Міністерство освіти і науки України

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Алгоритми та структури даних. Частина1.

Основи алгоритмізації.

Лабораторний практикум

Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського
як навчальний посібник для здобувачів ступеня бакалавра
за освітньою програмою «Інженерія програмного забезпечення інформаційних систем»
спеціальності 121 «Інженерія програмного забезпечення»

Укладач: І.І. Вітковська

Електронне мережеве навчальне видання

Київ
КПІ ім. Ігоря Сікорського
2024

УДК

Укладач: Вітковська Ірина Іванівна, старший викладач

Рецензент Солдатова М.О. канд. техн. наук, доцент, доцент кафедри інформаційних систем та технологій НТУУ «КПІ ім. Ігоря Сікорського»

Відповідальний редактор Ліщук К.І., к.т.н, доцент, доцент кафедри інформатики та програмної інженерії НТУУ «КПІ ім. Ігоря Сікорського»

Гриф надано Методичною радою КПІ ім. Ігоря Сікорського
(протокол № 1 від 26.09.2024 р.)
за поданням вченої ради факультету інформатики та обчислювальної техніки
(протокол № 1 від 29.08.2024 р.)

ХХХ **Алгоритми та структури даних. Частина 1**. Основи алгоритмізації [Електронний ресурс] : лаб. практикум : навч. посіб. для здобувачів ступеня бакалавра за освіт. програмою «Інженерія програмного забезпечення інформаційних систем» спец. 121 «Інженерія програмного забезпечення»/ КПІ ім. Ігоря Сікорського ; уклад.:І.І.Вітковська. – Електрон. текст. дані (1 файл). – Київ : КПІ ім. Ігоря Сікорського, 2024. – 101 с.

Навчальне електронне видання містить завдання до виконання лабораторних робіт, варіанти, вказівки щодо виконання та оформлення звіту, приклади виконання та контрольні питання для підготовки до захисту лабораторних робіт. Призначене для студентів спеціальності 121 «Інженерія програмного забезпечення».

УДК ХХХ.ХХ (ХХХ)

Реєстр. № НП 24/25-030. Обсяг 2,1 авт. арк.

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
проспект Берестейський, 37, м. Київ, 03056
https://kpi.ua
Свідоцтво про внесення до Державного реєстру видавців, виготовлювачів і розповсюджувачів видавничої продукції ДК № 5354 від 25.05.2017 р.

ЗМІСТ

ВСТУП

Дисципліна «Алгоритми та структури даних. Частина 1. Основи алгоритмізації» є базовою спеціальною нормативною дисципліною підготовки фахівців рівня бакалавр спеціальності 121 ««Інженерія програмного забезпечення».

Мета лабораторних занять – практичне підтвердження теоретичних положень дисципліни «Алгоритми та структури даних. Частина 1. Основи алгоритмізації», формування у студентів здатності створювати програмне забезпечення на базі алгоритмів з обробки структур даних, використовувати математичні та комбінаторні алгоритми для вирішення поставлених задач, застосовувати обчислювальні алгоритми при розробці програмного забезпечення автоматизованих систем.

Лабораторні заняття полягають у побудові студентами специфікацій програм, спрямованих на вирішення простих типових завдань програмування, та написанні програм процедурними мовами програмування, керуючись принципами структурного проектування та програмування.

Кожна лабораторна робота виконується особисто студентом згідно з індивідуальним завданням за варіантом під керівництвом викладача. Номер варіанта задається викладачем до кожної лабораторної роботи.

На захисті лабораторної роботи студент повинен показати результати роботи, оформити звіт і відповісти на додаткові запитання. Виконанню лабораторної роботи повинна передувати самостійна підготовка. В процесі якої необхідно вивчити відповідні алгоритми, конструкції мов програмування, виконати завдання, створити звіт та зберегти проект на хмарне сховище Github, Google Class або Google диск.

Лабораторна робота 1
ЛІНІЙНІ АЛГОРИТМИ

Мета – дослідити лінійні програмні специфікації для подання перетворювальних операторів та операторів суперпозиції, набути практичних навичок їх використання під час складання лінійних програмних специфікацій.

Основні теоретичні відомості

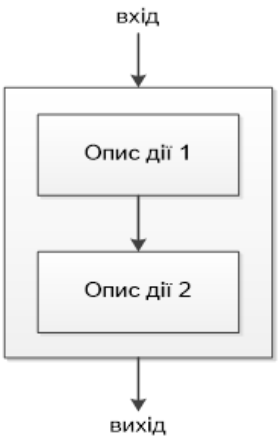
Лінійним прийнято називати обчислювальний процес, операції якого виконуються послідовно, в порядку їх запису. Кожна операція є самостійною й не залежить від будь-якої вимоги.

Лінійна структура – це найпростіший тип алгоритму, в процесі виконання якого відтворюється послідовне виконання блоків алгоритму. Лінійний обчислювальний процес характеризується тим, що кроки, на які він розбивається, виконуються послідовно в тому порядку, в якому вони подані.

Для лінійного обчислювального процесу характерним є наявність двох типів операторів: перетворювальних та суперпозицій (з'єднання перетворювальних дій). У блок-схемах для опису перетворювальних операторів використовується прямокутник, у середині якого розміщується текст з описом дії (рис.1, *a*).

Для опису операторів суперпозиції використовується послідовність прямокутників, з'єднаних лініями (послідовність перетворювальних операторів) (рис.1, *b*).

За допомогою лінійних алгоритмів вирішуються прості задачі, в яких число дій дорівнює числу блоків схеми алгоритму. Лінійні обчислювальні процеси застосовуються під час обчислення арифметичних виразів.



a) *b*)

Рисунок 1 - Опис перетворювального оператора (*a*) та оператора суперпозиції (*b*) у блок-схемах

Приклад застосування перетворювальних операторів та операторів суперпозиції під час складання програмних специфікацій

Приклад. Задано два значення A і B. Знайти їх суму Sum і добуток Mul.

Розв'язання

Умова задачі не передбачає перевірки будь-яких умов або повторення виконання операцій, тому обчислюваний процес для розв'язання задачі є лінійним. Програмну специфікацію напишемо у псевдокоді та графічній формі у вигляді блок-схеми.

Крок 1. Визначимо основні дії.

Крок 2. Деталізуємо дію суми.

Крок 3. Деталізуємо дію множення.

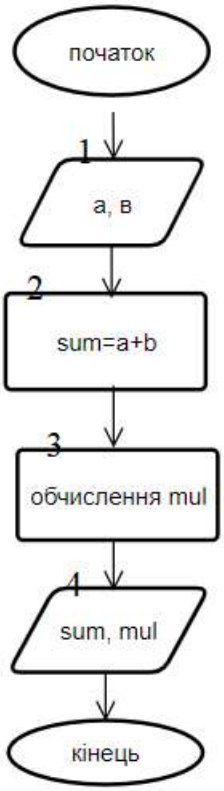
Псевдокод

крок 1	*крок 2*	*крок 3*
початок	**початок**	**початок**
1.ввід a,в	1.ввід a,в	1.ввід a,в
2. обчислення суми Sum	2.Sum := A + B	2. Sum := A + B
3. обчислення добутку Mul	3. обчислення добутку Mul	3.Mult := A * B
4. вивід Sum, Mul	4.вивід Sum, Mul	4.вивід Sum, Mul
кінець	**кінець**	**кінець**

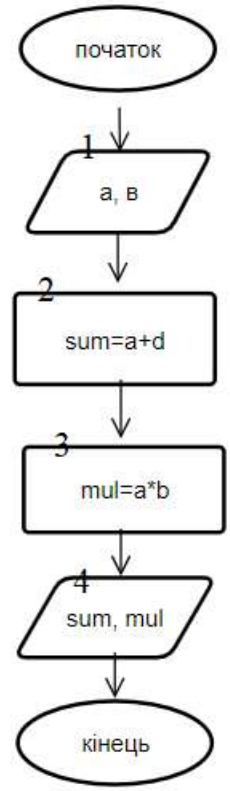
Блок-схема

крок1

крок 2



крок 3



Перевіримо правильність алгоритму

Таблиця 1. Тестування алгоритму

Блок	Приклад 1	Приклад 2	Приклад 3
Початок			
1.Ініціалізація a, в	3 5	4 -6	-3 -9
2. Обчислення sum	8	-2	-12
3.Обчислення mul	15	-24	27
4. Вивід	8 15	-2 -24	-12 27
Кінець			

Код мовою С.

```c
//Задано два значення A і B. Знайти їх суму Sum і добуток Mul.
#include <iostream>
    using namespace std;
    int main()
 {
    int a,
        b,
        sum,
        mul;

    cin >> a >> b;

    sum = a + b;
    mul = a * b;

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

    return 0;
}
```

Індивідуальне завдання

1. Виконати завдання згідно варіанту (табл.2)
2. Декомпозувати задачу, описати постановку.
3. Програмну специфікацію навести у псевдокоді та графічній формі.
4. Перевірити правильність алгоритму.
5. Код навести мовою С.

Таблиця 2. Варіанти завдань

№	Опис завдання
1	Задано два значення A і B. Знайти Y= $\sqrt{\lvert x\rvert + 1}$ + x∗ ln x, де x=2*a+2*|b|
2	Задано два значення A і B. Знайти Y= 3*x+5, де x= $\frac{a+b-\lvert a-b\rvert}{4}$
3	Задано значення A. Знайти Y = $\frac{5}{\lvert 5*X-3\rvert+8}$, де x= 9∗ ln a
4	Задано два значення A і B. Знайти Y = $\frac{X-7}{\cos^2 x}$, де x=|7-a+b|

№	Опис завдання
5	Задано координати двох протилежних вершин прямокутника: (x_1, y_1), (x_2, y_2). Сторони прямокутника паралельні осям координат. Знайти периметр і площу даного прямокутника.
6	Задано значення A, X і C. Знайти $y = \frac{cx^2 + mb^3 + \ln x + 7}{b + m + \sqrt{c}}$ де m=2*x+√x, b=a*c^2 – m
7	Задано два значення A і B. Знайти Y = $\sqrt{\lvert x^2 - 3 * x\rvert + 8}$, де x = 2*|b|+a
8	Задано значення C. Знайти $s = \sqrt{a + \cos(a + b) - b^2}$, де b = a+ sin c, a = c + 2
9	Задано координати трьох вершин трикутника: (x_1, y_1), (x_2, y_2), (x_3, y_3). Знайти його периметр, використовуючи формулу для відстані між двома точками на площині.
10	Задано два значення A і B. Знайти Y = $\sqrt{\lvert x - 3\rvert + \lvert x - 8\rvert}$, де x = $\sin(b + a)$
11	Задано довжини ребер A, B, C прямокутного паралелепіпеда. Знайти його об'єм та площу поверхні.
12	Змішано v1 літрів води температури t1 з v2 літрами води температури t2. Знайти об'єм і температуру суміші.
13	Задано два ненульових числа. Знайти суму, різницю, добуток і частку їх квадратів.
14	Дано гіпотенуза і катет прямокутного трикутника. Знайти другий катет і радіус вписаного кола.
15	Задано чотирицифрове натуральне число. Знайти різницю добутку його крайніх цифр та добутку внутрішніх цифр.
16	Відомо значення температури за шкалою Цельсія. Знайти відповідне значення температури за шкалою Фаренгейта, Кельвіна.

№	Опис завдання
17	Швидкість першого автомобіля V_1 км / год, другого - V_2 км / год, відстань між ними S км. Визначити відстань між ними через T годин, якщо вони рухаються в одному напрямку.
18	Задана площа ділянки в гектарах. Вивести площу ділянки спочатку в метрах квадратних, а потім кілометрах квадратних.
19	Користувач вводить два трицифрові числа. Утворити з них чотирицифрове число за правилом: 1 цифра = остання цифра першого числа, 2 цифра = остання цифра другого числа, 3 цифра = 1 цифра першого числа, 4 цифра = 1 цифра другого числа.
20	Здійснити перерахунок величини тимчасового інтервалу, заданого в хвилинах, в величину, виражену в годинах і хвилинах.
21	Задано два значення A і B. Знайти $Y = 3x^2 + 5x + 5$, де $x = \frac{\lvert a-b+5\rvert}{4}$
22	Задано два значення A і B. Знайти $Y = \frac{a}{\cos(x+4)}$, де x=7+3b
23	Задано два значення A і B. Знайти $Y = x^3 + 5x + b$, де $X = \frac{2b+3}{4}$
24	Металічний куб з довжиною ребра a переплавили в кулю. Визначити радіус R кулі, враховуючи, що об'єм кулі $V = \frac{4}{3}\pi R^3$, а об'єм куба $V = a^3$.
25	Трикутник задано координатами своїх вершин. Знайти периметр та площу трикутника.
26	Трикутник задано довжинами сторін. Знайти довжини бісектрис та радіуси вписаного та описаного кіл.
27	Задано тризначне число. У ньому закреслили першу зліва цифру і приписали її в кінці. Знайти отримане число.
28	Задано тризначне число. У ньому закреслили останню справа цифру і приписали її на початку. Знайти отримане число.

№	Опис завдання
29	Задано тризначне число. У ньому закреслили другу зліва цифру і приписали її в кінці. Знайти отримане число.
30	Задано тризначне число. У ньому закреслили другу справа цифру і приписали її на початку. Знайти отримане число.
31	Задано сторону квадрата. Знайти його діагональ, периметр та площу.
32	Задано довжину ребра куба. Знайти об'єм куба і площу його бічної поверхні.
33	Задано радіус кола. Знайти довжину кола і площу круга.
34	Задано значення A. Знайти Y = $\sqrt{x^4 + x^2} + 8$, де x = 2 * b + a * b, b = 6 * a^2
35	Виріб збирають з трьох вузлів собівартістю A, B та C грн. відповідно. На оплату складання одного виробу затрачають X грн. Визначити ціну виробу, якщо від його продажу планується отримання прибутку P%.

Вимоги до звіту

Звіт повинен містити:

- титульний аркуш (додаток 1);
- назву та мету роботи;
- варіант;
- постановку задачі;
- побудову математичної моделі;
- псевдокод алгоритму;
- блок схема алгоритму;
- випробування алгоритму;
- код мовою C;
- висновки.

Критерії оцінювання

Критерії оцінювання в процентах від максимального балу:

постановка задачі - 10%;

побудова математичної моделі – 10%

псевдокод алгоритму - 25%;

блок схема алгоритму – 25%

випробування алгоритму - 5%;

код програми – 20%;

висновки - 5%.

Контрольні питання

1. Що таке програмна специфікація?
2. На які питання потрібно дати відповіді, перш ніж почати складати програмну специфікацію?
3. Який обчислюваний процес називається лінійним?
4. Які блоки використовують у блок-схемах лінійного обчислюваного процесу?
5. Охарактеризувати принципи структурного програмування.

Лабораторна робота 2
АЛГОРИТМИ РОЗГАЛУЖЕННЯ

Мета – дослідити подання керувальної дії чергування у вигляді умовної та альтернативної форм та набути практичних навичок їх використання під час складання програмних специфікацій.

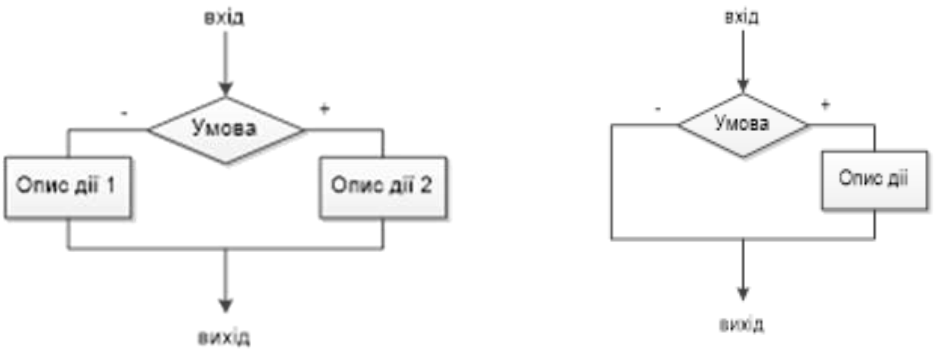
Основні теоретичні відомості

Обчислювальний процес називається розгалуженим, якщо для його реалізації передбачено кілька напрямів (гілок). Кожен окремий напрям процесу обробки даних є окремою гілкою обчислень. Обрання напряму залежить від заздалегідь визначеної ознаки, що може стосуватися первинних даних, проміжних або кінцевих результатів. Ознака характеризує властивість даних і має два або декілька значень. Напрям обирається за допомогою перевірки логічного виразу, результатом якого є дві відповіді: «так» або «ні». Відомі три форми оператора вибору: умовна, альтернативна, охоронна.

В блок-схемах для опису альтернативної форми оператора вибору використовується ромб, в якому розташовується умова, та два прямокутники, що описують дії. Оператор виконує тільки одну з дій: дію 2, якщо умова правильна, або дію 1, якщо умова неправильна (рис.2, *a*).

Умовна форма оператора вибору виконує тільки одну дію в разі, якщо умова правильна (рис.2, *b*).

Охоронна форма є узагальненням альтернативної форми вибору, це є множинний вибір. С точки зору структурного програмування вважається, що в використанні охоронної форми немає необхідності.



a) *b)*

Рисунок 2 - Опис оператора вибору альтернативної (*a*) та умовної (*b*) форм у блок-схемах

Приклади застосування альтернативної форми оператора вибору під час складання програмних специфікацій

Приклад 1. Задано три нерівних значення A, B і C. Знайти мінімальне значення.

Програмні специфікації напишемо у псевдокоді та графічній формі у вигляді блок-схеми.

Крок 1. Визначимо основні дії.

Крок 2. Деталізуємо дію знаходження мінімуму з використанням керувальної дії чергування.

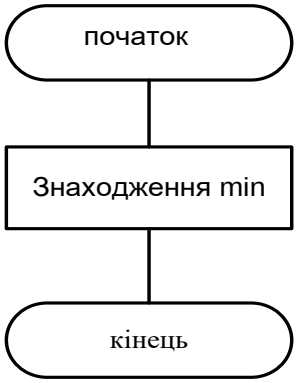
Псевдокод

крок 1
початок
знаходження min
кінець

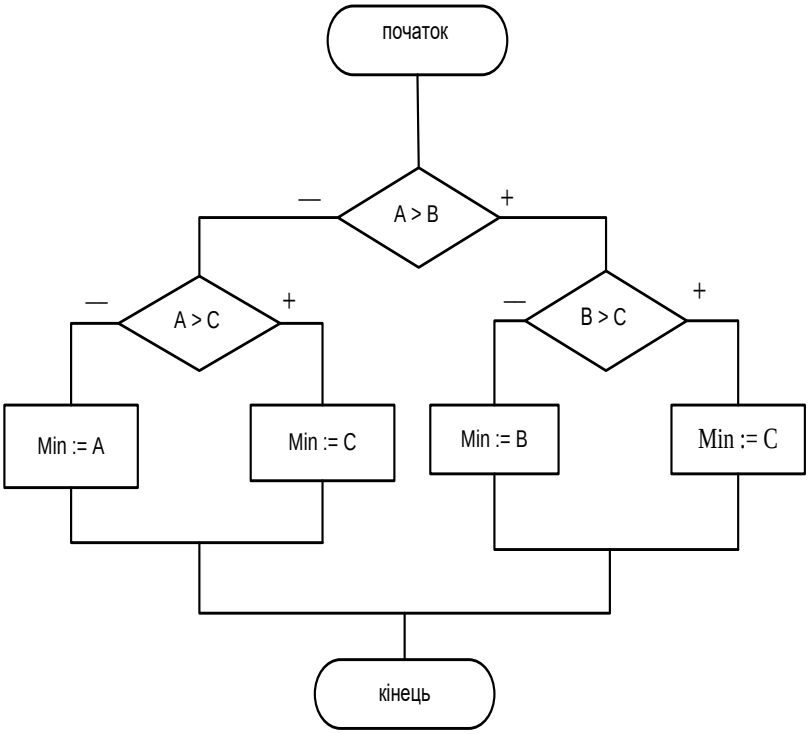
крок 2
```
початок
  якщо A > C
     то
       якщо B > C
        то
          min := C
        інакше
          min  := B
       все якщо
     інакше
        якщо A > C
        то
          min := A
        інакше
          min := C
        все якщо
    все якщо
кінець
```

Блок-схема

крок 1



крок 2



Перевірка правильності роботи алгоритму

Таблиця 3. Тестування алгоритму

A	B	C	MIN
5	8	1	1
4	12	22	4
6	3	18	3

Код програми мовою С:

```
#include <iostream>
#include <locale>
using namespace std;
int main()
{
    setlocale(LC_CTYPE, "ukr");
    int a,
        b,
        c,
        min;
    cout << "Введіть натуральне число " << "A= ";
    cin >> a;
    cout << "Введіть натуральне число " << "B= ";
    cin >> b;
    cout << "Введіть натуральне число " << "C= ";
    cin >> c;

    if (a > b)
    {
        if (b > c)
            min = c;
        else
            min = b;
    }
    else
    {
            if (a>c)
               min=c;
            else
                min = a;
     }
        cout << "Мінімальне значення = "<< min;
    return 0;
}
```

Результат роботи програми

```
Введ?ть натуральне число A= 6
Введ?ть натуральне число B= 3
Введ?ть натуральне число C= 18
М?н?мальне значення = 3
```

Приклад2. Задано натуральне число з діапазону 1 – 9999. Вивести його опис у вигляді: «одноцифрове число», «двоцифрове число» і т. д..

