МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**Український державний університет
науки і технологій**

Кафедра «Комп'ютерні інформаційні технології»

В авторській редакції

ОСНОВИ ПРОГРАМУВАННЯ

Навчально-методичні рекомендації до
виконання лабораторних робіт

Електронне видання

ДНІПРО
2025

УДК 004.42(076.5)
О 75

Упорядники:
доц. *Горбова Олександра Вікторівна*

Електронне видання

Схвалено Групою забезпечення якості освітньої програми
121 «Інженерія програмного забезпечення»
Протокол № 7 від 04.12.2024

О 75 Основи програмування : навчально-методичні рекомендації до виконання лабораторних робіт / упоряд. О. В. Горбова ; Укр. держ. ун-т науки і технологій. – Електрон. вид. – Дніпро : УДУНТ, 2025. – 68 с.

Навчально-методичні рекомендації призначені для використання студентами денної форми навчання освітнього ступеня «бакалавр» за ОПП «Інженерія програмного забезпечення» спеціальності 121 «Інженерія програмного забезпечення» під час виконання лабораторних робіт з дисципліни «Основи програмування».

Навчально-методичні рекомендації містять короткі теоретичні відомості для лабораторних робіт та порядок їх виконання, призначені для теоретичної підготовки до виконання лабораторних робіт. Подано варіанти індивідуальних завдань, контрольні питання та завдання для самоконтролю.

Іл. 10. Табл. 18. Бібліогр.: 15 назв.

ЗМІСТ

ПЕРЕДМОВА

Основи програмування є обов'язковою дисципліною у підготовці фахівців зі спеціальності 121 «Інженерія програмного забезпечення». У курсі навчальної дисципліни «Основи програмування» розглядаються загальні питання з алгоритму та алгоритмізації задач для вирішення їх на комп'ютері, високо рівневі мови програмування, на прикладі мови C/C++, оператори мови C/C++, основні та складені типи даних, функції, робота з файлами, створення багатофайлового проєкту та рекурсія.

Метою дисципліни є досягнення компетентностей, які основані на зазначених в освітньо-професійній програмі:

1. К01. Здатність до абстрактного мислення, аналізу та синтезу.
2. К02. Здатність застосовувати знання у практичних ситуаціях.
3. К05. Здатність вчитися і оволодівати сучасними знаннями.
4. К06. Здатність до пошуку, оброблення та аналізу інформації з різних джерел.
5. К13. Здатність ідентифікувати, класифікувати та формулювати вимоги до програмного забезпечення.
6. К14. Здатність брати участь у проектуванні програмного забезпечення, включаючи проведення моделювання (формальний опис) його структури, поведінки та процесів функціонування.
7. К15. Здатність розробляти архітектури, модулі та компоненти програмних систем.
8. К19. Володіння знаннями про інформаційні моделі даних, здатність створювати програмне забезпечення для зберігання, видобування та опрацювання даних.
9. К-20. Здатність застосовувати фундаментальні і міждисциплінарні знання для успішно-го розв'язання завдань інженерії програмного забезпечення.
10. К-26. Здатність до алгоритмічного та логічного мислення.

Метою даного видання є пояснення побудови алгоритмів до поставлених задач, створення програм, мовою C/C++, згідно створених алгоритмів, при використанні основних конструкцій та функцій, основ тестування чорною скринькою та принципів налагодження програм з використанням інструментарію IDE MS Visual Studio. Подаються приклади написання програм та розробки структурних схем. Це сприяє покращенню розуміння принципів розробки, тестування та налагодження програм.

Використання даного видання та виконання описаних лабораторних робіт сприяє досягненню результатів навчання, передбачених програмною дисципліни, а саме:

– ПР05. Знати і застосовувати відповідні математичні поняття, методи доменного, системного і об'єктно-орієнтованого аналізу та математичного моделювання для розробки програмного забезпечення.

– ПР06. Уміння вибирати та використовувати відповідну задачі методологію створення програмного забезпечення.

– ПР07. Знати і застосовувати на практиці фундаментальні концепції, парадигми і основні принципи функціонування мовних, інструментальних і обчислювальних засобів інженерії програмного забезпечення.

– ПР11. Вибирати вихідні дані для проектування, керуючись формальними методами опису вимог та моделювання.

– ПР13. Знати і застосовувати методи розробки алгоритмів, конструювання програмного забезпечення та структур даних і знань.

