |", i+2,a[i+1].name,a[i+1].price);

printf("\n\t+--+--------------+-------+ "); printf("+--+--------------+-------+ ");//24

}

}

int main()

{

goods a[20];

const int n = 20;

srand(time(NULL));

strcpy(a[0].name, "good #1");

strcpy(a[1].name, "good #2");

strcpy(a[2].name, "good #3");

strcpy(a[3].name, "good #4");

strcpy(a[4].name, "good #5");

strcpy(a[5].name, "good #6");

strcpy(a[6].name, "good #7");

strcpy(a[7].name, "good #8");

strcpy(a[8].name, "good #9");

strcpy(a[9].name, "good #10");

strcpy(a[10].name, "good #11");

strcpy(a[11].name, "good #12");

strcpy(a[12].name, "good #13");

strcpy(a[13].name, "good #14");

strcpy(a[14].name, "good #15");

strcpy(a[15].name, "good #16");

strcpy(a[16].name, "good #17");

strcpy(a[17].name, "good #18");

strcpy(a[18].name, "good #19");

strcpy(a[19].name, "good #20");

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

{

a[i].price = (double)(rand()%100+1)/10;

}

for(;;)

{

system("cls");

push\_table(a,n);\

double p1,p2;

char s1[10],s2[10];

int k = -1, k2 = -1;

printf("\nchoose good for compare ");

scanf("%d",&k);

p1=a[k-1].price;

strcpy(s1,a[k-1].name);

printf("\nchoose good for compare ");

scanf("%d",&k2);

p2=a[k2-1].price;

strcpy(s2,a[k2-1].name);

if(p1>p2)

printf("%s (%d) > %s (%d)",s1,p1,s2,p2);

else if (p1==p2)

printf("%s (%d) = %s (%d)",s1,p1,s2,p2);

else printf("%s (%.2lf) < %s (%.2lf)",s1,p1,s2,p2);

getch();

if(k == 0) break;

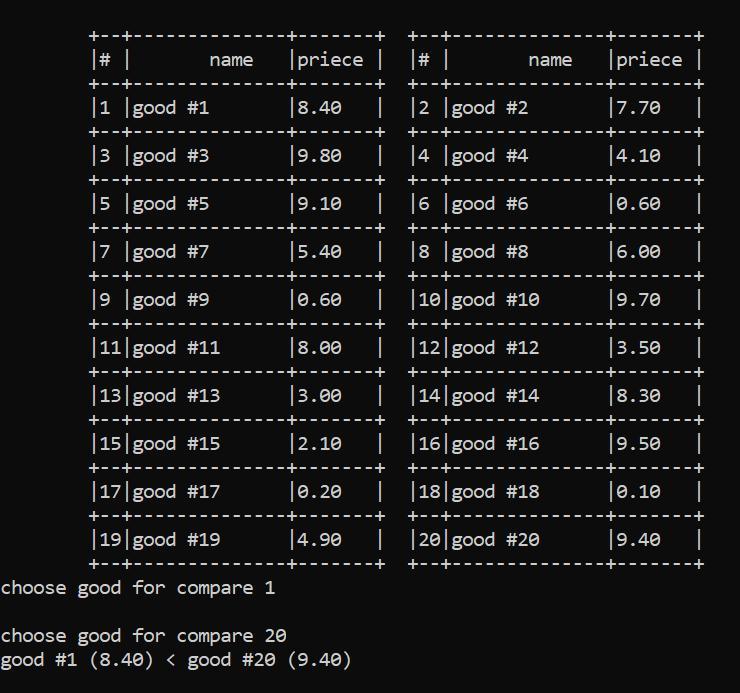
}

system("pause");

return 0;

}

1. **Додаток(б)(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. **СХЕМИ АЛГОРИТМІВ ОСНОВНИХ ФУНКЦІЙ**

Список підключених бібліотек:

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

Список використаних функцій:

void AddEnd(Elem \*\*st, Elem \*\*pt, int ran) - *додає елемент в кінець списку з значенням ran,*

void AddEnd\_\_( Elem \*\*st, Elem \*\*pt, Elem \*\*source) - *додає елемент в кінець списку зі значенням (\*source)->num.*

Список використаних структур:

typedef struct El1

{

int num;

struct El1 \*next;

} Elem;

* 1. **Висновок**

В результаті ми отримали програму, яка з одного лінійного одноз`вязного списку формує ще 2 в яких знаходяться лише позитивні та лише негативні значення. Також було вдосконалено навички роботи зі списками та показниками.