Математична постановка

Для коректної роботи алгоритму при довільних значеннях вхідних даних, слід перевірити чи введене ціле число відповідає умові задачі, тобто належить діапазону [1; 9999]. Якщо введене число відповідає умові, то алгоритм можна реалізувати за схемою:

- числа менші за 10 – одноцифрові,
- інші числа, але менші за 100, – двоцифрові,
- всі інші числа, менші за 1000 – трицифрові,
- решта чисел – чотирицифрові.

Тому потрібно застосовувати розгалужений алгоритм, альтернативну форму вибору. Програмні специфікації напишемо у псевдокоді та графічній формі у вигляді блок-схеми.

Крок 1. Визначимо основні дії.

Крок 2. Деталізуємо дію належності n до заданого діапазону з використанням альтернативної форми вибору.

Крок 3. Деталізуємо дію належності n до одноцифрових чисел з використанням альтернативної форми вибору.

Крок 4. Деталізуємо дію належності n до двоцифрових чисел з використанням альтернативної форми вибору.

Крок 5. Деталізуємо дію належності n до трицифрових чисел з використанням альтернативної форми вибору.

Псевдокод

крок 1
початок
1.ввід n
2.належність n до заданого діапазону
3.належність n до одноцифрових чисел

крок 2
початок
1.ввід n
2.**якщо (n** < 0) АВО (**n > 9999)**
то

4.належність n до двоцифрових чисел
5.належність n до трицифрових чисел
кінець

«Некоректні дані»
інакше
3.належність n до одноцифрових чисел
4.належність n до двоцифрових чисел
5.належність n до трицифрових чисел
все якщо
кінець

крок 3
початок
1.ввід n
2.**якщо (n** < 0) АВО (**n > 9999)**
то
«Некоректні дані»
інакше
3. **якщо** (**n** < 10)
то
«одноцифрове»
інакше
4.належність n до двоцифрових чисел
5.належність n до трицифрових чисел
все якщо
все якщо
кінець

крок 4
початок
1.ввід n
2.**якщо (n** < 0) АВО (**n > 9999)**
то
«Некоректні дані»
інакше
3. **якщо** (**n** < 10)
то
«одноцифрове»
інакше
4.**якщо (n <** 100**)**
то
«двоцифрове»
інакше
5.належність n до трицифрових чисел

все якщо
все якщо
все якщо
кінець

крок 5
початок
1.ввід n
2.**якщо (n** < 0) АВО (**n > 9999)**
то
«Некоректні дані»
інакше
3. **якщо** (**n** < 10)
то
«одноцифрове»
інакше
4.**якщо (n <** 100**)**
то
«двоцифрове»

інакше

5. **якщо (n <** 1000**)**

то

«трицифрове»

інакше

«чотирицифрове»

все якщо

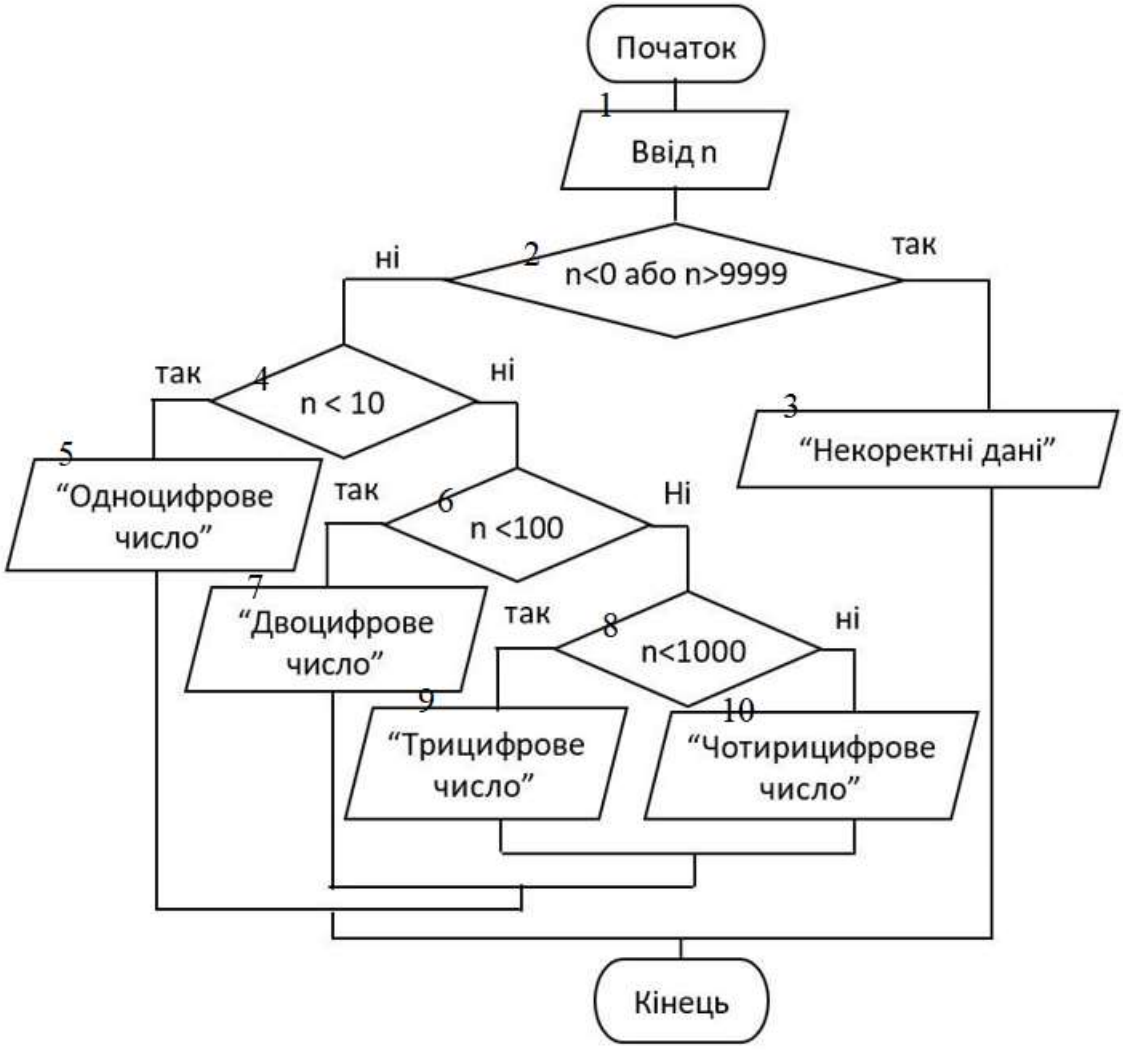
все якщо

все якщо

все якщо

кінець

Блок схема (крок 5)



Перевірка правильності роботи алгоритму

Таблиця 4. Тестування алгоритму

Блок	Приклад 1	Приклад 2	Приклад 3	Приклад 4	Приклад5
Початок					
1.Ініціалізація n	10001	23	256	6743	9
2. належність n до заданого діапазону	1001	23	256	6743	9
3.належність n до одноцифрових чисел	-	-	-	-	одноцифрове
4.належність n до двоцифрових чисел	-	двоцифрове	-	-	-
5.належність n до трицифрових чисел	-	-	трицифрове	-	-
6.належність n до чотирицифрових чисел				чотирицифрове	-
6. Вивід	некоректні дані	-	-	-	-
Кінець					

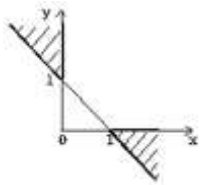
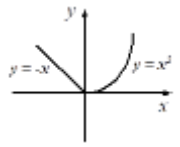
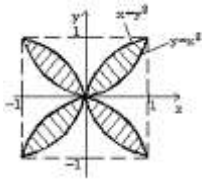
Код програми мовою С:

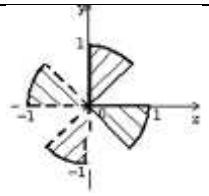
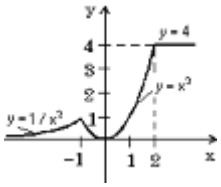
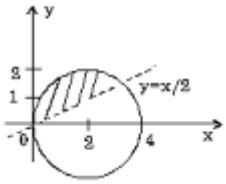
```
#include <iostream>
#include <locale>
using namespace std;
int main()
{
    setlocale(LC_CTYPE, "ukr");
    int n;
    cout << "Введіть натуральне число " << "від 1 до 9999 включно\n";
    cin >> n;

    if (n > 0 && n<10000)
    {
        if (n < 10) cout << n << " - одноцифрове число\n";
        else {
            if (n < 100)
            {
                cout<<n<<" - двоцифрове число\n";
            }
            else {
                if (n < 1000)
                {
                    cout << n << " - трицифрове число.\n";
                }
                else cout<<n<<" - чотирицифрове число.\n";
            }
        }
    }
     else {
     cout << "Ви ввели число, що не належить вказаному відрізку\n";
          }
    return 0;
}
```

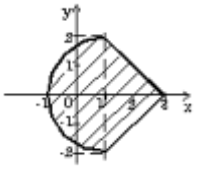
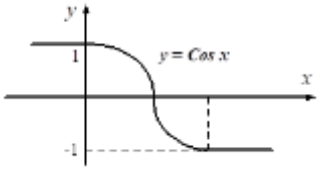
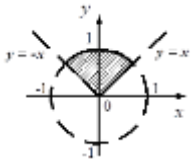
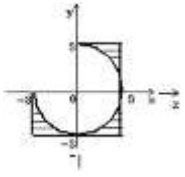
Індивідуальне завдання

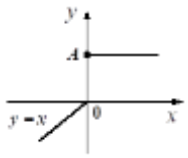
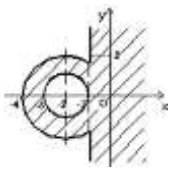
1. Виконати завдання згідно варіанту (табл.5)
2. Декомпозувати задачу, описати постановку.
3. Програмну специфікацію навести у псевдокоді та графічній формі.
4. Перевірити правильність алгоритму.
5. Код навести мовою С.

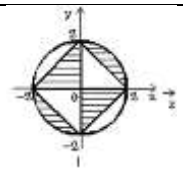
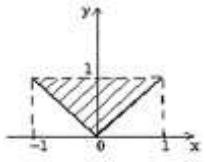
Таблиця 5. Варіанти завдань

№	Опис завдання
1	Задані дійсні числа *x*, *y*. Визначити, чи належить точка з координатами (*x*, *y*) заштрихованій частині площини: 
2	За заданими коефіцієнтами p та q знайти корені квадратного рівняння $x^2 = px + q$. Якщо рівняння має два різні корені x_1 та x_2, то знайти суму їх кубів. Якщо рівняння має один корінь x_1, то обчислити величину $\lg\|x_1\|$. А якщо рівняння коренів не має, то вивести відповідне повідомлення, а також обчислити відношення $p+q/\ pq$.
3	Обчислити $y = f(x)$, де функція $f(x)$ задана графіком: 
4	Заданий трикутник із сторонами *a*, *b*, *c*. Визначити типи кутів (прямий, тупий, гострий), що лежать навпроти цих сторін.
5	Задані дійсні числа *x*, *y*. Визначити, чи належить точка з координатами (*x*, *y*) заштрихованій частині площини 
6	За заданим номером місяця (ціле число з відрізка [1; 12]) визначити кількість днів в цьому місяці, за умови, що рік не високосний.
7	Задані дійсні числа *x*, *y*. Визначити, чи належить точка з координатами (*x*, *y*) заштрихованій частині площини:

№	Опис завдання
	
8	Задані дійсні числа a, b, c. З'ясувати, чи існує серед них хоча б одна пара, що дає у сумі парне число.
9	Обчислити $y = f(x)$, де функція $f(x)$ задана графіком 
10	По заданим координатам вершин трикутника на площині визначити тип трикутника (рівносторонній, рівнобедрений, різносторонній).
11	Задані дійсні числа x, y. Визначити, чи належить точка з координатами (x, y) заштрихованій частині площини 
12	За заданими коефіцієнтами a, b та c знайти корені квадратного рівняння $ax^2 + bx = c$. Якщо рівняння має два різні корені x_1 та x_2, то знайти суму їх модулів. Якщо рівняння має один корінь x_1, то обчислити величину $sin(\sqrt{\lvert x_1 \rvert})$. А якщо рівняння коренів не має, то вивести відповідне повідомлення, а також обчислити відношення $-b/2a$.
13	Задані дійсні числа x, y. Визначити, чи належить точка з координатами (x, y) заштрихованій частині площини:

№	Опис завдання
	
14	Задане ціле число d з відрізка [1; 7] задає день тижня починаючи з понеділка (1 – «понеділок», 2 – «вівторок» і т. д.). Скласти програму, яка за введеним числом d виведе відповідну назву дня тижня.
15	Обчислити $y = f(x)$, де функція $f(x)$ задана графіком : 
16	Задані дійсні додатні числа a, b, c, d. З'ясувати, чи можна прямокутник із сторонами a, b розмістити всередині прямокутник із сторонами c, d так, щоб кожна із сторін одного прямокутника була паралельна або перпендикулярна кожній стороні другого прямокутника.
17	Задані дійсні числа x, y. Визначити, чи належить точка з координатами (x, y) заштрихованій частині площини: 
18	З'ясувати, чи є вектор $\vec{a}$, заданий координатами a_1, a_2, a_3, і вектор $\vec{b}$, заданий координатами b_1, b_2, b_3, колінеарними.
19	Задані дійсні числа x, y. Визначити, чи належить точка з координатами (x, y) заштрихованій частині площини: 
20	Для заданого a знайти корінь рівняння $f(x) = 0$, де

№	Опис завдання
	$f(x)=\begin{cases}2ax+\lvert a-1\rvert, & \text{якщо } a>0\\ \dfrac{e^x}{\sqrt{1+a^2}}-1, & \text{інакше}\end{cases}$
21	Ціле число з діапазону 0-24 визначає поточну годину доби. Скласти програму, яка для вказаної години визначає пору доби: від 0 годин до 6 – «ніч», 6-11 – «ранок», 12-18 – «день», 19-24 – «вечір».
22	Обчислити $y = f(x)$, де функція $f(x)$ задана графіком 
23	За заданими коефіцієнтами p та q знайти корені квадратного рівняння $x^2 + px = q$. Якщо рівняння має два різні корені x_1 та x_2, то знайти суму їх кубів. Якщо рівняння має один корінь x_1, то обчислити величину $cos\lvert x_1\rvert$. А якщо рівняння коренів не має, то вивести відповідне повідомлення, а також обчислити відношення $q3$/р.
24	Задані дійсні числа *x*, *y*. Визначити, чи належить точка з координатами (*x*, *y*) заштрихованій частині площини 
25	З'ясувати, скільки розв'язків (один, безліч, не має) має система рівнянь, задана коефіцієнтами a_1, b_1, a_2, b_2 і правими частинами c_1, c_2: $$\begin{cases}a_1x + b_1y = c_1\\ a_2x + b_2y = c_2\end{cases}$$
26	Задані дійсні числа *x*, *y*. Визначити, чи належить точка з координатами (*x*, *y*) заштрихованій частині площини:

№	Опис завдання
	
27	Числа a і b виражають довжини катетів одного прямокутного трикутника, а c і d – іншого. З'ясувати, чи є ці трикутники подібними.
28	Дослідити область визначення і обчислити значення функції $$y(x) = \frac{\mathrm{Ln}\, d}{\lvert b^2 - a^2 \rvert \mathrm{Sin}\ \ c}$$
29	Задані дійсні числа x, y. Визначити, чи належить точка з координатами (x, y) заштрихованій частині площини 
30	Задане ціле число p, що означає оцінку в 12-бальній шкалі. Вивести опис відповідного рівня оцінки: 1-3 – «початковий рівень», 4-6 – «середній рівень», 7-9 – «достатній рівень», 10-12 – «високий рівень». Якщо ж p не належить відрізку [1; 12], вивести рядок «помилка».
31	Підприємець сплачує фіксовану суму податку у розмірі X грн, якщо сума його місячного прибутку менша за S_1 грн. Якщо ні, то X грн плюс p% від суми, яка перевищує S_1. Якщо ж сума прибутків підприємця перевищує S_2 ($S_2 > S_1$), то фіксована сума не стягується, а нараховується q% податку від загальної суми прибутків. Скласти програму для обчислення розміру податку на вказану суму прибутку.
32	У платіжній системі комісія за переказ коштів складає 10 грн, якщо сума переказу менша за 1000 грн, 1%, – на перекази сумою від 1000 до 10000 грн включно і 0,5% на суми, більші за 10000 грн. Скласти програму, яка для заданої суми переказу S розраховує розмір комісії та суму до сплати з урахуванням комісії.

№	Опис завдання
33	Задано дійсні позитивні числа a, b, c. Якщо існує трикутник зі сторонами a, b, c, то визначити його вид (прямокутний, гострокутний або тупокутний) і особливості (рівносторонній, рівнобедрений, різносторонній).
34	Робота світлофора для водіїв запрограмована таким чином: на початку кожної години протягом трьох хвилин горить зелений сигнал, потім протягом однієї хвилини - жовтий, протягом двох хвилин - червоний, протягом трьох хвилин - знову зелений. Дано дійсне число t, що означає час в хвилинах, що минув з початку чергової години. Визначити, сигнал якого кольору горить для водіїв в цей момент.
35	Оплата за телефонний дзвінок стягується за таким тарифом: p грн за першу хвилину розмови, незалежно від тривалості дзвінка, q грн за кожну наступну хвилину з щосекундною тарифікацією, після 10 хв розмови плата не стягується взагалі. Скласти програму, яка для заданої тривалості дзвінка за заданими тарифами p та q розраховує його вартість.

Вимоги до звіту

Звіт повинен містити:

- титульний аркуш;
- назву та мету роботи;
- варіант;
- постановку задачі;
- побудову математичної моделі;
- псевдокод алгоритму;
- блок схема алгоритму;
- випробування алгоритму;
- код мовою С;
- висновки.

Критерії оцінювання

Критерії оцінювання в процентах від максимального балу:

постановка задачі - 10%;

побудова математичної моделі – 10%

псевдокод алгоритму - 25%;

блок схема алгоритму – 25%

випробування алгоритму - 5%;

код програми – 20%;

висновки - 5%.

Контрольні запитання

1. Який обчислюваний процес називається розгалуженим?
2. Що таке логічна умова?
3. Як операції в логічних виразах Вам відомі? Що є результатом кожної з них?
6. Які форми оператору вибору Вам відомі?
7. Які блоки використовують у блок-схемах розгалуженого обчислюваного процесу?
4. На які питання потрібно дати відповіді, перш ніж складати блок-схему алгоритму розгалуженого обчислюваного процесу?
5. Які особливості побудови блок-схеми алгоритму розгалуженого обчислювального процесу?

Лабораторна робота 3
ЦИКЛІЧНІ АЛГОРИТМИ

Мета – дослідити подання операторів повторення дій та набути практичних навичок їх використання під час складання циклічних програмних специфікацій.

Основні теоретичні відомості

Цикл – це фрагмент обчислювального процесу, що повторюється багаторазово. Обчислювальні процеси, в яких виконуються одні й ті самі дії з різними даними, називаються циклічними. Послідовність операторів, що повторюється, складає тіло циклу. Лічильник циклу – це змінна, значення якої змінюється після кожної ітерації циклу, та яка визначає закінчення циклу. Для організації циклу потрібно передбачити такі значення:

- початкове значення змінної, яке буде змінюватися під час наступної ітерації циклу;
- кінцеве значення цієї змінної;
- крок зміни перед кожною новою ітерацією циклу.