– ПР23. Вміти документувати та презентувати результати розробки програмного забезпечення.

ЛАБОРАТОРНА РОБОТА № 1. АЛГОРИТМІЗАЦІЯ ЗАДАЧ. РОЗРОБКА ТА ВИКОНАННЯ ПРОГРАМ НА МОВІ С++.

Тема. Алгоритмізація задач. Розробка та виконання програм на мові С++.

Мета роботи. Ознайомитися з процесом розробки програм. Отримати навички розробки алгоритмів різних структур. Отримати навички розробки консольного додатка на мові С++ у середовищі Microsoft Visual Studio.

Зміст роботи:

1. ознайомитися з теоретичними відомостями і додатковими матеріалами до лабораторної роботи;
2. для задач A, B і C обраного індивідуального варіанту завдання:
   a) розробити алгоритм і тести для перевірки правильності його роботи;
   b) створити проект в Microsoft Visual Studio C++, набрати текст програми;
   c) здійснити компіляцію програми (якщо будуть виявлені помилки компіляції, то усунути їх);
   d) виконати програму, використовуючи розроблені тести (див. додаток А);
   e) переконатися в правильності реалізації завдання. У разі потреби, виправити помилки та повторити компіляцію і виконання програми;
3. оформити звіт з лабораторної роботи.

Зміст звіту:

1) вид та номер роботи, дисципліна, П.І.Б. та група виконавця;
2) тема, мета лабораторної роботи;
3) для задач A, B і C:
   a) постановка задач згідно з загальним та індивідуальним завданнями;
   b) опис алгоритму розв’язання завдання;
   c) опис програми, вхідних і вихідних даних;
   d) текст програми;
   e) тести для перевірки правильності програми;
   f) результати виконання програми та їх аналіз;
4) аналіз результатів та висновки щодо розробки програми.

Теоретичні відомості

Етапи розв’язування задач на комп’ютері [3]**.** Створення комп'ютерної програми – багатоетапний процес. Він включає в себе послідовність дій від постановки завдання до отримання рішення.

Загальне формулювання завдання. На цьому етапі завдання формулюється в змістовних термінах, визначаються вхідні і вихідні дані.

Математичне формулювання завдання. На цьому етапі визначаються математичні величини, які будуть описувати завдання, а також математичні зв'язки між ними, т. е. складається математична модель.

Вибір методу рішення. Одну й ту саму задачу може бути розв’язано за допомогою різних методів. Вибір методу визначається багатьма чинниками,

основними з яких є точність результатів, час розв’язування, обсяг займаної оперативної пам’яті.

Складання алгоритму рішення. Розробляється оптимальна логічна послідовність дій з урахуванням обраного методу рішення для здобуття певних результатів.

Складання програми. Написання та введення розробленої програми алгоритмічною мовою програмування в комп’ютер.

Налагодження програми. Пошук і виправлення можливих синтаксичних та алгоритмічних помилок у програмі.

Тестування програми. Відбувається підтвердження або спростування правильності роботи алгоритму. Для цього, як правило, вирішуються завдання з такими початковими даними, для яких відоме достовірне рішення.

Ця послідовність є притаманна для розв’язування якої завгодно задачі у програмний спосіб. Однак при підготовці задачі кожен етап може мати більш чи менш виражений характер.[3]

Алгоритм, його властивості, засоби описування [1]. Алгоритм (algorithm) – це система формальних правил, які чітко й однозначно окреслюють послідовність дій обчислювального процесу від початкових даних до шуканого результату.

Розробити алгоритм рішення означає розбити завдання на послідовно виконувані етапи. Алгоритм описує процес перетворення початкових даних в результати, оскільки для вирішення будь-якого завдання необхідно:

1) ввести початкові дані;
2) перетворити початкові дані в результати (вихідні дані);
3) вивести результати.

Основні властивості алгоритмів:

– детермінованість (визначеність) – однозначність результату обчислювального процесу за заданих початкових даних;

– дискретність – поділ обчислювального процесу на окремі елементарні кроки, виконувані послідовно;

– зрозумілість – усі дії, включені до алгоритму, мають бути у межах компетенції виконавця алгоритму.

– результативність – забезпечення здобуття розв’язку через певну кінцеву кількість кроків;

– масовість – забезпечення розв’язання якої завгодно задачі з класу однотипних.