* 1. **Додаток(а)**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

/\*5. 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.\*/

typedef struct El1

{

int num;

struct El1 \*next;

} Elem;

void AddEnd(Elem \*\*st, Elem \*\*pt, int ran)

{

Elem \*dd = (Elem \*)malloc(sizeof(Elem));

dd->num = ran;

if (\*st == NULL)

{

dd->next = NULL;

\*st = dd;

\*pt = dd;

}

else

{

dd->next = NULL;

(\*pt)->next = dd;

\*pt = dd;

}

}

void AddEnd\_\_( Elem \*\*st, Elem \*\*pt, Elem \*\*source)

{

Elem \*dd = (Elem\*)malloc(sizeof(Elem));

dd ->num = (\*source)->num;

if ( \*st == NULL)

{

dd->next = NULL;

\*st = dd;

\*pt = dd;

}

else

{

dd->next = NULL;

(\*pt)->next = dd;

\*pt = dd;

}

}

int main()

{

Elem \*st = NULL, \*pt = NULL, \*st\_pos = NULL, \*st\_neg = NULL, \*pt\_pos = NULL, \*pt\_neg = NULL;

const int n = 5;

srand(time(NULL));

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

{int ran;

ran = rand()%20-10;

AddEnd(&st, &pt,ran);

}

pt = st;

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

{

if (pt->num > 0)

{

AddEnd\_\_(&st\_pos, & pt\_pos,&pt);

}

if (pt->num < 0)

{

AddEnd\_\_(&st\_neg, &pt\_neg,&pt);

}

pt = pt->next;

}

pt = st;

pt\_neg = st\_neg;

pt\_pos = st\_pos;

printf("list 1\n");

while (pt !=NULL)

{

printf("%d ",pt->num);

pt = pt->next;

}

printf("\nlist 2\n");

while (pt\_pos !=NULL)

{

printf("%d ",pt\_pos->num);

pt\_pos = pt\_pos->next;

}

printf("\nlist 3\n");

while (pt\_neg !=NULL)

{

printf("%d ",pt\_neg->num);

pt\_neg = pt\_neg->next;

}

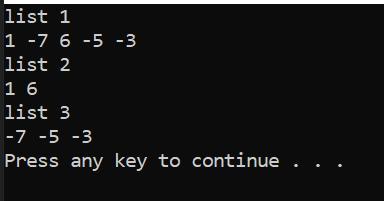
printf("\n");

system("pause");

return 0;

}

* 1. **Додаток(б)**

****

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

Варіант 5

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

2. 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**

Під час написання першої програми я використав fprintf для запису у файл елементів. Після закриття файлу, відкрив для читання з перевіркою(перевірка на те що fopen() != NULL є завжди). В циклі з зчитуванням fscanf() розміщена перевірка чи менше число значення D і якщо так то збільшуємо на одиницю значення counter. В другій програмі за допомогою моєї функції визначаємо першу букву(навіть якщо на початку не буква, функція знайде букву. І перезапише весь масив символів, крім наступних входжень першої літери.

* 1. **СХЕМИ АЛГОРИТМІВ ОСНОВНИХ ФУНКЦІЙ**

Список підключених бібліотек:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

Список використаних функцій:

2:

char Letter\_def(char s) - *повертає першу букву*

* 1. **Висновок**

В результаті було отримано 2 програми. Перша програма зчитує з файлу числа і порівнює їх з D. Друга ж видаляє всі наступні входження першої літери.

* 1. **Додаток(а)(1)**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

/\*5. Створіть файл дійсних чисел. Підрахуйте кількість елементів файлу, більших за n

(кількість елементів файлу, n та самі елементи вводяться з клавіатури).\*/

int main (){

float \*a;

FILE \*file;

int n,d;

printf("Enter n & d\n");

scanf("%d%d",&n,&d);

file = fopen("D:\\work\\Progrr\\LP\\task\_13\\data.txt","w");

if(file == NULL) exit(1);

else

{

a = (float \*)malloc(n\*sizeof(float));

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

{

printf("%d > ", i+1);

scanf("%f",&a[i]);

}

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

{

fprintf(file,"%.2f ", a[i]);