Необхідно контролювати поточне значення лічильника циклу для перевірки умови виходу з циклу. Такою умовою може бути:

- перевищення лічильником циклу кінцевого значення;
- виконання заданої кількості повторень;
- досягнення заданої точності розрахунків.

Розрізняють арифметичні та ітераційні цикли.

Арифметичні цикли (цикли з параметром) використовують, якщо кількість повторень циклу є знаною заздалегідь або може бути визначена на підставі закону зміни лічильника циклу (рис.3)

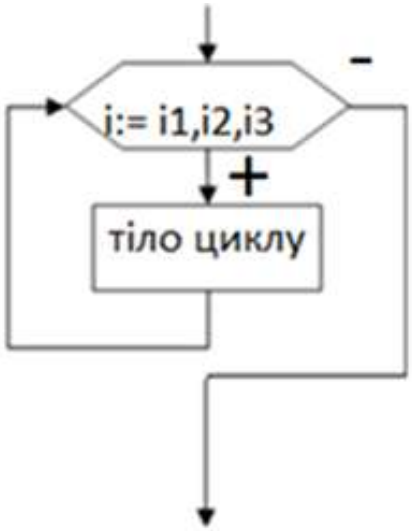


Рисунок 3 - Арифметичний цикл, де:

– i – змінна циклу (лічильник);

– вираз i1 - ініціалізація параметра, початкове значення циклу;

– вираз i2 - умова повторення, або кінцеве значення циклу;

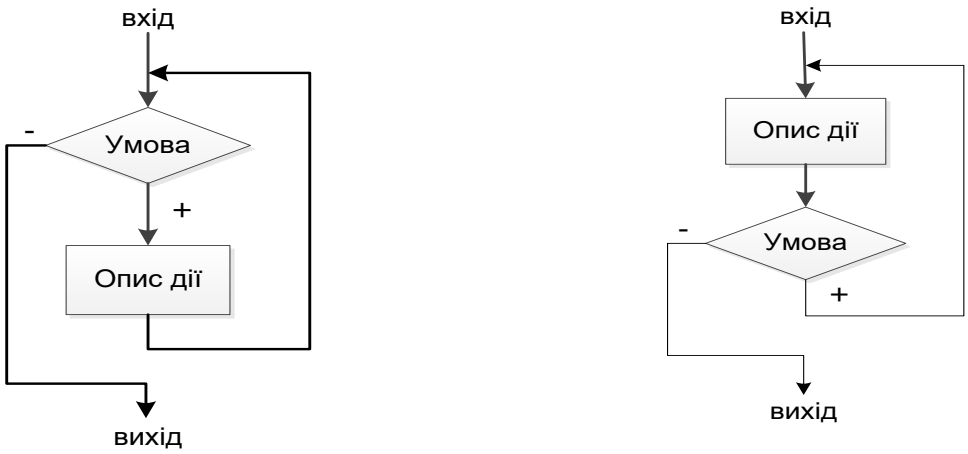
– вираз i3 - крок зміни параметра циклу.

Механізм реалізації:

1. Обчислюється вираз i1 (зазвичай, i1 = 0).
2. Обчислюється вираз i2 = i1 + i3.
3. Якщо значення виразу i2 відмінне від нуля (тобто, true), виконується тіло циклу, обчислюється вираз i3 і здійснюється перехід до пункту 2; якщо ж вираз i2 дорівнює нулю (тобто, false), то здійснюється вихід із циклу.

Істотним є те, що перевірка умови завжди виконується на початку циклу. Це означає, що тіло циклу може жодного разу не виконатися, якщо умова виконання відразу буде помилковою.

В блок-схемах для опису ітераційних циклів використовується основна та похідна схеми (рис.4). Ромб містить умову, а прямокутник – повторювану дію. За основною (*a*) схемою спочатку перевіряється умова, а потім у разі позитивного результату перевірки виконується тіло циклу. За похідною (*b*) схемою дія або дії, що складають тіло циклу, передують перевірці умови. З цього виходить, що за похідною схемою тіло циклу виконується хоча б один раз.



a) *b*)

Рисунок 4 - Основна (*a*) та похідна (*b*) схема ітераційного циклу.

Приклади застосування операторів повторення під час складання програмних специфікацій

Приклад 1. Задане дійсне число A та натуральне число N. Використовуючи один цикл і не використовуючи умовного оператора, знайти суму $1 - A + A^2 - A^3 + \ldots + (-1)^N A^N$.

Математична постановка.

В задачі шукаємо суму доданків з почерговою зміною знаку. Оскільки кожен наступний доданок відрізняється від попереднього, окрім протилежного знаку, на одиницю більшим степенем числа A, то для обчислення чергового доданка достатньо помножити попередній доданок на $-A$:

$(-1)^k A^k = (-1) \cdot (-1)^{k-1} \cdot A \cdot A^{k-1} = -A \cdot (-1)^{k-1} A^{k-1}$. Для розв'язання задачі використаємо арифметичний цикл, оскільки відома кількість ітерацій (n).

Алгоритм розв'язування задачі буде таким:

1. ідентифікація числа a, для якого обчислюється сума, та кількість доданків n;
2. ідентифікація початкового значення суми, та поточного доданка u;
3. в циклі n разів виконуються такі дії: знаходження значення наступного доданка та суми;
4. після завершення n ітерацій циклу обчислена сума виводиться на екран.

Крок 1. Визначимо основні дії.

Крок 2. Деталізуємо дію обчислення суми ряду

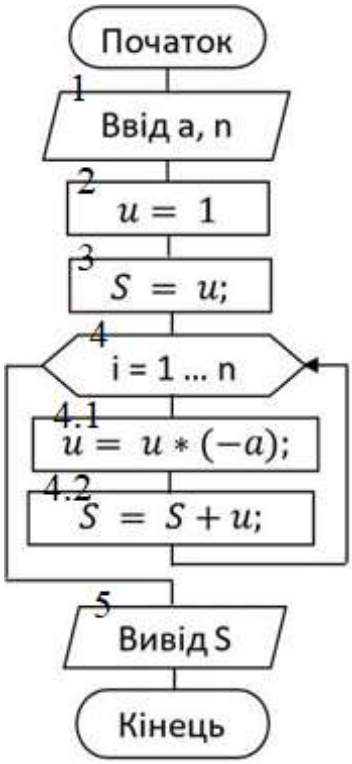
Псевдокод алгоритму

крок 1
початок
Ввід a, n
Ініціалізація u,s
Обчислення суми ряду
Вивід суми
кінець

крок 2
початок
Ввід a, n
U = 1
S = 1
Обчислення суми ряду
Вивід суми
кінець

крок 3
початок
Ввід a, n
U = 1
S = 1
повторити для i
u= u*(-a);
s= s + u
все повторити
Вивід суми
кінець

Блок схема алгоритму (крок 3)



Перевірка правильності алгоритму

Таблиця 6. Тестування алгоритму

i	A=4.5	u	s	A=8.6	u	s
1		-4.5	-3.5		-8.6	-7.6
2		20.250	16.750		73.960	66.360
3		-91.25	-74.375		-636.056	-569.696
4		410.063	335.678		5470.082	4900.386
5		-1845.281	-1509.594		-47042.702	-42142.3165

Код програми мовою С

```
#include <iostream>
using namespace std;
int main()
{
    double a,
        u = 1,
        s = u;
    int n;
    cout << "Enter double value a = ";
    cin >> a;
    cout << "Enter integer value n = ";
    cin >> n;

    for (int i = 0; i < n; i++)
    {
            u *= -a;
            s += u;
    }
printf("Summ s = %.3f", s);

}
```

Приклад2. Перевірити чи задане натуральне число m є числом Фібоначчі. Якщо так то вивести його порядковий номер n. Якщо ж ні, то вивести найближче до m число Фібоначчі, більше за m.

Математична постановка.

Процес розв'язування задачі полягає в поступовому знаходженні членів послідовності Фібоначчі $F_1 = 1, F_2 = 1, F_n = F_{n-1} + F_{n-2}, n = 3,4, \ldots$

і порівнянні їх з заданим користувачем числом m. Оскільки кількість повторень невідома, будемо використовувати ітераційний цикл. Програмну специфікацію напишемо у псевдокоді та графічній формі у вигляді блок-схеми.

Крок 1. Визначимо основні дії.

Крок 2. Деталізуємо дію обчислення двох перших чисел ряду Фібоначчі.

Крок 3. Деталізуємо дію знаходження числа з номером N ряду Фібоначчі.

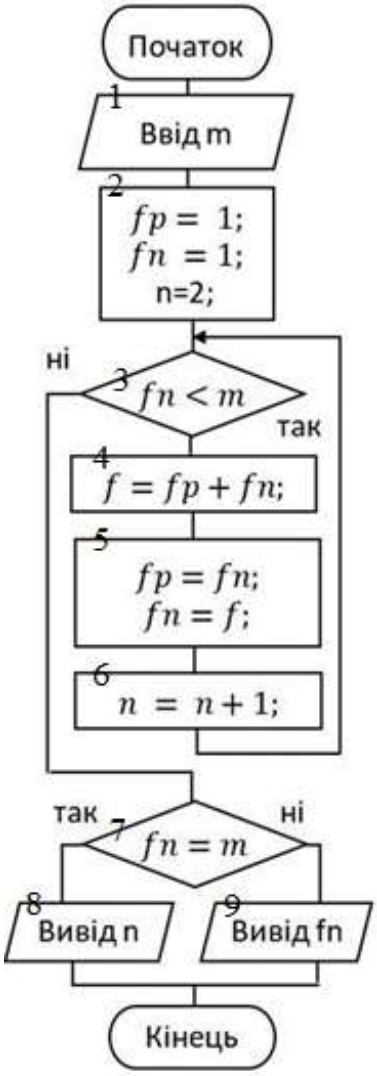
Псевдокод

крок 1
початок
ввід m
Обчислення двох перших чисел ряду Фібоначчі
Знаходження числа з номером N ряду Фібоначчі
вивід
кінець

крок 2
початок
ввід m
Fp = 1
Fn = 1
Знаходження числа з номером N ряду Фібоначчі
вивід
кінець

крок 3
початок
ввід m
Fp = 1
Fn = 1
N = 2
поки Fn < m
  повторити
    F=Fp + Fn;
    Fp = Fn;
    Fn = F;
    N = N + 1;
  все повторити
якщо Fn = m
 то
  вивід N
інакше
   вивід Fn
 все якщо
кінець

Блок-схема (крок 3)



Код програми мовою С

```
#include <iostream>
using namespace std;
int main() {
int m,
    f,
    fp = 1,
    fn = 1,
    n = 2;
cout << "Enter number m \n>>";
cin >> m;
while (fn < m)
    {
    f = fp + fn;
    fp = fn; fn = f;
    n++;
    }
if (fn == m)
    {
    cout << m << " is " << n
    << " Fibonacci number " << endl;
    }
else
    {
    cout << "First Fibonacci number after "
    << m << " is " << fn << endl;
    }
}
```

Приклад3 Знайти $S = \sqrt{x}$ із заданою точністю Ɛ=0.001 за формулою Герона.

Математична постановка

Якщо потрібно обчислити наближене значення квадратного кореня із заданою точністю ε (передбачається, що точне значення обчислити неможливо), потрібно використовувати наступний алгоритм:

1. Як нульове наближення S_0 взяти будь-яке число.

2. Обчислити за формулою наближене значення S_1.

3. Перевірити істинність рівності $| S_1 - S_0 | > \varepsilon$. Якщо вона є істинною, перейти до кроку 4, інакше до кроку 5.

4. $S_0 := S_1$, перейти до кроку 2.

5. S :=S_1, тобто, шуканому значенню кореня присвоїти значення S_1;

6. Записати значення S.

Оскільки кількість повторень невідома, будемо використовувати ітераційний цикл. Програмну специфікацію напишемо в псевдокоді та графічній формі у вигляді блок-схеми.

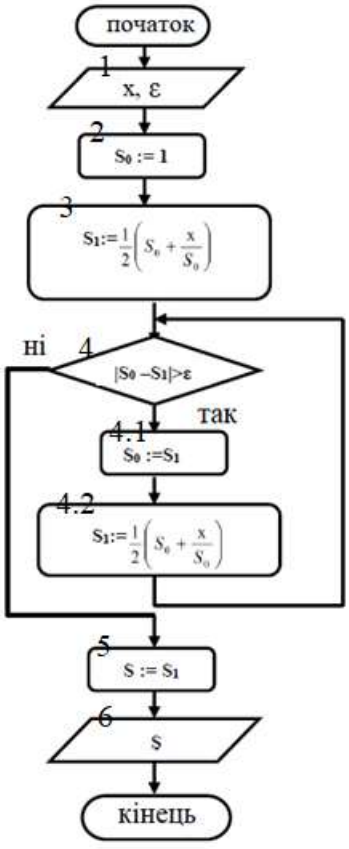
Псевдокод

крок 1
початок
ввід x,Ɛ
Обчислення S1
Обчислення S із заданою точністю
вивід
кінець

крок 2
початок
ввід x,Ɛ
S0 = 1
S1 = ½(S0+X/S0)
Знаходження числа з номером N ряду Фібоначчі
вивід
кінець

крок 3
початок
ввід x,Ɛ
S0 = 1
S1 = ½(S0+X/S0)
поки |S0-S1| > Ɛ
  повторити
    S0 = S1;
    S1 = ½(S0+X/S0)
  все повторити
   S = S1
  вивід S
кінець

Блок схема алгоритму (крок 3)



Перевірка правильності алгоритму $\sqrt{2}$ с точністю 0.001

1. S_0=1
2. $S_1 = \frac{1}{2}\left(1 + \frac{2}{1}\right) = \frac{3}{2} = 1.5$

3. $|S_1 - S_0| = \left|\frac{3}{2} - 1\right| = \frac{1}{2}$= 0.5 > 0.001 істина, переходимо до п.4
4. $S_0 = \frac{3}{2}$ =1.5 переходимо до п.2
5. $S_1 = \frac{1}{2}\left(\frac{3}{2} + \frac{4}{3}\right) = \frac{17}{12}$ = 1.4166
6. $|S_1 - S_0| = \left|\frac{3}{2} - \frac{17}{12}\right| = \frac{1}{12}$ = 0.083 > 0.001, істина, переходимо до п.4
7. ……

12. $|S_1 - S_0| = \left|\frac{666434}{470832} - \frac{577}{408}\right| = \frac{576}{408}$ = 0.0012 > 0.001, хибне твердження, переходимо до п.5

13. $S \approx S_1$ = 1.4142

Код програми мовою С

```
#include <iostream>
#include<cmath>
using namespace std;
int main()
{
    int x;
    double e,
        S,
        S0,
        S1;
    cout << "Enter integer x \n>>";
    cin >> x;
    cout << "Enter double e \n>>";
    cin >> e;

    S0 = 1;
    S1 =  (0.5 * (S0 + (x / S0)));
    while (fabs(S0 - S1) > e)
    {
        S0 = S1;
        S1= (0.5 * (S0 + (x / S0)));
    }
    S = S1;
    cout << "S= "<< S ;
}
```

Індивідуальне завдання

1. Виконати завдання згідно варіанту (табл.7)
2. Декомпозувати задачу, описати постановку.
3. Програмну специфікацію навести у псевдокоді та графічній формі.
4. Перевірити правильність алгоритму.
5. Код навести мовою С.

Таблиця 7. Варіанти завдань

№	Опис завдання
1	Дано два цілі числа A та B (A < B). Знайти суму всіх парних цілих чисел від 2A до 4B включно.
2	Обчислити з точністю Ɛ наближене значення функції e^{x} за формулою $S = 1 + X + \frac{X^2}{2!} + \frac{X^3}{3!} + \frac{X^4}{4!} + \frac{X^5}{5!} + \ldots$
3	Обчислити з точністю Ɛ наближене значення функції $\sin \mathrm{X}$ за формулою $S = X - \frac{X^3}{3!} + \frac{X^5}{5!} - \frac{X^7}{7!} + \frac{X^9}{9!} + \ldots$
4	Обчислити з точністю Ɛ наближене значення функції $(1 + X)^M$ за формулою $S = 1 + M \cdot X + \frac{M(M-1)X^2}{2!} + \frac{M(M-1)(M-2)X^3}{3!} + \frac{M(M-1)(M-2)(M-3)X^4}{4!} + ..$
5	Обчислити з точністю Ɛ наближене значення функції $\sqrt{1+X}$ за формулою $S = 1 + \frac{X}{2} - \frac{X^2}{2\cdot 4} + \frac{3\mathrm{X}^3}{2\cdot 4\cdot 6} - \frac{3\cdot 5\mathrm{X}^4}{2\cdot 4\cdot 6\cdot 8} + \ldots$
6	Обчислити з точністю Ɛ наближене значення функції arcsinX за формулою $S = X - \frac{X^3}{2\cdot 3} + \frac{3\mathrm{X}^5}{2\cdot 4\cdot 5} - \frac{3\cdot 5\mathrm{X}^7}{2\cdot 4\cdot 6\cdot 7} + \frac{3\cdot 5\cdot 7\mathrm{X}^9}{2\cdot 4\cdot 6\cdot 8\cdot 9} + ..$
7	Вивести на екран таблицю значень функції $y = \sin x + \sqrt{x}$ у рівномірно розподілених на відрізку $[a; b]$ точках $x_0 = a < x_1 < \cdots < x_n = b$, $x_k = x_{k-1} + h$, $h = const$. Межі відрізка $a \geq 0$, $b > a$ та кількість точок n ввести з клавіатури, крок h обчислити за довжиною відрізка

№	Опис завдання
	та кількістю точок. Знайти середнє арифметичне обчислених значень функції.
8	Обчислити з точністю Ɛ наближене значення функції $\frac{e^X - e^{-X}}{2}$ за формулою $S = X + \frac{X^3}{3!} + \frac{X^5}{5!} + \frac{X^7}{7!} + \frac{X^9}{9!} + ..$
9	Обчислити з точністю Ɛ наближене значення функції $\frac{e^X + e^{-X}}{2}$ за формулою $S = 1 + \frac{X^2}{2!} + \frac{X^4}{4!} + \frac{X^6}{6!} + \frac{X^8}{8!} + ..$
10	Стартова сума депозиту у банку складає S_0 грн. Кожного місяця банк нараховує власнику $p\%$ від поточного розміру суми депозиту. Окрім цього, кожного місяця власник сам поповнює депозит на фіксовану суму M грн. Визначити через скільки місяців сума депозиту досягне потрібної власнику величини S грн.
11	Обчислити з точністю Ɛ наближене значення функції $\mathrm{LN}\frac{1+X}{1-X}$ за формулою $S = 2 \cdot (X + \frac{X^3}{3} + \frac{X^5}{5} + \frac{X^7}{7} + \frac{X^9}{9} + ...)$
12	Послідовність цілих чисел задається рекурентними співвідношеннями: $A_1 = 3$, $A_2 = 7$, $A_n = 3A_{n-1} - 2A_{n-2}$, $n = 3,4, \ldots$ Скласти програму для знаходження порядкового номера найбільшого члена послідовності $\{A_n\}$, що не перевищує задане число m.
13	Обчислити з точністю Ɛ наближене значення функції $\mathrm{LN}\left(X + \sqrt{1 + X^2}\right)$ за формулою $S = X - \frac{1}{2} \cdot \frac{X^3}{3} + \frac{1}{2} \cdot \frac{3}{4} \cdot \frac{X^5}{5} - \frac{1}{2} \cdot \frac{3}{4} \cdot \frac{5}{6} \cdot \frac{X^7}{7} + \frac{1}{2} \cdot \frac{3}{4} \cdot \frac{5}{6} \cdot \frac{7}{8} \cdot \frac{X^9}{9} - \ldots$
14	Обчислити з точністю Ɛ наближене значення функції e^{-X^2} за формулою $S = 1 - \frac{X^2}{1!} + \frac{X^4}{2!} - \frac{X^6}{3!} + \frac{X^8}{4!} - \ldots + (-1)^N \frac{X^{2N}}{N!} + ..$