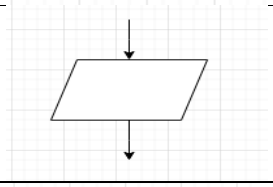
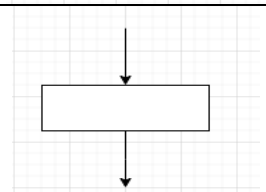
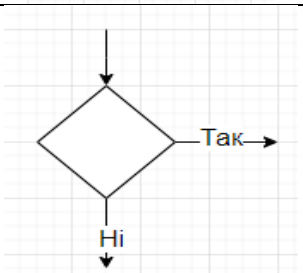
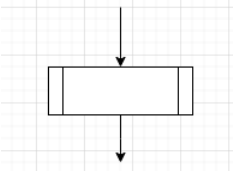
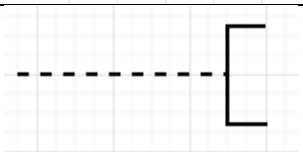
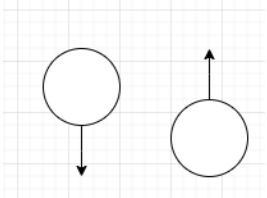
Задавати алгоритми можна у різні способи:

– словесне описування послідовності дій;
– графічне подавання (блок-схеми, діаграми Нассі-Шнейдермана);
– записування алгоритмічною мовою програмування.

Графічні засоби описування алгоритмів. Блок-схема [1] – опис структури алгоритму за допомогою геометричних фігур з лініями-зв'язками, що показують порядок виконання окремих інструкцій. У блок-схемі кожному типу дій (введенню початкових даних, обчисленню значень виразів, закінченню

обробки і т. п.) відповідає геометрична фігура, що називається блоком. Інформація у блоках записується на природній мові або мові математичних формул. Блоки з'єднуються лініями переходів (стрілками), що визначають черговість виконання дій. Якщо лінії не мають зламів, стрілками їх можна не позначати. В інших випадках їхній напрямок обов'язково позначають стрілкою. Лінію потоку зазвичай підводять до середини блока. Правила побудови алгоритмів визначені відповідними стандартами: блок-схема вибудовується в одному напрямі: або зверху вниз, або зліва направо, в порядку виконання дій. Операції різного типу зображуються в схемі різними геометричними фігурами (табл. 1).

Таблиця 1. Основні види блоків, їх призначення

Графічне зображення	Назва, дія, що виконується
	Початок/кінець алгоритму
	Блок введення/виведення
	Виконання обчислень або присвоєння значень
	Блок перевірки умов
	Виклик, раніше створених блоків програми.
	Коментар
	Поєднувач блоків

Діаграми Нассі-Шнейдермана (N-S-діаграми) [12] – це схеми, що ілюструють передачу управління в алгоритмі за допомогою вкладених один в одного блоків. N-S-діаграми більш компактні за рахунок відсутності явної вказівки ліній переходу по управлінню. Кожен блок має форму прямокутника і може бути вписаний у будь-який внутрішній прямокутник будь-якого іншого блоку.

Види алгоритмів [1]**.** Алгоритми можна представляти як схеми, що складаються з окремих базових елементів. Ці елементи об'єднуються в алгоритмічні конструкції. Залежно від особливостей своєї побудови алгоритми можна розділити на види:

1) лінійні (послідовні);
2) що розгалужуються;
3) циклічні;
4) рекурсивні.

Різноманітність алгоритмів визначається тим, що будь-який алгоритм складається з фрагментів, кожен з яких є алгоритмом одного з вказаних видів. Тому важливо знати структуру кожного з алгоритмів і принципи їх складання.

Лінійний алгоритм (послідовне виконання, структура слідування) – це алгоритм, який забезпечує отримання результату шляхом одноразового виконання послідовності дій, незалежно від вхідних даних і проміжних результатів. Дії в таких алгоритмах виконуються послідовно, одна за однією, тобто лінійно.