}

fclose(file);

int counter=0;

file = fopen("D:\\work\\Progrr\\LP\\task\_13\\data.txt","r");

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

{

fscanf(file,"%f",&a[i]);

if(a[i]<d) counter++;

// printf("%f ", a[i]);

}

fclose(file);

printf("\nin file num < %d = %d\n", d, counter);

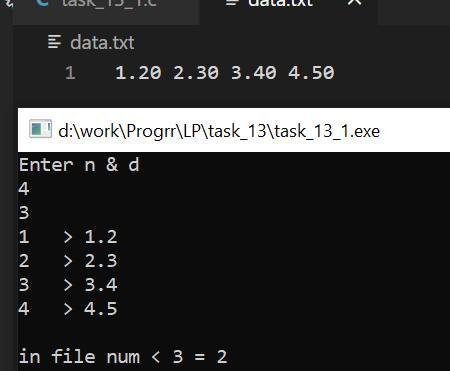
}

system("pause");

return 0;

}

* 1. **Додаток(б)(1)**

****

* 1. **Додаток(а)(2)**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

/\*Задано текст, слова в якому розділені пробілами і розділовими знаками. Розробити програму, яка вилучає в кожному

слові цього тексту всі наступні входження першої літери. \*/

char Letter\_def(char s)

{

if ((s < 'A') || ((s > 'Z') && (s < 'a')) || (s > 'z'))

return 1;

else if (s >= 'A' && s <= 'Z')

return s + 32;

else

return s;

}

int main()

{

char s[100];

printf("Text :\n");

fflush(stdin);

gets(s);

char buf = 1, buf\_capital = 1; int i = 0;

while (buf == 1)

{

buf =Letter\_def(s[i]);

i++;

}

char \*b;

b = (char\*)malloc(strlen(s)\*sizeof(char));

int k =0;

buf\_capital = buf - 32;

int counter = 0;

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

{

if((s[i] == buf) || (s[i] == buf\_capital))

{

counter++;

}

else

{

b[k++] = s[i];

}

}

b = (char\*)realloc(b,(k+1)\*sizeof(char));

b[k+1] = '\0';

strcpy(s,b);

s[k] = '\0';

printf("\n");

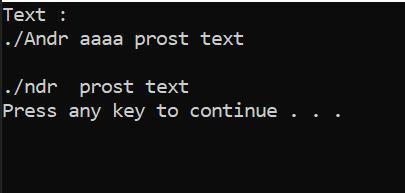
puts(s);

system("pause");

return 0;

}

* 1. **Додаток(б)(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.Розробити програми мовою С для розв’язання задачі згідно варіанту. Вхідні дані читаються з файлу, результат записується у новий файл. Ім’я файлів передаються через командний рядок, або вводяться з консолі.

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:

друге найменше за довжиною слово в порядку розташування, якщо їх декілька

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 ідентифікатори;

* 1. **ТЕОРЕТИЧНІ ВІДОМОСТІ**

В першій програмі я скористався тим, що потрібно зробити 2 перевірки: перша на кількість відкритих і закритих дужок і друга на їх правильний баланс( щоб не було такого “)(” ). Для цього є два окремих цикла for. В другій програмі використовується зчитування рядка і же з рядком проводяться маніпуляції. В третій програмі ми вводимо ідентифікатор а програма виводить всі слова в яких він зустрічається(слова записані в файл).

* 1. **СХЕМИ АЛГОРИТМІВ ОСНОВНИХ ФУНКЦІЙ**

Список підключених бібліотек:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

Список використаних функцій:

2:

int min\_2Length(char \*s) - *для визначення довжини 2 за мінімальністю слова*

int IS\_Letter(char s) - *для перевірки симовлу s чи він є буквою(англ алф)*

3:

int IS\_Letter(char s) - *для перевірки симовлу s чи він є буквою(англ алф)*

* 1. **Висновок**

Було покращено вміння роботи з символьними рядками та файлами.