№	Опис завдання
15	Обчислити з точністю Ɛ наближене значення функції $(1+X)^{-2}$ за формулою $S = 1 - 2X + 3X^2 - 4X^3 + 5X^4 - ..$
16	Для заданого числа $A \in [1; 20]$ знайти найменше натуральне число n, для якого сума $S = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$ буде більшою за A. Вивести цю суму.
17	Обчислити з точністю Ɛ наближене значення функції π за формулою $S = 4 \cdot (1 - \frac{1}{3} + \frac{1}{5} - ... + (-1)^{N-1}\frac{1}{2N-1} + ...)$
18	Обчислити з точністю Ɛ наближене значення функції $\frac{1}{\sqrt{1-X^2}}$ за формулою $S = 1 + \frac{1}{2}X^2 + \frac{1\cdot3}{2\cdot4}X^4 + \frac{1\cdot3\cdot5}{2\cdot4\cdot6}X^6 + \frac{1\cdot3\cdot5\cdot7}{2\cdot4\cdot6\cdot8}X^8 - ..$
19	Побудувати таблицю значень заданої функції двох змінних у заданому користувачем прямокутнику $[a;\ b] \times [c;\ d]$. Крок зміни значень змінних $x \in [a; b]$ та $y \in [c; d]$ обчислити таким чином, щоб на кожному з відрізків розміщувалося по 8 точок. Результат обчислень вивести на екран у вигляді прямокутної таблиці, в якій по горизонталі вказані різні значення змінної x, по вертикалі – значення змінної y, а на перехресті стовпця зі значенням змінної x та рядка зі значенням змінної y – відповідне значення функції. $u = 2x^2 + 3y^2$
20	Обчислити з точністю Ɛ суму нескінченного ряду за формулою $S = \sum_{N=0}^{\infty}(-1)^N \frac{X^{2N-1}}{4N^2-1}$
21	Обчислити з точністю Ɛ суму нескінченного ряду за формулою $S = \sum_{N=0}^{\infty}(-1)^{N+1} \frac{X^{2N}\ln(1+2N)}{4N^2}$
22	Обчислити з точністю Ɛ суму нескінченного ряду за формулою $S = \sum_{N=1}^{\infty}(-1)^{N+1} \frac{X^{-2^N}}{2^N(2N-1)}$

№	Опис завдання
23	Обчислити з точністю Ɛ суму нескінченного ряду за формулою $$S=\sum_{N=1}^{\infty}(-1)^{N+1}\frac{(2^{2\mathrm{N}}-1)X^{-2\mathrm{N}}}{2^{2\mathrm{N}}2\mathrm{N}}$$
24	Обчислити з точністю Ɛ суму нескінченного ряду за формулою $$S=\sum_{N=1}^{\infty}(-1)^{N}\frac{(2\mathrm{N}-1)X^{-N}}{(N+5)2\mathrm{N}}$$
25	Обчислити з точністю Ɛ суму нескінченного ряду за формулою $$S=\sum_{N=1}^{\infty}\frac{X^{3\mathrm{N}}(3\mathrm{N}+1)}{N!}$$
26	Побудувати таблицю значень заданої функції двох змінних у заданому користувачем прямокутнику $[a;\ b]\times[c;\ d]$. Крок зміни значень змінних $x\in[a;b]$ та $y\in[c;d]$ обчислити таким чином, щоб на кожному з відрізків розміщувалося по 8 точок. Результат обчислень вивести на екран у вигляді прямокутної таблиці, в якій по горизонталі вказані різні значення змінної x, по вертикалі – значення змінної y, а на перехресті стовпця зі значенням змінної x та рядка зі значенням змінної y – відповідне значення функції. $$u=\sqrt{2xy+4y^3}$$
27	Обчислити з точністю Ɛ суму нескінченного ряду за формулою $$S=\frac{1}{3\lg X}+\frac{1+2}{(3\lg X)^2}+\frac{1+2+3}{(3\lg X)^3}+\ldots+\frac{1+2+\ldots+N}{(3\lg X)^N}+\ldots$$
28	Обчислити з точністю Ɛ суму нескінченного ряду за формулою $S=X+\frac{X^3}{Y^3+X}+\frac{X^5}{Y^6+X^2}+\ldots+\frac{X^{2\mathrm{N}+1}}{Y^{3\mathrm{N}}+X^N}+\ldots,$
29	Обчислити з точністю Ɛ суму ряду за формулою $$\sum_{n=1}^{\infty}\frac{2}{(2n+1)(2n+3)}$$

№	Опис завдання
30	Побудувати таблицю значень заданої функції двох змінних у заданому користувачем прямокутнику $[a; b] \times [c; d]$. Крок зміни значень змінних $x \in [a; b]$ та $y \in [c; d]$ обчислити таким чином, щоб на кожному з відрізків розміщувалося по 8 точок. Результат обчислень вивести на екран у вигляді прямокутної таблиці, в якій по горизонталі вказані різні значення змінної x, по вертикалі – значення змінної y, а на перехресті стовпця зі значенням змінної x та рядка зі значенням змінної y – відповідне значення функції. $u = 5\ sin(x + y)$;
31	Обчислити з точністю Ɛ суму ряду за формулою $$\sum_{n=1}^{\infty}(\sqrt{n+2} - 2\sqrt{n+1} + \sqrt{n})$$
32	Обчислити з точністю Ɛ= 10^{-6} суму ряду за формулою $$\sum_{n=1}^{\infty}\frac{(2n)!^2}{(4n)!}$$ Скільки членів ряду потрібно?
33	Обчислити з точністю Ɛ = 10^{-8} суму ряду за формулою $$\sum_{n=1}^{\infty}\frac{(-1)^n}{2n-1}\left(\frac{1-x}{1+x}\right)^n$$ Скільки членів ряду потрібно?
34	Обчислити з точністю Ɛ=10^{-5} суму ряду за формулою $$\sum_{n=1}^{\infty}\frac{n}{n+1}\left(\frac{x}{2x+1}\right)^n$$ Скільки членів ряду потрібно?
35	Обчислити з точністю Ɛ=10^{-7} суму ряду за формулою $$\sum_{n=1}^{\infty}\frac{1*3*\ldots*(2n-1)}{2*4*\ldots*2n}\left(\frac{2x}{1+x^2}\right)^n$$ Скільки членів ряду потрібно?

Вимоги до звіту

Звіт повинен містити:

- титульний аркуш;
- назву та мету роботи;
- варіант;
- побудову математичної моделі;
- псевдокод алгоритму;
- блок схема алгоритму;
- випробування алгоритму;
- код програми;
- висновки.

Критерії оцінювання

Критерії оцінювання в процентах від максимального балу:

- побудова математичної моделі – 20%;
- псевдокод алгоритму – 20%;
- блок схема алгоритму – 20%;
- випробування алгоритму – 15%;
- код програми – 20%;
- висновки – 5%.

Контрольні запитання

1. Дати визначення циклу.
2. Який обчислюваний процес називається циклічним?
3. Який цикл називається арифметичним?
4. Що потрібно визначити для організації циклу?
5. Що може бути умовою виходу з арифметичного циклу?
6. Що таке лічильник циклу?
7. Який цикл називається ітераційним?
8. У яких випадках доцільно використовувати ітераційні цикли і чому?

Лабораторна робота 4
ДОПОМІЖНІ АЛГОРИТМИ

Мета - дослідити особливості роботи допоміжних алгоритмів та набути практичних навичок їх використання під час складання програмних специфікацій підпрограм.

Основні теоретичні відомості

Дуже часто в різних задачах зустрічається послідовність дій, яку доводиться виконувати декілька разів для різних наборів даних. Така послідовність, зазвичай може бути описана як окремий алгоритм або «підзадача», що є частиною основної. Програму, що реалізує такий алгоритм називають *підпрограмою*. До розв'язування великих та складних задач застосовують *декомпозицію* – підхід що полягає в розбитті основної задачі на прості задачі, *підпрограми*, які взаємодіють між собою.

Підпрограми реалізуються на базі допоміжних алгоритмів.

Приклади застосування допоміжних алгоритмів під час складання програмних специфікацій

Приклад 1 Дані дві послідовності натуральних чисел. Підрахувати кількість пар (x_i, y_i), i = 1,2, ..., n, у яких числа кратні один одному, тобто, або x_i ділиться націло на y_i, або навпаки. Обчислити суму часток від ділення у знайдених парах. Підрахувати кількість некратних пар та обчислити суму залишків від ділення в цих парах.

Математична постановка

Частина дій в задачі повторюється багато разів. Складемо блок схему ділення двох натуральних чисел *m* і *n*. Знайдемо частку і залишок від ділення більшого числа на менше. Алгоритм використовуватимемо як допоміжний. Програмну специфікацію напишемо у графічній формі у вигляді блок-схеми.

Крок 1. Визначимо основні дії.

Крок 2. Деталізуємо дію ділення двох натуральних чисел, як допоміжний алгоритм DIV

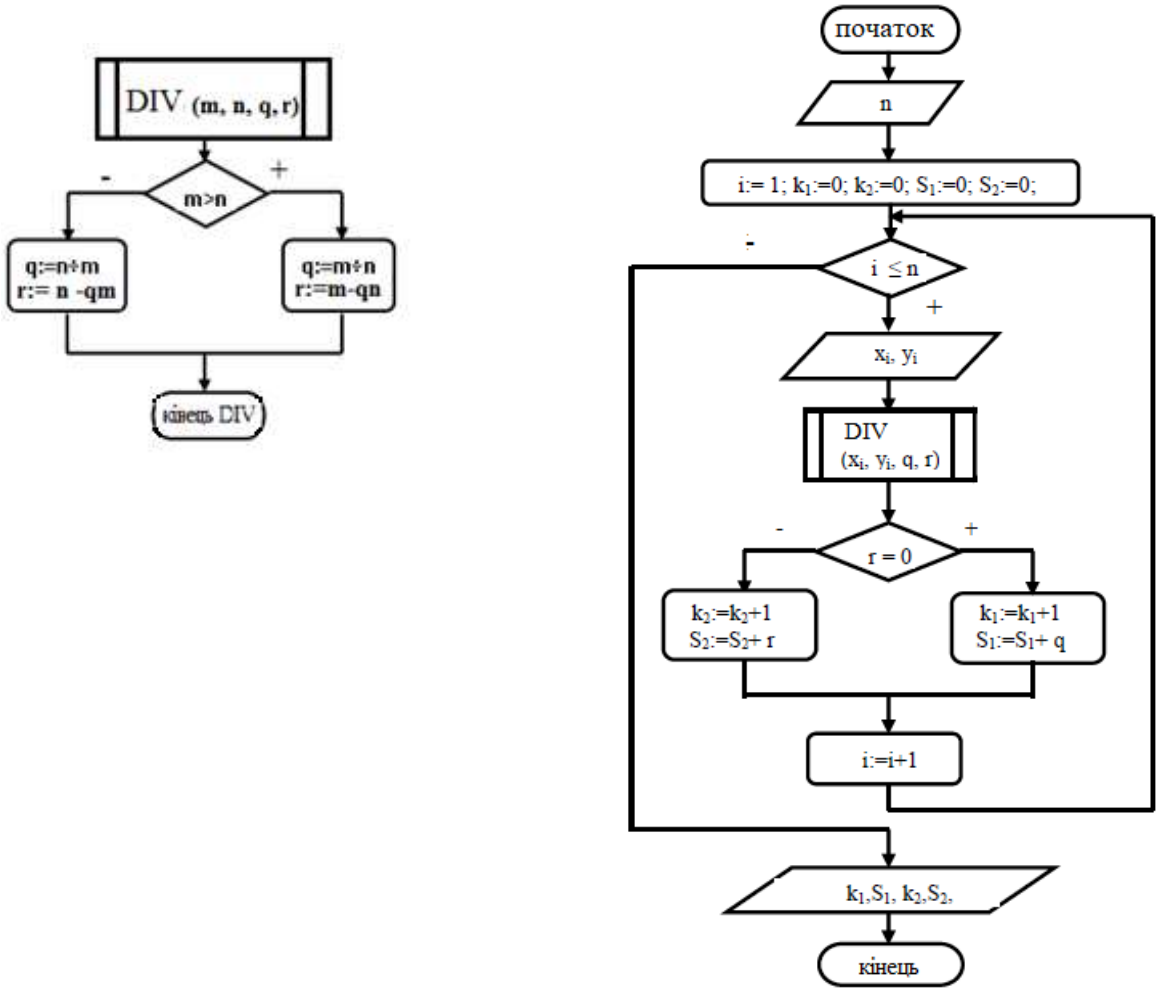
Крок 3. Деталізуємо дію знаходження кількості кратних пар

Крок 4. Деталізуємо дію знаходження кількості некратних пар

Крок 5. Деталізуємо дію знаходження суми залишків в некратних парах

Крок 6. Деталізуємо дію знаходження суми часток від ділення в кратних парах

Загальна блок схема допоміжного та основного алгоритму наведена на рисунку 5 (a,b).



a) *b*)

Рисунок 5 - Загальна блок-схема допоміжного *a*) та основного алгоритму *b*)

Приклад 2 Створити за допомогою підпрограми два масиви A та B, по 12 елементів кожен, заповнивши їх дійсними числами за правилом

$x_k=(-1)^k\sqrt{(k^2+p)(k+q^2)}$, де p та q – деякі задані користувачем числа (різні для кожного з масивів), k = 1,2, . . . ,12.. Масив C утворити з масиву B, замінивши в ньому всі від'ємні елементи середнім значенням елементів масиву A. Вивести масиви на екран, в кожному масиві знайти елемент, значення якого найближче до числа 1, розробивши для цього окремі підпрограми.

Математична постановка

У підсумку наш код міститиме три підпрограми: одну для заповнення масиву за вказаним правилом, другу – для виводу елементів масиву на екран, а третю – для пошуку в масиві елемента, значення якого найближче до 1. Загальна схема роботи програми буде такою:

- заповнення масивів A та B;
- на основі масивів A та B за вказаним правилом створення масиву C;
- до кожного з масивів застосовується підпрограма для виводу на екран, та підпрограма для пошуку елемента, значення якого найближче до 1, знайдене значення виводить на екран.

Програмну специфікацію напишемо у графічній формі у вигляді блок-схеми.

Крок 1. Визначимо основні дії.

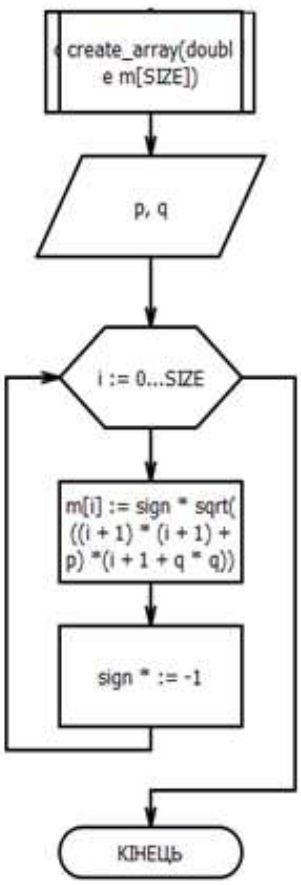
Крок 2. Деталізуємо дію заповнення масивів A та B, як допоміжний алгоритм create_array

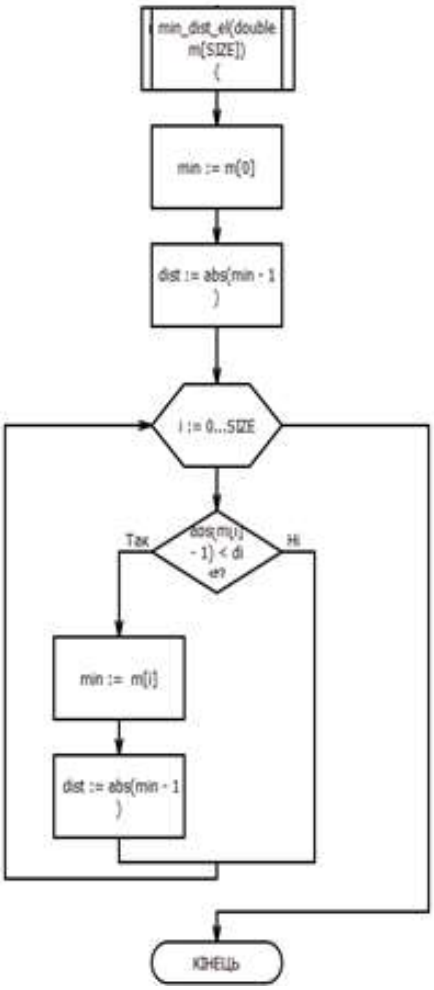
Крок 3. Деталізуємо дію створення масиву C

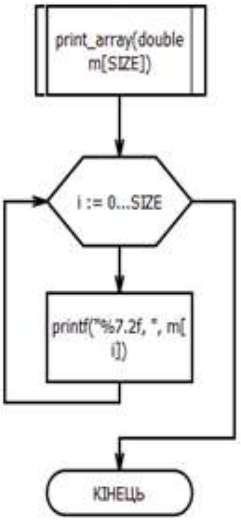
Крок 4. Деталізуємо дію виводу на екран як допоміжний алгоритм print_array

Крок 5. Деталізуємо дію пошуку елемента як допоміжний алгоритм min_dist_el

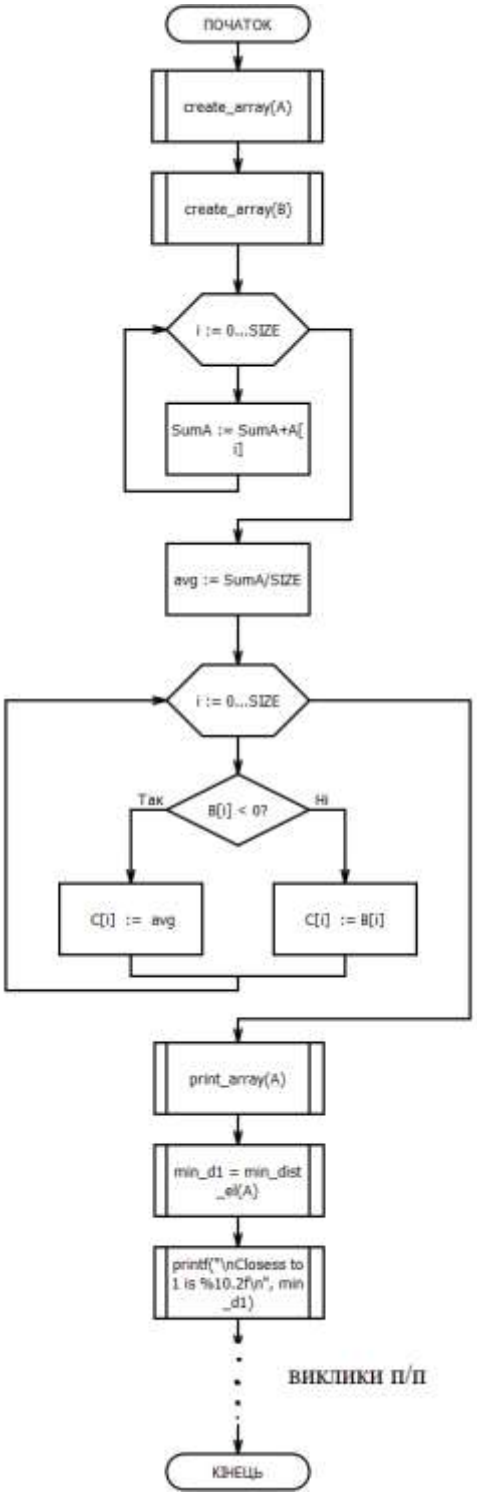
Допоміжні алгоритми







Основний алгоритм



Код мовою С

Головна програма

```c
#include <iostream>
#define SIZE 12
using namespace std;

void create_array(double m[SIZE]);
void print_array(double m[SIZE]);
double min_dist_el(double m[SIZE]);

int main()
{
    double A[SIZE], B[SIZE], C[SIZE];
    create_array(A);
    create_array(B);
    double SumA = 0,
           min_d1;
    for (int i = 0; i < SIZE; i++)
    {
        SumA += A[i];
    }
    double avg = SumA / SIZE;
    for (int i = 0; i < SIZE; i++)
    {
        if (B[i] < 0)
        {
            C[i] = avg;
        }
        else
        {
            C[i] = B[i];
        }
    }
    cout << "Array A:\n";
    print_array(A);
    min_d1 = min_dist_el(A);
    printf("\nClosess to 1 is %10.2f\n", min_d1);
    cout << "Array B:\n";
    print_array(B);
    min_d1 = min_dist_el(B);
    printf("\nClosess to 1 is %10.2f\n", min_d1);
    cout << "Array C:\n";
    print_array(C);
    min_d1 = min_dist_el(C);
    printf("\nClosess to 1 is %10.2f\n", min_d1);
}
```

Підпрограми

```c
void create_array(double m[SIZE])
{
    int sign = -1;// -1^n при n = 1
    double p, q;
    cout << "p = ";
    cin >> p;
    cout << "q = "; cin >> q;
    for (int i = 0; i < SIZE; i++)
    {
        m[i] = sign * sqrt(((i + 1) * (i + 1) + p) *(i + 1 + q
        sign *= -1; // обчислення -1 ^ (k + 1)
    }
}
void print_array(double m[SIZE])
{
    for (int i = 0; i < SIZE; i++)
    {
        printf("%7.2f, ", m[i]);
    }
}

double min_dist_el(double m[SIZE])
{
    double min = m[0];// початкове наближення  його відстань до
    double dist = abs(min - 1);
    for (int i = 0; i < SIZE; i++)
    {
        if (abs(m[i] - 1) < dist)
        {// знайдено новий елемент найближчий до 1
            min = m[i];
            dist = abs(min - 1);
        }
    }    return min;
}
```

Індивідуальне завдання

1. Виконати завдання згідно варіанту (табл.8)
2. Декомпозувати задачу, описати постановку.
3. Програмну специфікацію навести у псевдокоді та графічній формі.
4. Програмну реалізацію алгоритму навести мовою програмування С.
5. Тестування програми.