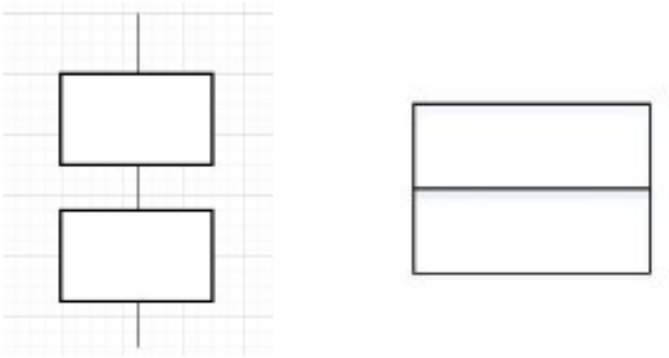
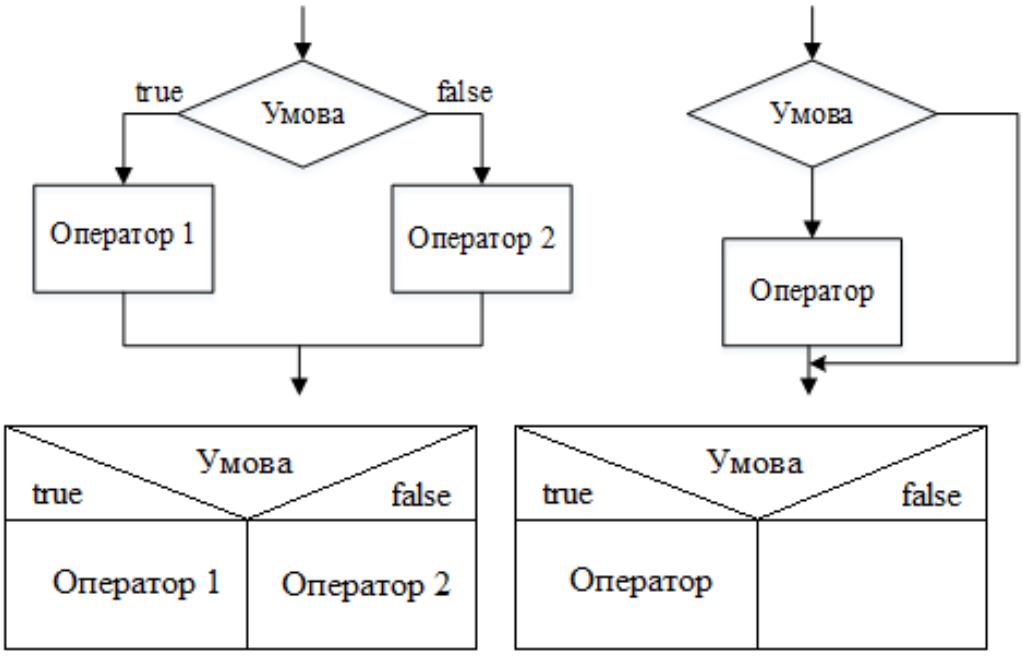


Рис. 1. Елементи блок-схеми та діаграми Нассі-Шнейдермана
для реалізації лінійних алгоритмів

Розгалужений алгоритм (умова, структура вибору) – у класичному варіанті ця структура розглядається як вибір дій у разі виконання або невиконання заданої умови. Розгалуження бувають повними і неповними. Повне розгалуження – це розгалуження, в якому певні дії визначені й у разі виконання, і в разі невиконання умови. Неповне розгалуження – це розгалуження, в якому дії визначені тільки у разі виконання (або у разі невиконання) умови (Див. рис. 2).

Циклічний алгоритм (цикл, структура повторення) – це алгоритм, у якому передбачено повторення деякої серії команд. За допомогою цієї структури описуються однотипні дії, що повторюються декілька разів. Такі алгоритми забезпечують виконання довгої послідовності дій, записаних порівняно короткою послідовністю команд.(Див. рис. 2).



а) б)

Рис. 2. Елементи блок-схеми та діаграми
Нассі-Шнейдермана для реалізації розгалужених алгоритмів
а – повна альтернатива; б – неповна альтернатива