* 1. **Додаток(а)(1)**

#include <conio.h>

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

// #include <time.h>

/\*5. 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

\*/

int main()

{

char s[100];

printf("Text:\n");

gets(s);

int counter\_open = 0, counter\_cl = 0;

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

{

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

s[i] == '}' ||

s[i] == ']')

counter\_cl++;

if (s[i] == '(' ||

s[i] == '{' ||

s[i] == '[')

counter\_open++;

}

int f = 0;

if (counter\_open != counter\_cl)

f = 1;

int l1 = 0, l2 = 0;

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

{

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

s[i] == '}' ||

s[i] == ']')

l1++;

if (l1 > l2)

{

printf("-\_- NO \")\" > na %d", counter\_cl - counter\_open);

f = 0;

break;

}

else if (l2 == l1 && i == strlen(s) - 1)

printf("Balanced");

if (s[i] == '(' ||

s[i] == '{' ||

s[i] == '[')

l2++;

}

if (f == 1)

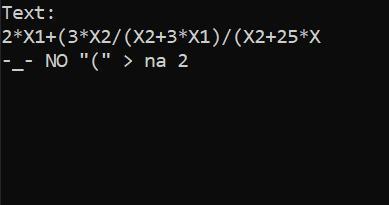
printf("-\_- NO \"(\" > na %d", counter\_open - counter\_cl );

getch();

return 0;

}

* 1. **Додаток(б)(1)**

****

* 1. **Додаток(а)(2)**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

/\*5) друге найменше за довжиною слово в порядку розташування, якщо їх декілька;\*/

int min\_2Length(char \*s)

{

int counter = 0, min = strlen(s), min\_2 = strlen(s);

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

{

if (((s[i] >= 'A') && (s[i] <= 'Z')) || ((s[i] >= 'a') && (s[i] <= 'z')))

{

counter++;

}

else if(counter!=0)

{

if (counter < min)

{

min\_2 = min;

min = counter;

}

if (counter < min\_2 && counter > min)

min\_2 = counter;

counter = 0;

}

}

if(counter!=0)

{

if (counter < min)

{

min\_2 = min;

min = counter;

}

if (counter < min\_2 && counter > min)

min\_2 = counter;

counter = 0;

}

return min\_2;

}

int IS\_Letter(char s)

{

if (((s >= 'A') && (s <= 'Z')) || ((s >= 'a') && (s <= 'z')))

{

return 0;

}

else

{

return 1;

}

}

int main()

{

FILE \*in\_file, \*out\_file;

char s[100];

in\_file = fopen("in.txt", "r");

out\_file = fopen("out.txt", "w");

if (in\_file == NULL)

exit(1);

else

{

if (out\_file == NULL)

{

exit(1);

}

else

{

while (!feof(in\_file))

{

int min\_2\_len = 0, k = 0;

fgets(s, 100, in\_file);

min\_2\_len = min\_2Length(s);

char word[min\_2\_len];

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

{

if (IS\_Letter(s[i]) == 0)

{

word[k++] = s[i];

}

else if (k == min\_2\_len)

{

for (int j = 0; j < min\_2\_len; j++)

{

fprintf(out\_file,"%c",word[j]);

}

fprintf(out\_file," ");

k = 0;

}

else

{

k = 0;

}

}

fprintf(out\_file,"\n");

}

}

}

fclose(in\_file);

fclose(out\_file);

system("pause");

return 0;

}

* 1. **Додаток(б)(2)**

****

* 1. **Додаток(а)(3)**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

/\*

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

5) ідентифікатори;

\*/

int IS\_Letter(char s)

{

if (((s >= 'A') && (s <= 'Z')) || ((s >= 'a') && (s <= 'z')))

{

return 0;

}

else

{

return 1;

}

}

int main()

{

FILE \*in\_file = fopen("task\_14\_3in.txt", "r");

if (in\_file == NULL)

exit(1);

else

{

char ident[100];

printf("Enter ident\n");

fflush(stdin);

gets(ident);

char s[100];

int k = 0;

int j, g = 0, f = 0, p = 0;

while (!feof(in\_file))

{

fgets(s, 100, in\_file);

k =0;

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

{

if (IS\_Letter(s[i]) == 1)

{

k = i + 1;

}

else

{

if (s[i] == ident[0])

{

p = 0;

j = i;

while (g < strlen(ident))

{

if (s[j] == ident[g])

p = 1;

else

{

p = 0;

break;

}

j++;

g++;

}

g = 0;

if (p == 1)

{

while (IS\_Letter(s[k]) == 0)

{

printf("%c", s[k]);

k++;

}

i = k - 1;

printf(" ");

}

}

}

}

printf("\n");

}

}

system("pause");

return 0;

}

* 1. **Додаток(б)(3)**