Таблиця 8. Варіанти завдань

№	Опис завдання
1	Реалізувати підпрограму для обчислення довжини відрізка AB на координатній площині за координатами точок $A(x_a,y_a)$ та $B(x_b, y_b)$. Використати створену підпрограму в програмі для знаходження периметра трикутника, заданого координатами його вершин.
2	Створити і заповнити масиви дійсних чисел A та B, кожен по 16 елементів за допомогою підпрограми, яка заповняє масив введеними з клавіатури числами. Масив C утворити з масиву B, віднявши від кожного його елемента відповідний елемент масиву A. Розробити підпрограму для виводу масиву на екран і з її допомогою вивести усі масиви. В кожному масиві знайти суму квадратів мінімального та максимального елементів, створивши для обчислення окрему підпрограму.
3	Обчислити добуток двох цілих додатних чисел $f(a,b)=ab$ за таким алгоритмом: $f(a,b)=a+f(a,b-1)$, якщо b – непарне, $f(a,b)=2f(a,b/2)$, якщо b – парне, $f(a,b)=0$, якщо $b=0$.
4	Створити підпрограму для перевірки того, чи є задане натуральне число простим (натуральне число називають простим, якщо воно не має жодного натурального дільника, окрім себе та одиниці). Застосувати цю підпрограму до підрахунку кількості простих чисел в заданому наборі (масиві) натуральних чисел.
5	Розробити підпрограму для перевірки того, чи є задане натуральне число квадратом деякого натурального числа. Використовуючи її вивести усі повні квадрати із заданого набору (масиву) натуральних чисел.

№	Опис завдання
6	Створити підпрограму для перевірки того, чи задане натуральне число є паліндромом (дорівнює числу, записаному тими ж цифрами в зворотному порядку). Застосувати цю підпрограму до виводу усіх паліндромів з заданого відрізка $[A;\ B]$.
7	Створити підпрограму для знаходження найбільшого спільного дільника (НСД) двох натуральних чисел за алгоритмом Евкліда. Написати програму для знаходження НСД чотирьох заданих чисел, враховуючи, що НСД(m, n, k) = НСД(НСД(m, n), k).
8	Задано координати вершин трикутника $A(x_a,y_a)$, $B(x_b,y_b)$ та $C(x_c,y_c)$. Користувач вводить координати x та y деякої точки $M(x,\ y)$ на площині. Визначити найближчу до M вершину трикутника ABC. В програмі використати підпрограму для обчислення відстані між двома точками на площині.
9	Реалізувати підпрограму для обчислення площі рівнобедреного трикутника за відомими довжинами його основи та бічної сторони. Використати створену підпрограму в програмі для знаходження площі бічної поверхні правильної n-кутної піраміди зі стороною основи a та бічним ребром l.
10	Створити підпрограму для знаходження суми цифр натурального числа. У заданому масиві з 20-ти цілих додатних чисел знайти число з максимальною сумою цифр, використовуючи створену підпрограму. Якщо чисел з такою сумою цифр декілька, то вивести перше з них.
11	Задано натуральне число n. Вивести усі натуральні числа з відрізка $[n; 2n]$, які можна подати у вигляді суми квадратів двох натуральних чисел. Розробити та використати підпрограму для перевірки того, чи є задане натуральне число квадратом деякого натурального числа.
12	Створити підпрограму для знаходження найбільшого спільного дільника (НСД) двох натуральних чисел за алгоритмом Евкліда. Використовуючи цю підпрограму вивести на екран усі пари взаємно

№	Опис завдання
	простих чисел (НСД = 1) серед елементів заданого масиву натуральних чисел.
13	Розробити підпрограму для перевірки того, чи є задане натуральне число квадратом деякого натурального числа. Використовуючи її порахувати кількість повних квадратів натуральних чисел, що належать заданому відрізку $[A; B]$
14	Реалізувати підпрограму для обчислення довжини відрізка AB в тривимірних декартових координатах за відомими координатами вершин $A(x_a, y_a, z_a)$ та $B(x_b, y_b, z_b)$. Використати створену підпрограму в програмі для визначення найближчої до заданої точки $P(x, y, z)$ вершини трикутника ABC, заданого координатами його вершин.
15	Реалізувати підпрограму для обчислення довжини відрізка AB на координатній площині за координатами точок $A(x_a,y_a)$ та $B(x_b, y_b)$. Використати створену підпрограму в програмі для знаходження довжини ламаної $A_1A_2 \ldots A_{10}$, координати вершин якої задано за допомогою пари масивів $X(10)$ та $Y(10)$.
16	Створити підпрограму для знаходження кількості цифр натурального числа. Заповнити довільно масив з 20-ти цілих додатних чисел. Використовуючи створену підпрограму вивести спочатку одноцифрові елементи масиву, потім двоцифрові, трицифрові і т. д.
17	Задано трикутник розміром m×n, де m, n – цілі числа і m>0, n>0. Обчислити площу трикутника на основі залежності: $$S(n,m)=\begin{cases}1, \text{якщо } n=m=1;\\ S(n-1,m)+m, \text{якщо } n>1;\\ S(n,m-1)+1, \text{якщо } m>1.\end{cases}$$
18	Перевірити, чи введене користувачем натуральне число підтверджує гіпотезу Гольдбаха, яка полягає в тому, що будь яке парне натуральне число, більше за 2 можна подати у вигляді суми двох простих чисел (натуральне число називають простим, якщо воно не має жодного

№	Опис завдання
	натурального дільника, окрім себе та одиниці). В програмі використати підпрограму для перевірки того, чи є вказане натуральне число простим.
19	Утворити чотири масиви дійсних чисел P, Q, R та S, по 17 елементів кожен, за допомогою підпрограми, яка заповняє масив введеними з клавіатури числами. Створити підпрограми для виведення масиву на екран та для обчислення в масиві відношення максимального його елемента до мінімального. Використовуючи створені підпрограми вивести на екран усі масиви та відношення максимального елемента до мінімального в кожному з них.
20	Три масиви дійсних чисел A, B та C утворюються з 17 елементів кожен. Розробити підпрограму, яка заповнює масив випадковими числами з відрізка $[-5x;\ 4x]$ (число x задає користувач, окремо для кожного масиву) та скористатися нею для заповнення масивів A, B та C. Реалізувати підпрограми для того, щоб вивести масиви на екран, та знайти в кожному відношення абсолютної величини суми від'ємних елементів до суми додатних елементів. Використовуючи створені підпрограми вивести масиви та вказану величину на екран.
21	Масиви дійсних чисел K та M, що містять по 10 елементів кожен, заповнюються випадковими числами з відрізка [-10; 10] за допомогою спеціально створеної для цього підпрограми. Масив P утворити з масиву M, замінивши в ньому всі додатні елементи максимальним елементом масиву K. Реалізувати підпрограми для виводу елементів масиву на екран та для знаходження в ньому суми від'ємних елементів і виконати їх із кожним масивом.
22	Обчислити кількість комбінацій з n різних елементів по m. Кількість комбінацій визначається формулою

№	Опис завдання
	$C_n^m = \begin{cases} 1, \text{ якщо } m = 0, n > 0 \text{ або } m = n \geq 0; \\ 0, \text{ якщо } m > n \geq 0; \\ C_{n-1}^{m-1} + C_{n-1}^{m} \text{ в інших випадках.} \end{cases}$
23	Три масиви дійсних чисел A, B та C по 12 елементів кожен заповняються елементами за правилом $\frac{(-1)^k x}{\sqrt{k}}$, $k = 1,2,\ldots,12$, де x деяке задане користувачем число (різне для кожного масиву). Реалізувати підпрограму для заповнення масивів елементами за вказаним правилом та використати її в програмі. Вивести масиви на екран, в кожному масиві знайти добуток від'ємних елементів, створивши для цього відповідні підпрограми, та викликаючи їх в основній програмі.
24	Створити підпрограму, яка заповнює масив з 15 дійсних чисел введеними з клавіатури даними та використати її для заповнення заданих в програмі масивів $M(15)$, та $T(15)$. Масив S утворити з масиву T, замінивши в ньому всі від'ємні елементи мінімальним елементом масиву M. За допомогою підпрограм вивести масиви на екран та знайти в кожному масиві кількість елементів, абсолютна величина яких менша за 1.
25	Утворити три масиви дійсних чисел M, P та Q, кожен містить по 15 елементів та заповнити їх за допомогою створеної для цього підпрограми випадковими числами з відрізка [0; p] (число p задає користувач, окремо для кожного масиву). Вивести масиви на екран, в кожному масиві знайти різницю максимального та мінімального елемента, використовуючи для цього окремі підпрограми.
26	Утворити масиви дійсних чисел U, V та W, кожен містить по 11 елементів та заповнити їх елементами за допомогою підпрограми, яка заповнює масив елементами за правилом $x_k = 7\ sin\ (\sqrt[3]{(k-m)(k-n)})$, $k = 1,2,\ldots,11$, де m та n – деякі задані користувачем числа (різні для кожного з масивів). Вивести масиви на екран, в кожному масиві знайти

№	Опис завдання
	добуток елементів, абсолютна величина яких більша за 1, створивши для цього відповідні підпрограми.
27	Реалізувати підпрограму для вводу з клавіатури елементів масиву та використати її для заповнення двох масивів дійсних чисел A та B, по 14 елементів кожен. Масив C утворити за правилом $C_k = min\{A_k; B_k\}$. Створити підпрограму для виводу елементів масиву на екран та виконати її для кожного із масивів. У кожному масиві знайти середнє арифметичне додатних елементів за допомогою спеціально розробленої для цього підпрограми.
28	Обчислити значення функції Аккермана для двох невід'ємних цілих чисел n та m, де: $A(n,m)=\begin{cases} m+1, \text{ якщо } n=0; \\ A(n-1,1), \text{ якщо } n\neq 0, m=0; \\ A(n-1,A(n,m-1)), \text{ якщо } n>0, m>0. \end{cases}$
29	Створити три масиви дійсних чисел K, M та P, по 12 елементів кожен та заповнити їх за допомогою створеної для цього підпрограми випадковими числами з відрізка $[-a; a]$ (число a задає користувач, окремо для кожного масиву). Реалізувати підпрограми для виведення масиву на екран та для знаходження в ньому кількості елементів, з відрізка [0; 3]. Вивести кожен масив разом із шуканою величиною на екран за допомогою створених підпрограм.
30	Реалізувати підпрограму для створення та заповнення масиву з 15 дійсних чисел за правилом $sin(x^3) + 3x\ sin\ k$, k = 1,2, . . . ,15, x – деяке задане користувачем число. Використовуючи її утворити три масиви дійсних чисел $M(15)$, $P(15)$ та $Q(15)$. Вивести масиви на екран, в кожному масиві знайти півсуму максимального та мінімального елемента з використанням окремих, розроблених для цього підпрограм.

№	Опис завдання
31	Три масиви дійсних чисел X, Y та W, складаються з 20 елементів кожен. Створити підпрограму для заповнення масиву введеними з клавіатури числами та використати її для вводу елементів заданих масивів. Вивести масиви на екран, в кожному масиві знайти різницю квадратів максимального та мінімального елементів, використовуючи окремі, розроблені для цього підпрограми.
32	Утворити масиви дійсних чисел A та B по 12 елементів кожен та заповнити їх випадковими числами з відрізка $[-50; 50]$ за допомогою створеної для цього підпрограми. Масив C утворити за правилом $C_k = max\{|A_k|; 2B_k\}$. Вивести масиви на екран, в кожному масиві знайти номер елемента, найближчого за величиною до 1. Вивід кожного масиву та пошук у ньому вказаної величини оформити у вигляді окремих підпрограм.
33	Створити і заповнити масиви дійсних чисел U, V та X, кожен по 16 елементів за допомогою підпрограми, яка заповняє масив випадковими числами з відрізка $[-2x; 3x]$ (число x задає користувач, окремо для кожного масиву). Вивести масиви на екран, в кожному масиві знайти мінімальний додатний елемент. Для виводу масиву та пошуку у ньому вказаної величини реалізувати окремі підпрограми.
34	Утворити масиви дійсних чисел M, P по 12 елементів кожен та заповнити їх введеними користувачем даними за допомогою окремої, розробленої для цього підпрограми. Масив S утворити з масиву P, замінивши в ньому всі додатні елементи мінімальним елементом масиву M. Вивести масиви на екран, в кожному масиві знайти номер максимального елемента, використовуючи окремі, розроблені для цього підпрограми.
35	Три масиви дійсних чисел A, B та C містять по 15 елементів кожен. Заповнити масиви A та B введеними користувачем даними за допомогою окремої, розробленої для цього підпрограми. Масив C

№	Опис завдання
	утворити додаванням оберненого масиву A до масиву B, тобто $C_1 = A_{15} + B_1$, $C_2 = A_{14} + B_2$, $C_3 = A_{13} + B_3$, і т.д. Вивести масиви на екран, в кожному масиві знайти суму елементів, значення яких належать інтервалу (-2; 0). Вивід кожного масиву та пошук у ньому вказаної величини оформити у вигляді окремих підпрограм.

Вимоги до звіту

Звіт повинен містити:

- титульний аркуш;
- назву та мету роботи;
- варіант;
- постановку задачі;
- побудову математичної моделі;
- псевдокод алгоритму;
- блок схема алгоритму;
- код програми;
- тестування програми;
- висновки.

Критерії оцінювання

Критерії оцінювання в процентах від максимального балу:

- постановка задачі - 15%;
- побудова математичної моделі – 15%
- псевдокод алгоритму - 20%;
- блок схема алгоритму – 20%
- код програми – 20 %;
- тестування програми – 5%;
- висновки - 5%.

Контрольні запитання

1. Дати визначення «функція» та «процедура». В чому їх істотна відмінність?
2. Що таке прототип функції?
3. Наведіть способи передачі параметрів у підпрограму, надайте приклади.
4. Наведіть способи повертання результатів з підпрограми, надайте приклади.
5. Чи можна використовувати глобальні змінні у підпрограмах?
6. Що таке локальні об’єкти підпрограми?
7. Що таке процедурна абстракція та абстракція керування?

Лабораторна робота 5

ЛІНІЙНІ ТА НЕЛІНІЙНІ СТРУКТУРИ ДАНИХ

Мета - вивчити можливості та особливості створення та обробки лінійних та нелінійних структур даних.

Теоретичні відомості

Структура даних – це засіб зберігання даних для забезпечення доступу до них для читання та запису певним способом. Не існує однієї структури даних, що працює найкращим чином для всіх задач, тому у різних умовах та обмеженнях використовуються різні структури.

Лінійні структури даних – це такі структури, елементи яких мають форму послідовності або лінійного списку. До лінійних структур даних відносять масиви, стеки, черги та зв’язані лінійні списки.

Стек – це структура даних, яка побудована на метафорі «магазин автомату» або «трубка з одним відкритим кінцем» та забезпечує доступ до елементів тільки з одного кінця за правилом «перший увійшов – останній вийшов» - First in – Last out (LIFO). Стек (рис.6) забезпечує дві операції з його елементами - «помістити» - Push, та «витягнути» - Pop. Один й той же елемент може бути оброблений в стеці тільки один раз, коли він знаходиться на вершині.

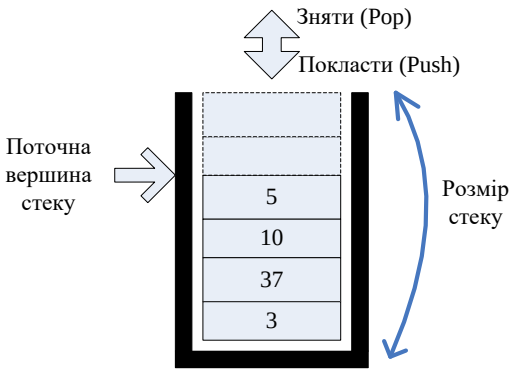


Рисунок 6 - Стек

Черга – це структура даних, яка побудована на метафорі «черга обслуговування» або «однонаправлена трубка з двома відкритими кінцями» та забезпечує правило «перший увійшов – перший вийшов» - First in – First out (FIFO). Черга (рис.7) забезпечує дві операції з його елементами – «Помістити у чергу» - Enqueue та «Взяти з черги» - Dequeue. Елемент може бути взятий з черги тільки один раз.

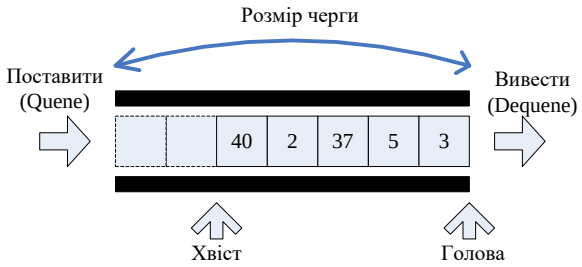


Рисунок 7 - Черга

Зв’язаний лінійний список – це структура, всі елементи якої зв’язані один з іншим за допомогою посилань (рис.8). На перший елемент списку посилається окрема змінна, яка містить його адресу. Кожний елемент списку містить покажчик з адресою наступного елементу. Останній елемент списку містить посилання на *null*.

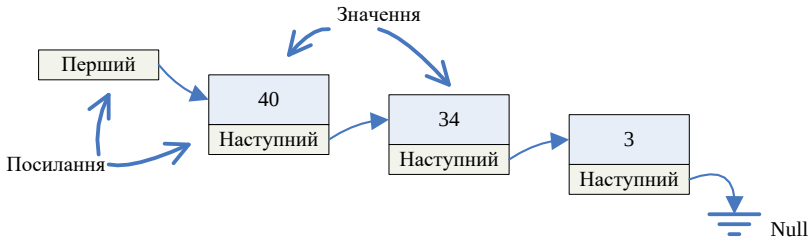


Рисунок 8 - Зв’язаний список

Нелінійні структури даних – це структури, елементи яких не можливо представити у вигляді послідовності. Дерево – це нелінійна структура, яка зберігає данні у зв’язаних вузлах, що утворюють ієрархію (рис.9). Кожний вузол дерева (окрім кореневого) зв’язаний зі старшим вузлом та може бути зв’язаним з

молодшими вузлами. Бінарне дерево – це таке дерево, кожний вузол якого може мати тільки два молодших вузла.

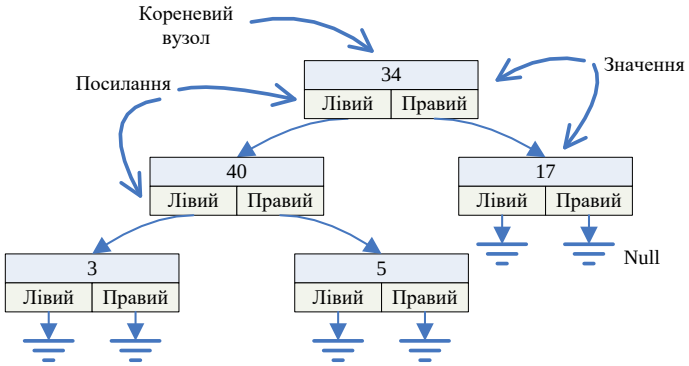


Рисунок 9 - Дерево

Для реалізації структур даних застосовується векторне та зв’язане подання. При цьому останнє використовується частіше завдяки гнучкості та ефективному використанню пам’яті.