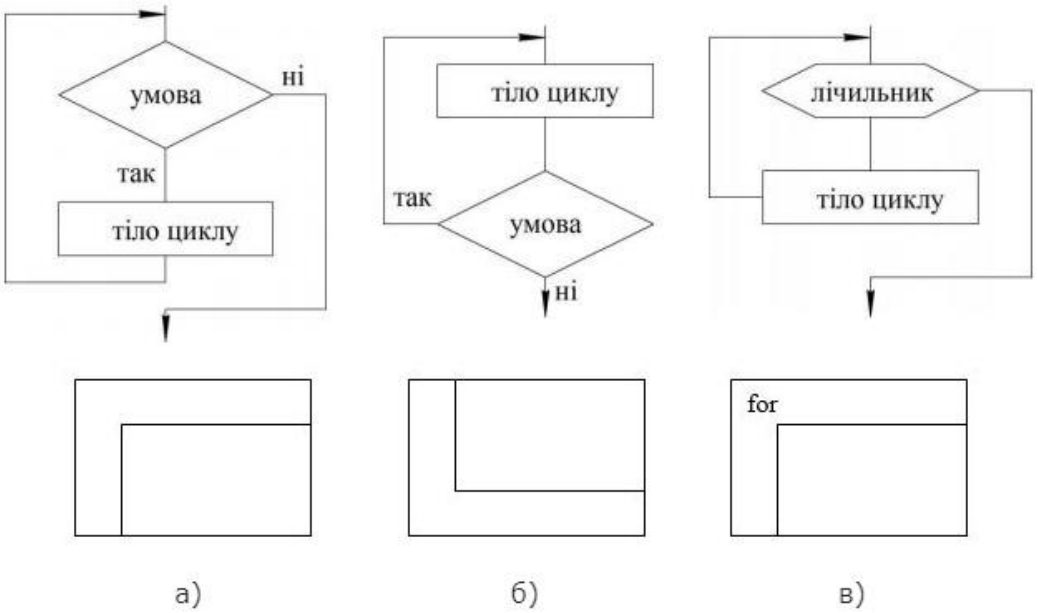


Рис. 3. Елементи блок-схеми та діаграми Нассі-Шнейдермана
для реалізації циклічних алгоритмів:
а – з передумовою, б – з післяумовою, в – з лічильником

Основна особливість базових алгоритмічних структур – це їх повнота, тобто цих структур достатньо для створення найскладнішого алгоритму.

Тестування алгоритмів [1]**.** Процес тестування можна розділити на три етапи:

1) перевірка в нормальних умовах (передбачає тестування на основі даних, які характерні для реальних (типових) умов функціонування алгоритму);

2) перевірка в екстремальних умовах (тестові дані включають граничні значення області зміни вхідних даних, які повинні сприйматися алгоритмом як правильні дані; типовими прикладами таких значень є дуже маленькі або дуже великі числа або відсутність даних);

3) перевірка у виняткових ситуаціях (проводиться з використанням даних, значення яких лежать за межами допустимої області значень).

Найгірша ситуація складається тоді, коли алгоритм сприймає невірні дані як правильні і видає невірний, але правдоподібний результат. Алгоритм повинен відкидати будь-які дані, які він не в змозі обробляти правильно.

Тестові дані повинні забезпечити перевірку всіх можливих умов виникнення помилок:

– повинна бути випробувана кожна гілка алгоритму;

– черговий тест повинен контролювати щось таке, що ще не було перевірено на попередніх тестах;

– перший тест повинен бути максимально простий, щоб перевірити, чи працює алгоритм взагалі;

– арифметичні операції в тестах повинні максимально спрощуватися для зменшення обсягу обчислень;

– мінімізація обчислень не повинна знижувати надійності контролю;

– тестування повинно бути цілеспрямованим і систематизованим, так як випадковий вибір вихідних даних привів би до:

1) труднощів у визначенні ручним способом очікуваних результатів;

2) того, що неперевіреними можуть виявитися багато ситуацій;

– ускладнення тестових даних має відбуватися поступово.

Приклад. Складемо таблицю (табл. 2) з тестами для перевірки алгоритму для задачі знаходження коренів квадратного рівняння $ax2 + bx + c = 0$.

Таблиця 2. Тестування алгоритму

№	Назва тесту (мета тестування)	Вхідні дані			Очікувані результати
		a	b	c	
1	Два різних кореня (D > 0)	1	1	-2	x1 = 1, x2 = -2
2	Корні рівні (D = 0)	1	2	1	x1 = x2 = -1
3	Дійсних коренів немає ((D < 0)	2	1	2	Дійсних коренів немає
4	Всі коефіцієнти нульові	0	0	0	x – будь-яке число
5	Неправильне рівняння (a = b = 0, c ≠ 0)	0	0	5	Коренів немає (неправильне рівняння)
6	Лінійне рівняння (a = 0, b ≠ 0)	0	3	6	x = -2
7	Вільний член рівняння нульовий (a ≠ 0, b ≠ 0, c = 0)	1	-1	0	x1 = 0, x2 = 1

Етапи розробки програми [4, 5, 15]. Процес розробки програми на мові С++ включає чотири етапи (рис. 4):

1) написання та редагування тексту програми за допомогою текстового редактора, зберігання її у вигляді вихідного файлу з розширенням .cpp.

2) препроцесорна обробка та компіляція програми, які дозволяють отримати об'єктний файл. Вихідний файл обробляється препроцесором і, в разі необхідності, до тексту програми приєднуються інші файли (з розширенням .cpp, .h). Далі компілятор перевіряє модернізований файл на наявність синтаксичних помилок і, якщо помилки не виявлені, переводить текст програми в машинний код, який записується у файл з розширенням .obj.

3) компоновка (зборка, об'єднання) об'єктного файлу програми з другими об'єктними файлами стандартних та спеціальних бібліотек (якщо необхідно) у виконавчий файл з розширенням exe.

4) налагодження (процес виявлення помилок) і тестування програми.

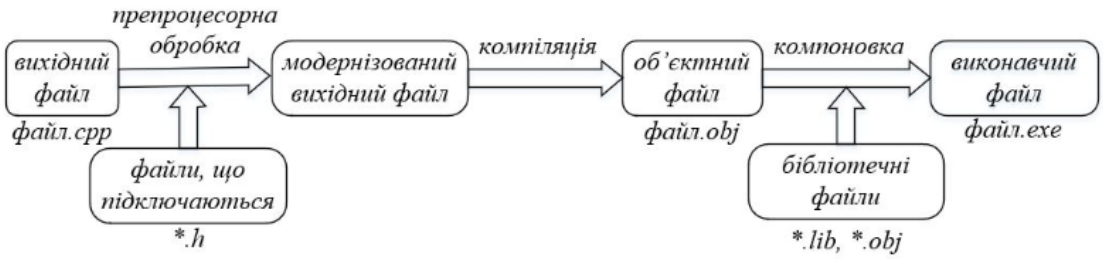


Рис. 4. Схема отримання виконавчого файлу

Середовище розробки програмного забезпечення Microsoft Visual Studio [15]. Розробка програм відбувається за допомогою спеціальних комплексів програм, які називаються системами програмування і дозволяють створювати програми на визначеної реалізації мови програмування. Системи програмування навіть одного виробника мають різні версії, які відображають розвиток технології програмування і еволюцію середовища виконання програм. Деталі процесу будування програм описані в документації системи програмування.

Microsoft Visual Studio – це система програмування від компанії Microsoft для розробки програмного забезпечення:

– під операційні системи Windows, iOS і Android,

– Web- і хмарних додатків.

Microsoft Visual Studio дозволяє розробляти консольні додатки і додатки з графічним інтерфейсом, у тому числі з підтримкою технології Windows Forms.

Microsoft Visual Studio об'єднує групу вікон різного призначення, меню і панелі інструментів в інтегроване середовище розробки або IDE (Integrated Development Environment), яке значно спрощує виконання різних завдань розробки програмного забезпечення.

Microsoft Visual Studio включає в себе

– редактор вихідного коду з підтримкою технології IntelliSense (система автодоповнення),

– компілятор,
– автоматизовані засоби зборки,
– засоби налагоджування програм та інші засоби.

Microsoft Visual C ++ (зазвичай скорочують до Visual C ++ або MSVC) - це назва для бібліотек і засобів розробки на мові асемблера, C ++ і C, що входять до складу Visual Studio в Windows. Ці засоби і бібліотеки дозволяють створювати додатки універсальної платформи Windows (UWP), власні класичні і серверні додатки Windows, кросплатформені бібліотеки і додатки для Windows, Linux, Android і iOS, а також керовані програми та бібліотеки, що використовують платформу .NET Framework. За допомогою Visual C ++ можна розробити що завгодно - від простих консольних додатків до найскладніших додатків для настільних систем Windows, від драйверів пристроїв і компонентів операційної системи до кроссплатформенних ігор для мобільних пристроїв і від найдрібніших пристроїв Інтернету речей до багатосерверних високопродуктивних обчислювальних систем в хмарі Azure.