Векторний список для всіх лінійних структур даних моделюється значеннями індексованого типу (рис.10) Тому в даному випадку пропонується для зберігання та обробки значень списку (стека, черги) використовувати індексовану змінну, що зберігатиме одновимірне мультизначення з компонентів того типу, який мають дані лінійного списку, а для вказівки на один або два кінці списку – змінні цілого типу, що зберігатимуть індекси компонентів – крайніх вузлів списку.

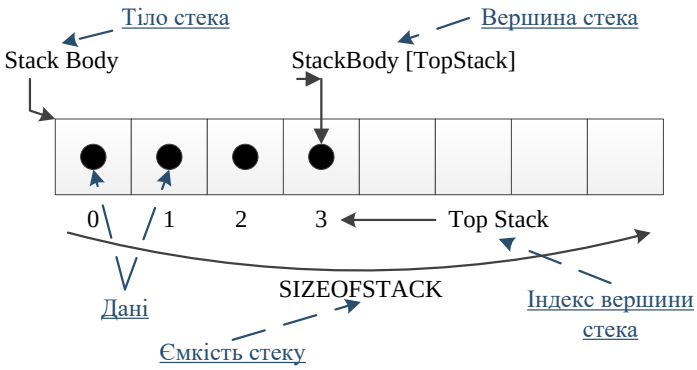


Рисунок 10 - Структура даних Стек у векторному поданні

Стеку векторному поданні має бути описаний наступним чином:

```
const int SIZEOFSTACK = <макс. кількість значень структури даних>;
struct Stack{
        <тип значень структури даних> StackBody[SIZEOFSTACK];
        int TopStack;
};
Stack myStack;
```

Зв'язний список для лінійних структур даних моделюється за допомогою вказівних змінних і динамічного розподілу пам'яті в «купі». При цьому значення в «купі» не розташовуються впритул, а навігація у списку здійснюється завдяки зв'язкам вузлів списку (рис.11). Тому в даному випадку пропонується для зберігання та обробки значень списку (стека, черги) використовувати покажчик іменованого типу, одне поле якого зберігатиме дані, інші поля, описані типом-покажчиком на той самий іменований тип, відповідатимуть за зчеплення вузлів.

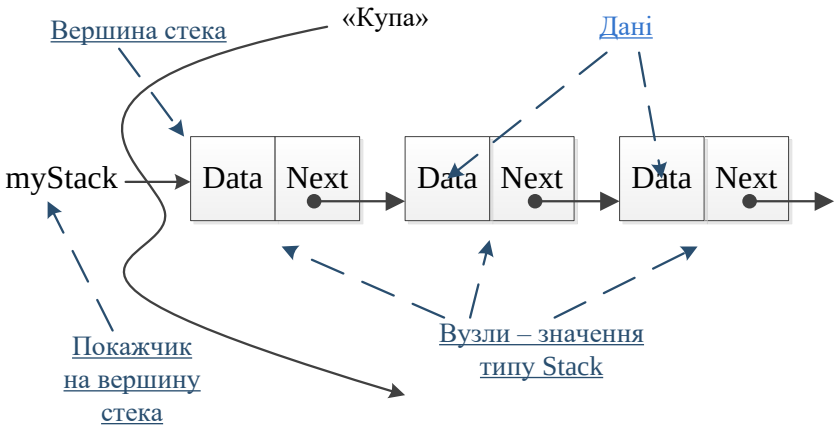


Рисунок 11 - Структура даних Стек у зв'язаному поданні

Стек у зв'язаному поданні має бути описаний наступним чином:

```
struct Stack
{ <тип значень структури даних стек> Data;
  Stack *Next;
};
Stack *myStack;
```

Приклад застосування структур даних під час складання програмних специфікацій

Приклад. Створити векторний стек символьних значень (максимальна можлива довжина стеку – 10), додати в стек N (0…..10), видалити M (0…..15) значень.

Постановка

Крок 1 Визначимо основні дії.

Крок 2 ініціюємо вхідні дані

Крок 3 деталізуємо дію додавання N елементів стеку

Крок 4 деталізуємо дію видалення M елементів стеку

Псевдокод

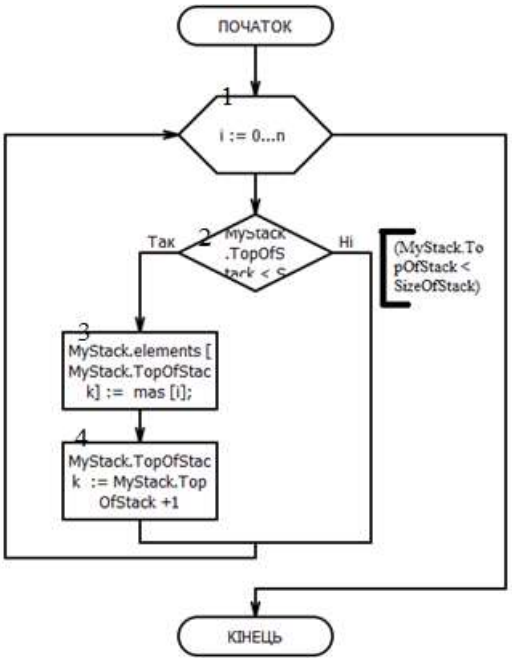
крок 1
початок
1. ініціалізація вхідних даних
2. деталізуємо дію додавання N елементів стеку
3. деталізуємо дію видалення M елементів стеку
4.**кінець**

крок 2
початок
1.N=8
M=14
MyStack.TopOfStack =**0**
2. **повторити**
для (i=0; N;i++)

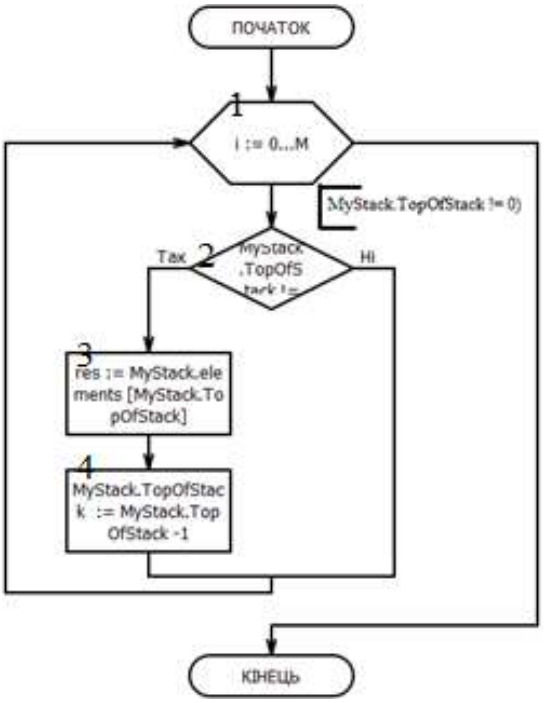
2.1.**якщо** (MyStack.TopOfStack < SizeOfStack)
то
MyStack.elements [MyStack.TopOfStack] = mas [i];
MyStack.TopOfStack ++;
все якщо
все повторити
3. деталізуємо дію видалення M елементів стеку
кінець

крок 3
початок
1.N=8
M=14
MyStack.TopOfStack =0
2. **повторити**
для (i=0;_N;i++)
2.1.**якщо** (MyStack.TopOfStack < SizeOfStack)
то
MyStack.elements [MyStack.TopOfStack] = mas [i];
MyStack.TopOfStack ++;
все якщо
все повторити
3. **повторити**
для (i=0; M ;i++)
3.1.**якщо** (MyStack.TopOfStack != 0)
то
rez=MyStack.elements [MyStack.TopOfStack]
MyStack.TopOfStack --
все якщо
все повторити

Блок схема додавання N елементів стеку



Блок схема видалення M елементів стеку



Код програми на С

```
#include <iostream>
using namespace std;
int main()
{
    const int SizeOfStack = 10;
    struct Stack
    {
        char elements[SizeOfStack];
        int TopOfStack;
    };
    char rez;
    int i,
        N,
        M;
    cout << "how many elements to insert  1....10      ";
    cin >> N;
    cout << "how many elements to delete  1....15       ";
    cin >> M;
    Stack MyStack;
    MyStack.TopOfStack = 0;
    char mas[10] = { 'M', 'y', 'p', 'r', 'o', 'g', 'r', 'a', 'm', '!' };
    for (i = 0; i < N; i++)
        if (MyStack.TopOfStack < SizeOfStack)
        {
            MyStack.elements[MyStack.TopOfStack] = mas[i];
            MyStack.TopOfStack++;
        }
    for (i = 0; i < M; i++)
        if (MyStack.TopOfStack != 0) {
            rez = MyStack.elements[MyStack.TopOfStack];
            MyStack.TopOfStack--;
        }
}
```

Індивідуальне завдання

1. Виконати завдання згідно варіанту (табл.9)
2. Декомпозувати задачу, описати постановку.
3. Програмну специфікацію навести у псевдокоді та графічній формі.
4. Перевірити правильність алгоритму.
5. Код навести мовою С.

Таблиця 9. Варіанти завдань

№	*Структура даних*	*Представлення*	*Тип даних*	*Завдання*
1	Стек	векторне	дійсний	В новий стек переписати всі додатні елементи, підрахувати кількість
2	Черга	зв’язане	цілий	Підрахувати середньо арифметичне елементів черги, видалити всі парні значення
3	Черга	векторне	символьний	В нову чергу переписати всі літери "S" із номером її входження
4	Односпрямований список	зв’язане	цілий	Знайти суму елементів списку, значення яких більше за значення введеного елементу (нумерація починається з голови списку)
5	Черга	зв’язане	символьний	Видалити всі символи, що розташовані до символу «!». Підрахувати їх кількість
6	Стек	зв’язане	символьний	Перевірити, чи утворює рядок символів паліндром?
7	Стек	векторне	дійсний	Знайти перший елемент більший за середнє значення
8	Односпрямований список	зв’язане	цілий	Отримати новий список зі значень елементів менших за середнє значення
9	Односпрямований список	зв’язане	символьний	Знайти перше входження заданого символу. Переписати в новий список елементи після цього символу

№	*Структура даних*	*Предста влення*	*Тип даних*	*Завдання*
10	Стек	зв’язане	цілий	Знайти максимальний елемент, в новий стек записати елементи, які розташовані після максимального
11	Односпрям ований список	векторне	дійсний	Отримати новий список зі значень позитивних елементів поточного списку. Видалити всі від’ємні елементи поточного списку
12	Черга	зв’язане	цілий	Знайти кількість елементів, кратних 2. Отримати нову чергу з цих елементів
13	Стек	векторне	дійсний	Видалити елементи, які розташовані до мінімального елементу
14	Черга	зв’язане	символь ний	Видалити послідовності елементів, які розташовані по зростанню їх кодів
15	Черга	векторне	символь ний	Замінити елементи, що розташовані на парних позиціях, на «А». В нову чергу переписати подвоєну кількість цих символів
16	Односпрям ований список	зв’язане	цілий	Отримати новий список зі значень елементів значення яких більші за задане значення. Видалити елементи, менші за задане значення
17	Черга	векторне	цілий	Знайти добуток елементів, які розташовані на парних позиціях та кратних заданому значенню
18	Стек	зв’язане	дійсний	Знайти перше входження елементу меншого за середнє значення. Знайти суму цих елементів

№	*Структура даних*	*Предста влення*	*Тип даних*	*Завдання*
19	Стек	векторне	дійсний	Знайти суму елементів, які розташовані на не парних позиціях стеку. Записати їх в новий стек
20	Односпрям ований список	зв’язане	символь ний	Отримати новий список зі значень елементів, більших за символ «?» (нумерація починається з голови списку). Підрахувати їх кількість
21	Односпрям ований список	векторне	цілий	Знайти суму елементів, що розташовані на парних позиціях поточного списку та є більшими за середнє значення
22	Стек	зв’язане	символь ний	В новий стек записати елементи, розташовані до заданого символу, знайти їх кількість
23	Односпрям ований список	векторне	дійсний	Знайти добуток елементів, що розташовані на не парних позиціях поточного списку та менші за задане значення
24	Черга	зв’язане	цілий	Знайти перший від’ємний елемент, подвоїти всі наступні елементи
25	Стек	векторне	цілий, символь ний	В першому стеці знаходяться цілі числа, в другому – арифметичні операції, виконати арифметичні дії, результати занести в третій стек
26	Черга	зв’язане	символь ний	Отримати нову чергу зі значень елементів більших за задане значення. Підрахувати кількість

№	*Структура даних*	*Представлення*	*Тип даних*	*Завдання*
27	Черга	векторне	символьний	В нову чергу записати задане слово
28	Односпрямований список	зв’язане	дійсний	Отримати новий список зі значень позитивних елементів поточного списку. Видалити всі від’ємні елементи
29	Черга	векторне	цілий	Знайти кількість елементів, кратних 2, які розташовані на парних позиціях. Отримати новий список з цих значень
30	Стек	зв’язане	символьний	Перевірити, чи є паліндромом введена послідовність символів
31	Стек	векторне	дійсний	Елементи, що розташовані на парних позиціях, записати в новий стек, видалити задану кількість елементів
32	Черга	зв’язане	цілий	Отримати нову чергу зі значень елементів більших за середнє, підрахувати кількість
33	Черга	векторне	символьний	Замінити елементи, що розташовані на парних позиціях, на заданий символ. Видалити 3 елемента
34	Односпрямований список	зв’язане	дійсний	Видалити вказану кількість елементів з N позиції
35	Стек	зв’язане	цілий	Парні елементи записати в один стек, не парні – в інший

Вимоги до звіту

Звіт повинен містити:

- титульний аркуш;
- назву та мету роботи;
- варіант;
- побудову математичної моделі;
- псевдокод алгоритму;
- блок схема алгоритму;
- випробування алгоритму;
- код програми;
- висновки.

Критерії оцінювання

Критерії оцінювання в процентах від максимального балу:

- побудова математичної моделі – 20%;
- псевдокод алгоритму – 20%;
- блок схема алгоритму – 20%;
- випробування алгоритму – 15%;
- код програми – 20%;
- висновки – 5%.

Контрольні запитання

1. Що таке структура даних?
2. Наведіть класифікацію значень структур даних.

3. Наведіть типи та засоби представлень лінійних структур даних.

4. Наведіть типи та засоби представлень нелінійних структур даних.

5. Поясніть сутність векторної реалізації структур даних

6. Поясніть сутність зв’язаної реалізації структур даних

7. Наведіть приклад опису типу вузла лінійного списку у зв’язаній реалізації та поясніть устрій вузла

8. Наведіть приклад опису односпрямованого та двоспрямованого списків та поясніть різницю.

Лабораторна робота 6

АЛГОРИТМИ ПОШУКУ В ПОСЛІДОВНОСТЯХ

Мета – дослідити методи пошуку у впорядкованих і невпорядкованих послідовностях та набути практичних навичок їх використання під час складання програмних специфікацій.

Основні теоретичні відомості

Пошук є однією з найчастіше використовуваних операцій у програмуванні. Взагалі, різноманітних видів пошуку дуже багато і вони розрізняються між собою способом організації даних. Але у кожному алгоритмі пошуку є свої переваги і недоліки.

Для реалізації операції пошуку в програмуванні часто використовують спеціальну структуру даних – *масив*.

Масив є індексованою сукупністю однотипних даних, що позначаються спільним іменем змінної. Доступ до кожного окремого значення даних здійснюється за іменем масиву та номером елемента в ньому.

Лінійний пошук - самий простий вид пошуку деякого елемента серед інших, що здійснюється за допомогою перевірки кожного елемента до тих пір, поки вони не будуть збігатися. Загальна ідея цього виду пошуку така: усі елементи розглядаються послідовно, один за одним. Перевагами такого пошуку є простота його реалізації, він не потребує додаткового об’єму пам’яті або додаткової роботи з функціями. Це дозволяє працювати вже під час отримання даних.

Для обробки значень елементів послідовностей використовується оператор повторення. Дії в тілі оператора повторення виконуються n разів (де n – кількість елементів у послідовності). За допомогою змінної i можна звернутися до кожного з елементів послідовності за його порядковим номером, наприклад *A[i]*.

Бінарний пошук, це такий вид пошуку, у якому пошук елемента з множини відбувається за допомогою ділення деякої кількості раз цієї множини навпіл. Бінарний пошук застосовується для відсортованих множин.

Перевагами такого пошуку є швидкість пошуку, якщо порівнювати з послідовним пошуком. Але є й такий недолік, що двійковий пошук може використовуватись лише на упорядкованій множині.

Інтерполяційний пошук здійснюється у впорядкованому масиві за величиною ключів кожного елемента. У цьому алгоритмі пошуку величини повинні бути рівномірно розподілені у деякому їх інтервалі від *x* до *y*. З цього виходить, що якщо є відома величина *C* ключа пошуку, то положення потрібного нам запису можливо передбачити, а не шукати у всьому масиві. Цей пошук використовується частіше, ніж бінарний.

Приклади застосування алгоритмів пошуку під час складання програмних специфікацій

Приклад 1. Задано послідовність значень A[n]. Знайти мінімальне значення у послідовності

Математична постановка

Для вирішення поставленої задачі використаємо алгоритм знаходження мінімального значення шляхом перебору всіх елементів. Програмні специфікації напишемо у псевдокоді та графічній формі у вигляді блок-схеми.

Крок 1. Визначимо основні дії.

Крок 2. Деталізуємо дію ініціалізації початкових значень для пошуку (визначення мінімального значення як першого елемента послідовності *(Min := A[1])*.

Крок 3. Деталізуємо дію перебору та перегляду елементів послідовності з використанням оператора повторення. Повторення виконується n разів, тобто поки *i* не буде дорівнювати *n+1*.

Псевдокод

крок 1
початок
Ініціалізація початкових значень для пошуку
Перебір та перегляд елементів послідовності
кінець

крок 2
початок
Min := A[0]
i := 1
Перебір та перегляд елементів послідовності
кінець

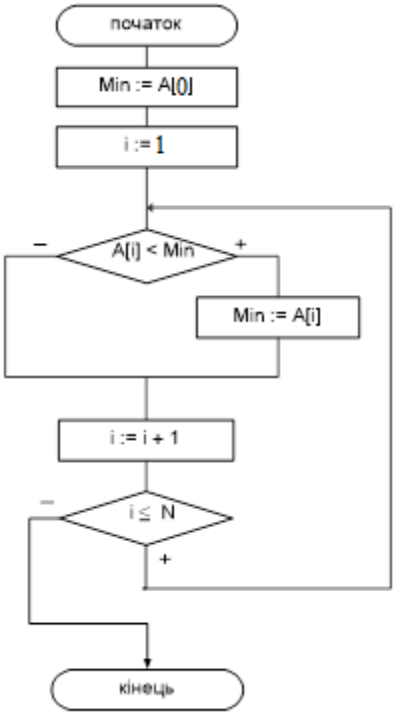
крок 3
початок
Min := A[0]
i := 1
повторити
  якщо *A[i] < Min*
    то
      Min := A[i]
  все якщо
  i := i + 1
  поки *i ≤ N*
  все повторити
кінець

Блок-схема

крок 1

крок 2

крок 3



Перевірка правильності алгоритму

A[10] = { 4, 8,2,7,6,3,1,9,5,3 }

№	***min***	***A[i]***	***A[i] < min***
0	4	8	8<4 false
1	4	2	2<4 true
2	2	7	7< 2 false
3	2	6	6< 2 false
4	2	3	3< 2 false
5	2	1	1< 2 true
6	1	9	9< 1 false
7	1	5	5< 1 false
8	1	3	3< 1 false

Код мовою С

```
#include <iostream>
using namespace std;
int const  N = 10;
int main()
{
    int A[N] = { 4, 8,2,7,6,3,1,9,5,3 },
        i,
        min;

    min = A[0];
     i = 1;
     do
    {
        if (A[i] < min)
        {
            min = A[i];
        }
        i = i + 1;
     } while (i < N);
    cout << "MIN = "<<min;
}
```

Приклад 2. Задано послідовність значень A[n]. Знайти задане значення в послідовності

Математична постановка

Для вирішення поставленої задачі використаємо алгоритм бінарного пошуку. Для цього знайдемо середній елемент послідовності та визначимо частину послідовності для подальшого пошуку. Програмні специфікації напишемо у псевдокоді та графічній формі у вигляді блок-схеми.

Крок 1. Визначимо основні дії.

Крок 2. Деталізуємо дію ініціалізації початкових значень для пошуку

Крок 3. Деталізуємо дію перебору та перегляду елементів послідовності з використанням оператора повторення.

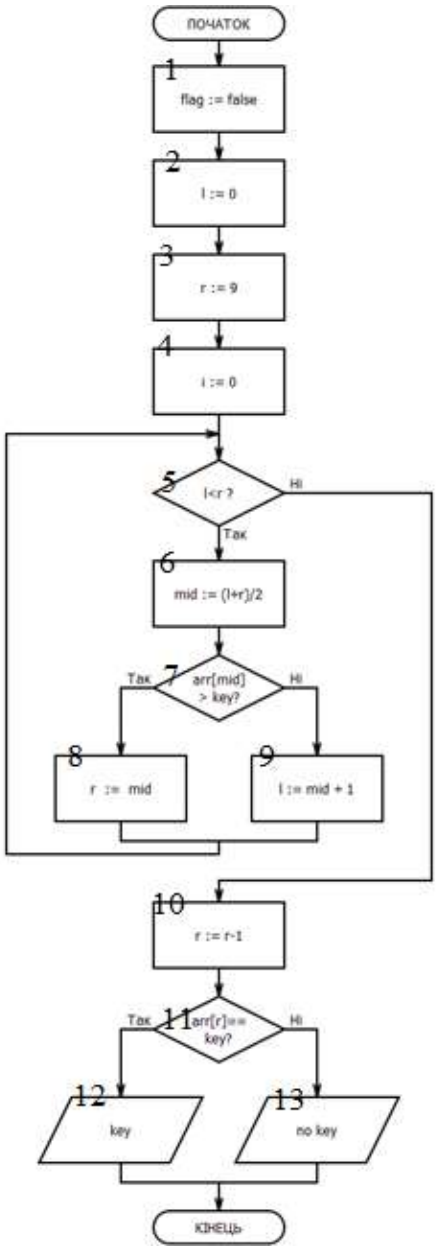
Псевдокод

крок 1
початок
Ініціалізація початкових значень для пошуку
Перебір та перегляд елементів послідовності
кінець

крок 2
початок
flag = false;
l = 0
r = 9
Перебір та перегляд елементів послідовності
кінець

крок 3
початок
flag = false;
l = 0;
r = 9;
поки *(l < r)*
повторити
mid = (l + r) / 2
якщо *(arr[mid] > key)*
то
r = mid
else
l = mid + 1
все якщо
все повторити
r=r-1
кінець

Блок схема алгоритму (крок 3)



Перевірка правильності алгоритму

Відсортований масив:

12 34 56 66 69 71 75 78 81 100
ключ: 81
Індекс елементу 81 в масиві =: 8
ключ: 67
Елемент відсутній

Код програми мовою С

```
#include <iostream>
#include <algorithm>
using namespace std;

int main() {
    setlocale(LC_ALL, "ukr");

    int arr[10];
    int key;
    cout << "Введіть 10 цілих чисел значень: " << endl;

    for (int i = 0; i < 10; i++) {
        cin >> arr[i];
    }
    sort(arr, arr + 10);
    cout << endl << "Введіть ключ: ";
    cin >> key;
    bool flag = false;
    int l = 0;
    int r = 9;
    int mid;

    while (l < r) {
        mid = (l + r) / 2;
        if (arr[mid] > key)
            r = mid;
        else l = mid + 1;
    }
    r--;

    if (arr[r] == key)
        cout << "Індекс елементу " << key << " в масиві =: " << r;
    else cout << "Елемент відсутній";
    return 0;
}
```

Індивідуальне завдання

1. Виконати завдання згідно варіанту (табл.10)
2. Декомпозувати задачу, описати постановку.
3. Програмну специфікацію навести у псевдокоді та графічній формі.
4. Перевірити правильність алгоритму.
5. Написати програму мовою С, використовуючи, як мінімум 2 алгоритми пошуку, яка складається з наступних дій:

– опису трьох змінних індексованого типу з 10 символьних значень;
– ініціювання двох змінних виразами згідно з варіантом (табл. 1);
– ініціювання третьої змінної рівними значеннями двох попередніх змінних;
– обробки третьої змінної згідно з варіантом.

6. Порівняти роботу алгоритмів пошуку.

Таблиця 10. Варіанти завдань

№	Вираз для обчислення елемента		Знайти
	1-го масиву	2-го масиву	
1	117 + i	127 – 2 * i	Два максимальних елементи та їх суму
2	5 * i + 30	60 - 5 * i	Добуток елементів, коди яких менше 40
3	55 - 2 * i	40 + 3 * i	Знайти значення, яке є мінімальним та, більшим за задане P
4	2 * i + 23	49 - 2 * i	Різницю між кодами максимального та мінімального елементів
5	120 - i	110 + i	Кількість елементів, коди яких менші за 115
6	73 - i	64 + 2 * i	Кількість елементів між максимальним та мінімальним елементами

№	Вираз для обчислення елемента		Знайти
	1-го масиву	2-го масиву	
7	95 - 3 * i	74 + 3 * i	Середнє арифметичне елементів, коди яких менші за 82
8	45 + 2 * i	61 – 2 * i	Середнє арифметичне елементів, коди яких більші за 55
9	i * i + 76	85 - i	Кількість елементів, коди яких діляться на 5
10	100 + i	110 – i * i	Добуток кодів елементів
11	2 * i + 40	52 - 2 * i	Елемент, який має максимальний код
12	95 + i	105 - i	Суму елементів, коди яких більші за 101
13	62 + 3 * i	74 - i	Суму кодів мінімального та максимального елементів
14	i + 58	63 - i	Елемент, який має мінімальний код
15	43 - i	37 + i	Добуток елементів, коди яких більші за 40
16	115 + i	125 – 2 * i	Суму двох максимальних елементів
17	5 * i + 25	55 - 5 * i	Добуток елементів, коди яких менші за 82
18	60 - 2 * i	40 + 3 * i	Перше входження елементу з кодом 52
19	2 * i + 22	48 - 2 * i	Суму кодів між максимальним та мінімальним елементами
20	130 - i	120 + i	Кількість елементів, коди яких менші за 127
21	74 - i	64 + 2 * i	Суму двох мінімальних елементів
22	92 - 3 * i	71 + 3 * i	Середнє арифметичне елементів, коди яких менші за 82
23	35 + 3 * i	56 – 2 * i	Середнє арифметичне елементів, коди яких більші за 38
24	i * i + 76	80 - i	Суму парних кодів елементів
25	100 + i	110 – i * i	Добуток кодів елементів, більших за 100
26	2 * i + 42	54 - 2 * i	Кількість елементів, які менші за максимальний код

№	Вираз для обчислення елемента		Знайти
	1-го масиву	2-го масиву	
27	95 + i	105 - i	Суму елементів, коди яких більші за 101
28	66 + 3 * i	78 - i	Суму кодів мінімального та максимального елементів
29	i + 58	63 - i	Елементи, які менші за середньоарифметичне значення
30	43 - i	37 + i	Добуток елементів, коди яких більші за 40
31	120 - i	110 + i	Замінити максимальний елемент середньоарифметичним
32	74 - i	65 + 2 * i	Кількість елементів, коди яких менші за 67
33	93 - 3 * i	72 + 3 * i	Суму непарних кодів елементів
34	44 + 2 * i	55 – 2 * i	Кількість елементів, коди яких діляться на 3
35	i * i + 74	78 - i	Суму двох мінімальних елементів

Вимоги до звіту

Звіт повинен містити:

- титульний аркуш;
- назву та мету роботи;
- варіант;
- постановку задачі;
- побудову математичної моделі;
- псевдокод алгоритму;
- блок схема алгоритму;
- перевірка правильності алгоритму;
- код програми;
- висновки.

Критерії оцінювання

Критерії оцінювання в процентах від максимального балу:

- постановка задачі - 10%;
- побудова математичної моделі – 10%
- псевдокод алгоритму - 20%;
- блок схема алгоритму – 20%
- перевірка правильності алгоритму - 15%
- код програми – 20 %;
- висновки - 5%.

Контрольні запитання

1. У чому полягає ідея лінійного пошуку? Сформулюйте умову припинення перегляду елементів.
2. У якому випадку може використовуватися бінарний пошук? У чому його основна ідея?
3. Сформулюйте алгоритм бінарного пошуку.
4. Що таке інтерполяційний пошук? Сформулюйте алгоритм перегляду елементів.
5. Опишіть переваги та недоліки різних алгоритмів пошуку.

Лабораторна робота 7

АЛГОРИТМИ СОРТУВАННЯ ПОСЛІДОВНОСТЕЙ

Мета – дослідити алгоритми сортування, набути практичних навичок використання цих алгоритмів під час складання програмних специфікацій.

Основні теоретичні відомості

Матриця розглядається як іменована сукупність декількох послідовностей значень, де кожен елемент має два порядкові номери (індекси): перший – номер послідовності, другий – номер елементу в обраній послідовності.

Для обробки значень елементів матриці використовуються два оператори повторення: один вкладений в другий. За допомогою змінної i можна звернутися до послідовності, а змінної j – до елемента послідовності, наприклад A[i,j] (рис.12).

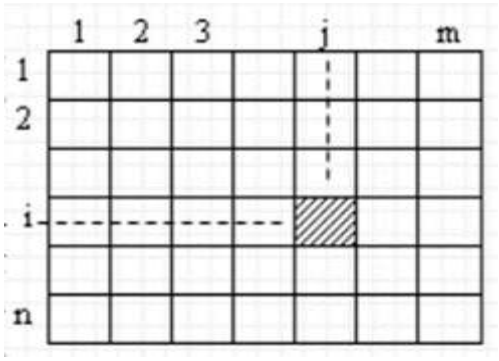


Рисунок 12 - Матриця, де i – номер послідовності (рядок), j – номер елементу (стовбець). Елемент A[i,j] знаходиться на перетині рядка та стовбця.

Задачі на перетворення матриць або послідовностей передбачають використання операторів повторення та їх з’єднання. Будь-якій операції перетворення передує виконання лінійного пошуку.

Сортування послідовностей – це процес упорядкування елементів послідовностей за певним критерієм. Будь-яке сортування передбачає використання вкладених дій повторення.

Розглянемо найпростіші алгоритми бульбашки та вибірки на прикладі сортування послідовності A[n] за зростанням:

Алгоритм бульбашки

```
початок
k := 1
поки k ≤ N                          // кількість проходжень по послідовності
   повторити                        // залежить від розмірності послідовності
   i := N                           // перебір елементів послідовності з кінця
   поки i > k                       // до вже впорядкованих елементів
       повторити
       якщо A[i] < A[i-1]   то      // порівняння сусідніх елементів
temp := A[i]                        // якщо правий елемент має більше
A[i] := A[i-1]                      // значення, то обмін їх місцями через
A[i-1] := temp                      // змінну temp
      все якщо
       i := i - 1
    все повторити
   k := k + 1
все повторити
кінець
```

Алгоритм вибірки

```
початок
k := 1                              // кількість проходжень по послідовності
поки k < N
 повторити
   indMin := k                     // запам'ятати індекс найменшого значення
   i := k + 1
   поки i ≤ N                       // перебір елементів послідовності з
     повторити                      // наступного і до кінця
      якщо A[i] < A[indMin]         // порівняння поточного елементу A[i] з
то                                  // найменшим A[indMin]
indMin := i                         // якщо поточний елемент менший, то
     все якщо                       // запам'ятати його індекс
i := i + 1
 все повторити
temp := A[k]                        // після знаходження мінімального
A[k] := A[indMin]                   // значення в невпорядкованій частині
A[indMin] := temp                   // обміняти його місцями з елементом A[k]
    k := k + 1
все повторити
кінець
```

Приклади застосування алгоритмів сортування під час складання програмних специфікацій

Приклад1 Задано множину послідовностей значень A[m,n]. Замінити в кожному стовпці множини A[m,n] додатні значення на суму елементів цього стовпця.

Крок 1. Визначимо основні дії.

Крок 2. Деталізуємо дію перебору стовпців множини послідовностей.

Крок 3. Деталізуємо дію обчислення суми елементів стовпців.

Крок 4. Деталізуємо дію заміщення додатних значень у стовпці на знайдену суму.

Псевдокод

крок 1
```
початок
Перебір стовпців множини послідовностей
Обчислення суми елементів стовпців
Заміщення додатних чисел в стовпці на знайдену суму
кінець
```

крок 2
```
початок
j := 1
поки j ≤ N
   повторити
   Обчислення суми елементів стовпців
   Заміщення додатних значень в стовпці на знайдену суму
   j := j + 1
все повторити
кінець
```

крок 3
```
початок
j := 1
поки j ≤ N
   повторити
   i := 1
   S := 0
   поки i ≤ M
      повторити
      S := S + A[i,j]
      i := i + 1
   все повторити
   Заміщення додатних значень в стовпці на знайдену суму
   j := j + 1
все повторити
кінець
```

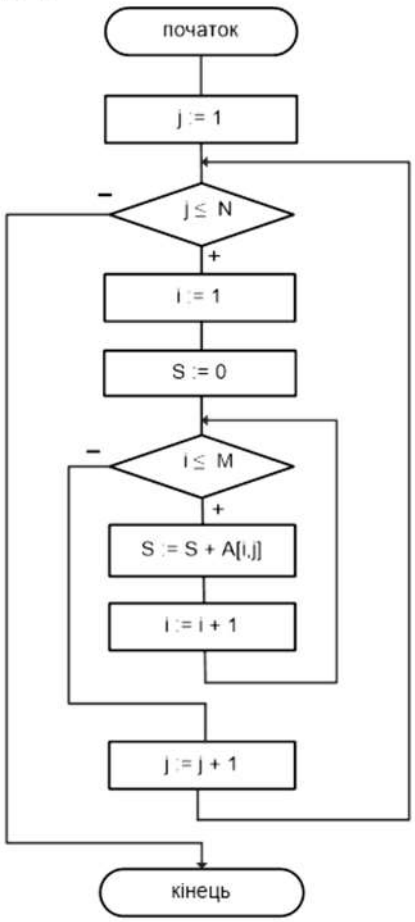
крок 4
```
початок
j := 1
поки j ≤ N
   повторити
   i := 1
   S := 0
   поки i ≤ M
      повторити
      S := S + A[i,j]
      i := i + 1
   все повторити
   i := 1
   поки i ≤ M
      повторити
      якщо A[i,j] · 0 то
         A[i,j] := S
      все якщо
      i := i + 1
   все повторити
   j := j + 1
все повторити
кінець
```

Блок - схема

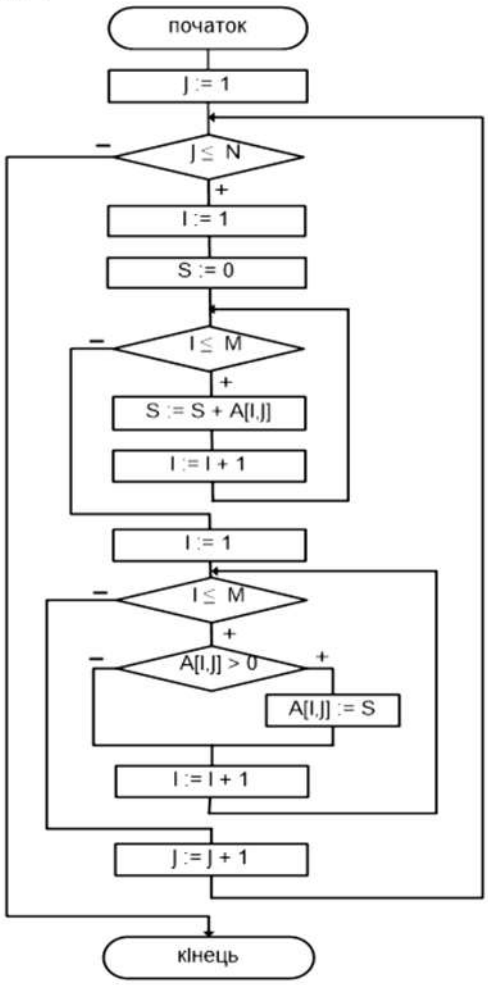
крок 1

крок 2

крок 3


початок
j := 1
j ≤ N
–
+
i := 1
S := 0
i ≤ M
–
+
S := S + A[i,j]
i := i + 1
j := j + 1
кінець

крок 4


початок
J := 1
J ≤ N
–
+
I := 1
S := 0
I ≤ M
–
+
S := S + A[I,J]
I := I + 1
I := 1
I ≤ M
–
+
A[I,J] > 0
–
+
A[I,J] := S
I := I + 1
J := J + 1
кінець

Код програми мовою С

```
#include <iostream>

int main()
{
    const int M = 4;
    const int N = 3;
    int j,
        i,
        S;

    int A[M][N] = { {1,-2,3},
                    {-4,-5,16},
                    {-6,3,-7},
                    {5,-9,2} };

    j = 0;

    while (j <= N)
    {
        i = 0;
        S = 0;
        while (i < M)
        {
            S = S + A[i][j];
            i++;
        }

       i=0;


        while (i<M)
        {
            if (A[i][j] > 0)
                A[i][j] = S;
            i++;
        }
        j++;
    }

}
```

Проміжний результат роботи програми

A	0x004ffc88 {0x004ffc88 {-4, -2, 3}, 0x004ffc9...	int[4][3]
[0]	0x004ffc88 {-4, -2, 3}	int[3]
[1]	0x004ffc94 {-4, -5, 16}	int[3]
[2]	0x004ffca0 {-6, 3, -7}	int[3]
[3]	0x004ffcac {-4, -9, 2}	int[3]
i	4	int
j	0	int
M	4	const int
N	3	const int
S	-4	int

Приклад 2 Задано послідовність A[n]. Сформувати нову послідовність B[m] з додатних значень послідовності A[n] та відсортувати її за зростанням алгоритмом бульбашки.

Розв'язання

Програмні специфікації напишемо у псевдокоді та графічній формі у вигляді блок-схеми.

Крок 1. Визначимо основні дії.

Крок 2. Деталізуємо дію формування послідовності B[m], де m – кількість додатних значень послідовності A[n].

Крок 3. Деталізуємо дію сортування послідовності B[m].

Псевдокод

крок 1
```
початок
Формування послідовності B[m]
Сортування послідовності B[m]
кінець
```

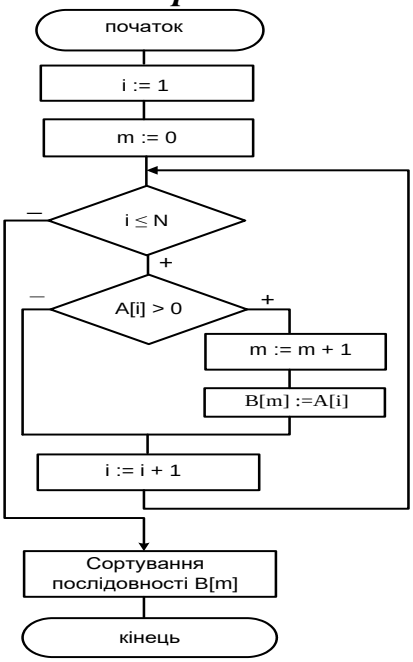
крок 2
```
початок
i := 1
m := 0
поки  i ≤ N
   повторити
   якщо A[i] > 0 то
      m := m + 1
      B[m] := A[i]
   все якщо
   i := i + 1
все повторити
Сортування послідовності B[m]
кінець
```

крок 3
```
початок
i := 1
m := 0
поки  i ≤ N
   повторити
   якщо A[i] > 0 то
      m := m + 1
      B[m] := A[i]
   все якщо
   i := i + 1
все повторити
k := 1
поки k ≤ m
   повторити
   i := m
   поки i < k
       повторити
       якщо A[i] < A [i-1] то
          temp := A[i]
          A[i] := A[i-1]
          A[i-1] := temp
       все якщо
       i:= i-1
   все повторити
   k := k +1
все повторити
кінець
```

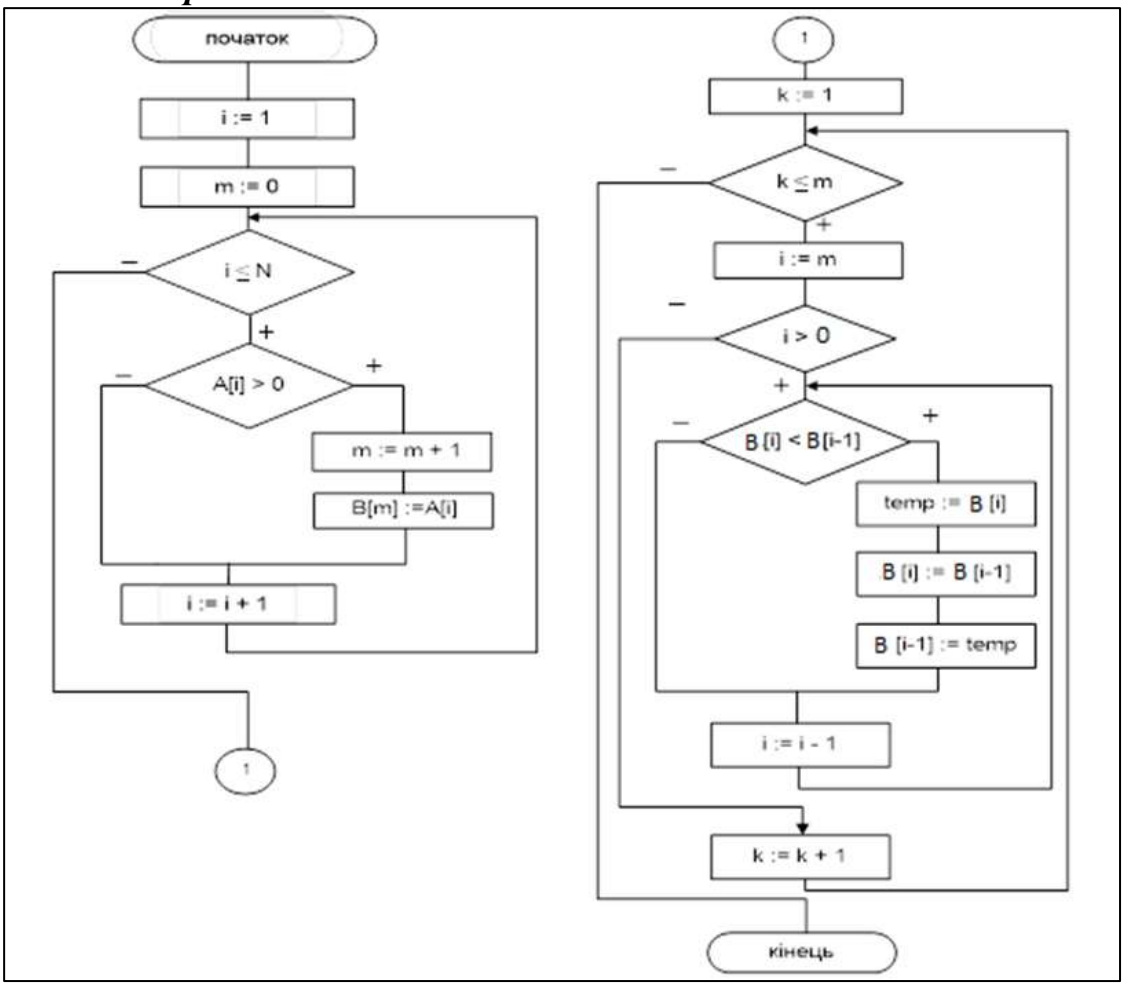
Блок схема

крок 1

крок 2



крок 3



Код мовою С

```
int main()
{

    const int N = 10;
    int m,
        i,
        k,
        temp,
        B[N];

    int A[N] = { 8,11,-14,6,-3,12,-4,3,22,9 };

    i = 0;
    m = 0;

    while (i < N)
    {
        if (A[i] > 0)
        {
            B[m] = A[i];
            m++;
        }
        i++;
    }
    k = 0;
    while (k < m-1)
    {
        i = m - 1;
        while (i > 0)
        {
            if (B[i] < B[i - 1])
            {
                temp = B[i];
                B[i] = B[i - 1];
                B[i - 1] = temp;;
            }
            i--;
        }
        k++;
    }
}
```

Проміжний результат роботи програми, формування послідовності B.

A	0x007af7c4 {8, 11, -14, 6, -3, 12, -4, 3, 22, 9}	int[10]
B	0x007af7f4 {8, 11, 6, 12, 3, 22, 9, -858993460, -858993460, -85899...	int[10]
[0]	8	int
[1]	11	int
[2]	6	int
[3]	12	int
[4]	3	int
[5]	22	int
[6]	9	int
[7]	-858993460	int

Проміжний результат роботи програми, сортування послідовності B за зростанням алгоритмом бульбашки.

A	0x012ff714 {8, 11, -14, 6, -3, 12, -4, 3, 22, 9}	int[10]
B	0x012ff744 {3, 6, 8, 9, 11, 12, 22, -858993460, -858993460, -85899...	int[10]
[0]	3	int
[1]	6	int
[2]	8	int
[3]	9	int
[4]	11	int
[5]	12	int
[6]	22	int
[7]	-858993460	int
[8]	-858993460	int
[9]	-858993460	int
i	0	int

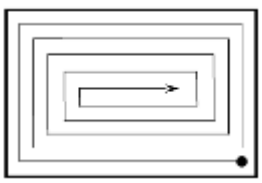
Індивідуальне завдання

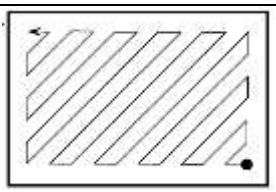
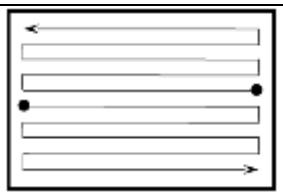
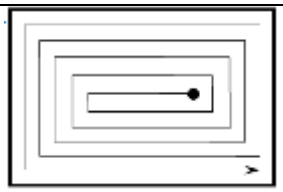
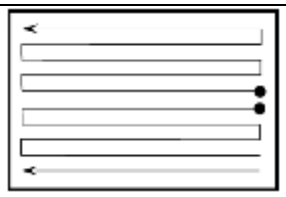
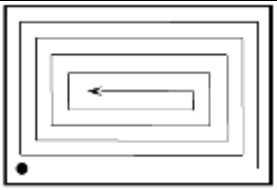
6. Виконати завдання згідно варіанту (табл.11)
7. Декомпозувати задачу, описати постановку.
8. Програмну специфікацію навести у псевдокоді та графічній формі.
9. Написати програму мовою С, яка складається з наступних дій:

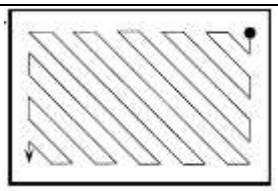
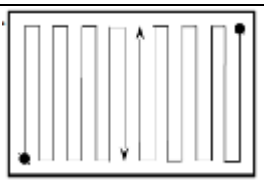
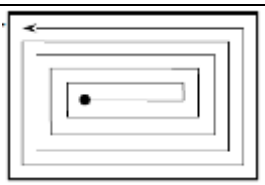
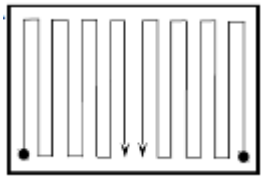
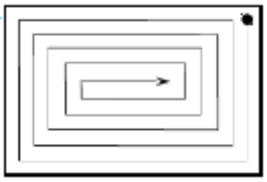
– опис та ініціювання змінної індексованого типу (двовимірний масив) згідно з варіантом (табл. 11);
– створення та ініціювання нової змінної індексованого типу (одновимірний масив), що обчислюються згідно з варіантом (табл. 11).

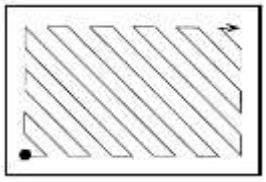
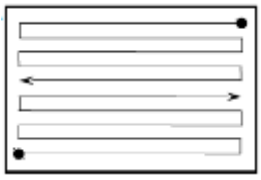
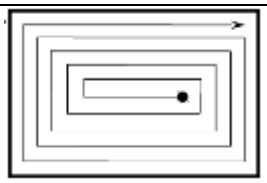
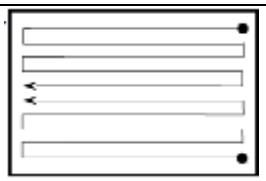
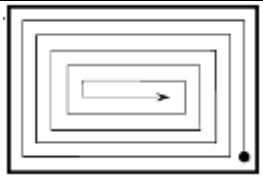
10. Протестувати програму.

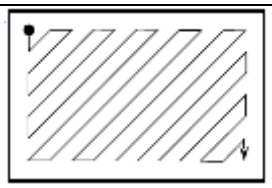
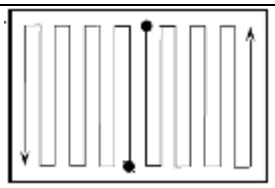
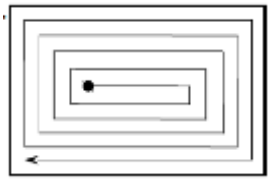
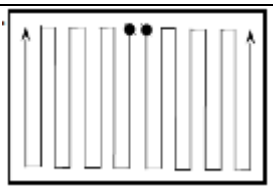
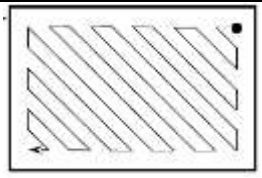
Таблиця 11. Варіанти завдань

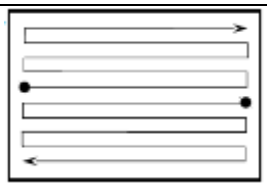
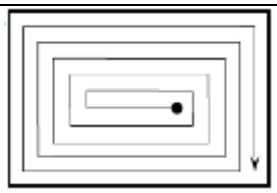
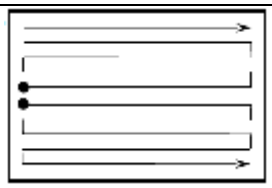
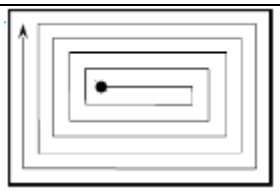
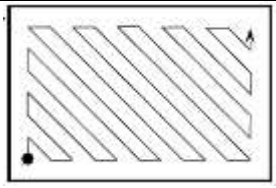
№	Розмір-ність	Тип даних	Спосіб заповнення двовимірного масиву	Обчислення елементів одновимірного масиву
1	5 x 5	Цілий		Із значень елементів головної діагоналі двовимірного масиву. Відсортувати обміном за зростанням.

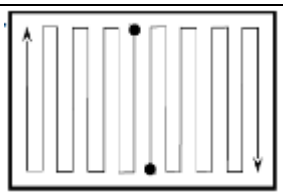
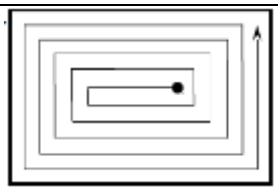
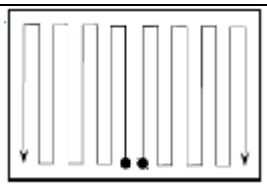
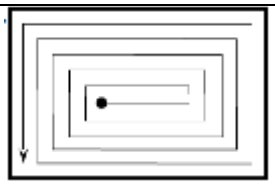
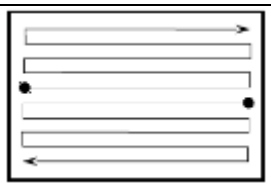
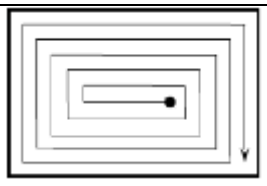
№	Розмір-ність	Тип даних	Спосіб заповнення двовимірного масиву	Обчислення елементів одновимірного масиву
2	6 x 5	Дійсний		Із середнього арифметичного значення елементів рядків двовимірного масиву. Відсортувати обміном за спаданням.
3	6 x 4	Дійсний		Із суми значень елементів рядків двовимірного масиву. Відсортувати методом вставки за зростанням.
4	4 x 4	Цілий		Із мінімальних значень елементів стовпців двовимірного масиву. Відсортувати методом вставки за спаданням.
5	5 x 7	Цілий		Із суми значень елементів стовпців двовимірного масиву. Відсортувати методом бульбашки за зростанням.
6	7 x 5	Дійсний		Із максимальних значень елементів рядків двовимірного масиву. Відсортувати методом бульбашки за спаданням.

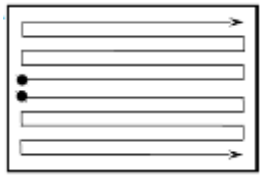
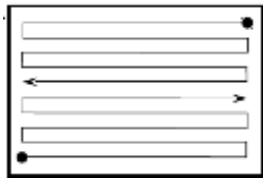
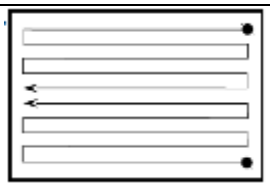
№	Розмір-ність	Тип даних	Спосіб заповнення двовимірного масиву	Обчислення елементів одновимірного масиву
7	8 x 5	Дійсний		Із добутку значень елементів рядків двовимірного масиву. Відсортувати обміном за зростанням.
8	4 x 6	Цілий		Із максимальних значень елементів стовпців двовимірного масиву. Відсортувати обміном за спаданням.
9	6 x 6	Цілий		Із додатних значень елементів головної діагоналі двовимірного масиву. Відсортувати методом Шела за зростанням.
10	4 x 8	Дійсний		Із мінімальних значень елементів стовпців двовимірного масиву. Відсортувати методом Шела за спаданням.
11	7 x 5	Дійсний		Із суми від’ємних значень елементів рядків двовимірного масиву. Відсортувати методом вставки за зростанням.

№	Розмір-ність	Тип даних	Спосіб заповнення двовимірного масиву	Обчислення елементів одновимірного масиву
12	6 x 4	Цілий		Із максимальних значень елементів рядків двовимірного масиву. Відсортувати методом вставки за спаданням.
13	6 x 6	Цілий		Із додатних значень елементів головної діагоналі двовимірного масиву. Відсортувати методом бульбашки за зростанням.
14	4 x 6	Дійсний		Із добутку від’ємних значень елементів стовпців двовимірного масиву. Відсортувати методом бульбашки за спаданням.
15	8 x 5	Дійсний		Із суми додатних значень елементів рядків двовимірного масиву. Відсортувати обміном за зростанням.
16	5 x 7	Цілий		Із середнього арифметичного від’ємних значень елементів стовпців двовимірного масиву. Відсортувати обміном за спаданням.

№	Розмір-ність	Тип даних	Спосіб заповнення двовимірного масиву	Обчислення елементів одновимірного масиву
17	6 x 5	Цілий		Із суми додатних значень елементів рядків двовимірного масиву. Відсортувати методом вставки за зростанням.
18	5 x 5	Дійсний		Із від’ємних значень елементів побічної діагоналі двовимірного масиву. Відсортувати методом вставки за спаданням.
19	4 x 8	Дійсний		Із добутку значень елементів стовпців двовимірного масиву. Відсортувати методом Шела за зростанням.
20	5 x 4	Цілий		Із середнього арифметичного від’ємних значень елементів рядків двовимірного масиву. Відсортувати методом Шела за спаданням.
21	8 x 5	Цілий		Із добутку додатних значень елементів рядків двовимірного масиву. Відсортувати обміном за зростанням.

№	Розмір-ність	Тип даних	Спосіб заповнення двовимірного масиву	Обчислення елементів одновимірного масиву
22	5 x 7	Дійсний		Із середнього арифметичного додатних значень елементів стовпців двовимірного масиву. Відсортувати обміном за спаданням.
23	8 x 4	Дійсний		Із добутку від'ємних значень елементів рядків двовимірного масиву. Відсортувати методом вставки за зростанням.
24	7 x 7	Цілий		Із значень елементів побічної діагоналі двовимірного масиву. Відсортувати методом вставки за спаданням.
25	5 x 8	Цілий		Із добутку додатних значень елементів стовпців двовимірного масиву. Відсортувати методом бульбашки за зростанням.
26	6 x 5	Дійсний		Із середнього арифметичного від'ємних значень елементів рядків двовимірного масиву. Відсортувати методом бульбашки за спаданням.

№	Розмір-ність	Тип даних	Спосіб заповнення двовимірного масиву	Обчислення елементів одновимірного масиву
27	8 x 4	Дійсний		Із добутку від'ємних значень елементів рядків двовимірного масиву. Відсортувати обміном за зростанням.
28	5 x 5	Цілий		Із додатних значень елементів головної діагоналі двовимірного масиву. Відсортувати обміном за спаданням.
29	5 x 8	Цілий		Із мінімальних значень елементів стовпців двовимірного масиву. Відсортувати методом Шела за зростанням.
30	7 x 6	Дійсний		Із добутку від'ємних значень елементів рядків двовимірного масиву. Відсортувати методом Шела за спаданням.
31	8 x 4	Дійсний		Із добутку значень елементів рядків двовимірного масиву. Відсортувати методом Шела за зростанням.
32	5 x 8	Цілий		Із середнього арифметичного від'ємних значень елементів стовбців

№	Розмір-ність	Тип даних	Спосіб заповнення двовимірного масиву	Обчислення елементів одновимірного масиву
				двовимірного масиву. Відсортувати методом Шела за спаданням.
33	8 x 7	Цілий		Із добутку додатних значень елементів рядків двовимірного масиву. Відсортувати обміном за зростанням.
34	5 x 7	Дійсний		Із середнього арифметичного додатних значень елементів стовпців двовимірного масиву. Відсортувати обміном за спаданням.
35	8 x 4	Дійсний		Із добутку від’ємних значень елементів рядків двовимірного масиву. Відсортувати методом вставки за зростанням.

Вимоги до звіту

Звіт повинен містити:

- титульний аркуш;
- назву та мету роботи;
- варіант;
- постановку задачі;
- побудову математичної моделі;
- псевдокод алгоритму;

- блок схема алгоритму;
- код програми;
- тестування програми;
- висновки.

Критерії оцінювання

Критерії оцінювання в процентах від максимального балу:

- постановка задачі - 15%;
- побудова математичної моделі – 15%
- псевдокод алгоритму - 20%;
- блок схема алгоритму – 20%
- код програми – 20 %;
- тестування програми – 5%;
- висновки - 5%.

Контрольні запитання

1. Що таке сортування?
2. За якими ознаками можна класифікувати алгоритми сортування?
3. В чому полягає принцип сортування вибіркою?
4. В чому полягає принцип сортування бульбашкою?
5. Опишіть переваги та недоліки цих алгоритмів.
6. Класифікація масивів.
7. Що означає термін «обхід двовимірного масиву»?
8. Скільки циклів та індексних змінних потрібно для виконання обходу двовимірного масиву?
9. Опишіть спосіб простого обходу масиву по рядках.
10. Опишіть спосіб простого обходу масиву по стовпчиках.

ПЕРЕЛІК ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. *Томас Г. Кормен Чарлз Е. Лейзерсон Роналд Л. Рівест Кліфорд Стайн.* Вступ до алгоритмів. — К.І.С, 2023. — 1288 с.
2. *Wirth* N. Algorithms and Data Structures. — Oberon 2004. - 360 p.
3. *Stephens* R. Essentials Algorithms: A Practical Approach to Computer Algorithms. — John Wiley & Sons, Inc., Indianapolis, Indiana 2013. — 544 p.
4. *Roughgarden* T. Algorithms Illuminated Part 1: The Basics — Soundlikeyourself Publishing, LLC, 2017. — 256 p.
5. *Sedgewick* R., *Wayne* K. Algorithms (4th Edition). — Addison-Wesley Professiona, 2011. — 976 p.
6. *Вітковська* І. Курс лекцій з дисципліни «Алгоритми та структури даних 1. Основи алгоритмізації».[Електронний ресурс]. – Режим доступу: https://do.ipo.kpi.ua/course/view.php?id=4786#section-1
7. *Вітковська* І. Методичні вказівки до виконання лабораторних робіт з дисципліни «Алгоритми та структури даних 1. Основи алгоритмізації».[Електронний ресурс]. – Режим доступу: https://do.ipo.kpi.ua/course/view.php?id=4786#section-2
8. Головна сторінка MSDN [Електронний ресурс]. – Режим доступу: http://msdn.microsoft.com

ДОДАТОК А

ТИТУЛЬНИЙ АРКУШ ЗВІТУ

Міністерство освіти і науки України
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи № 1 з дисципліни
«АЛГОРИТМИ ТА СТРУКТУРИ ДАНИХ-1.
ОСНОВИ АЛГОРИТМІЗАЦІЇ»

«ЛІНІЙНІ АЛГОРИТМИ»

Варіант_____

Виконав студент _____
(шифр, прізвище, ім'я, по батькові)

Перевірив викладач _____
(прізвище, ім'я, по батькові)

Київ 20__