Середовище програмування Microsoft Visual Studio C++ використовує концепцію робочої області (solution). В одній робочій області може міститися декілька проектів. Проект – це всі файли, необхідні для побудови виконуваного файлу. Незважаючи на наявність в робочій області декількох проектів, працювати можна тільки з одним (активний проект). Для робочої області створюється окрема папка, що включає каталог і кілька файлів, основним з яких є конфігураційний файл області з розширенням .sln. Всередині робочої області кожен проект має власний каталог (якщо проект один, то його каталогом може бути каталог робочої області), в якому знаходиться конфігураційний файл проекту з розширенням .vcproj і вихідні файли проекту з розширеннями .h (заголовні файли) і .cpp (файли реалізації).

Консольний додаток [9, 15]. Консольний додаток – це програма, яка для взаємодії з користувачем використовує консоль – клавіатуру і монітор, що працює в режимі відображення символьної інформації (літери, цифри і спеціальні знаки). Консольний додаток працює у вікні командного рядка (рис. 2), яке часто називають вікном консолі. Консольний додаток не має графічного інтерфейсу. Проект консольного додатку створюється пустім і передбачає додавання в нього вихідних файлів вручну.

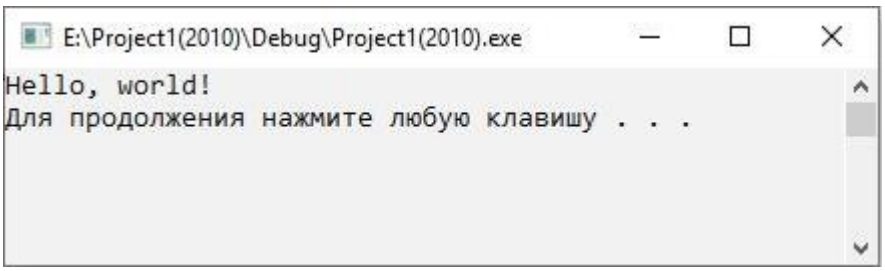


Рис. 5. Консольний додаток працює у вікні командного рядка

Стандартне вікно командного рядка – це вікно з символами білого кольору на чорному тлі. Параметри шрифтів, колір тексту та фону, використання буфера обміну та інші властивості командного рядка можна змінити. Клацніть лівою

кнопкою мишки іконку вікна консолі, у меню виберіть пункт «Властивості» та налаштуйте вікно консолі відповідно до своїх уподобань.

Структура програми мовою С++ [4, 5, 9, 11]. Можна виділити наступні великі частини типового вихідного файлу:

1) підключення заголовних файлів;
2) оголошення символічних констант, допоміжних функцій, класів і типів даних, глобальних змінних;
3) заголовок головної функції (main);
4) визначення головної функції – блок, що містить:
   - оголошення локальних змінних і констант і їх ініціалізацію (присвоєння початкових значень);
   - введення вихідних даних (діалог з користувачем);
   - обробка даних - обчислення виразів, виконання операторів;
   - вивід результатів;
   - повернення коду завершення головної функції.

Подібне структурування тексту вихідного модулю, який розміщується в одному файлі з розширенням .cpp, не єдино можливе, але слідування даного зразка полегшує розуміння і налагодження програм.

Рекомендації щодо оформлення коду програми [7]**.** Загальні принципи, що дозволяють написати синтаксично вірну програму, є такими:

– перш, ніж використовувати функції, змінні, типи даних, слід оголосити їх або підключити файли з їх оголошеннями;

– в будь-якій послідовності дій потрібно прагнути до тріади: ініціалізація (введення), обробка, повернення значення (виведення);

– проміжні змінні оголошувати і ініціалізувати на початку блоків або операторів, в яких вони використовуються.

Намагайтеся писати акуратний код, дотримуйтесь всюди однакової величини відступів – код повинен легко читатися, в ідеалі – бути самодокументуемим, інакше ніхто не стане в ньому розбиратися.

Правила оформлення тексту програми, спрямовані на полегшення розуміння сенсу і підвищення наочності, такі:

1) розділяти логічні частини програми порожніми рядками;
2) розділяти операнди і операції пробілами;
3) для кожної фігурної дужки (або тільки для дужки, яка закривається) відводити окремий рядок;
4) в кожному рядку повинно бути не більш одного оператора;
5) обмежувати довжину рядка 60-70 символами;
6) відступами зліва відображати вкладеність операторів і блоків;
7) довгі оператори розташовувати в декількох рядках;
8) проводити алгоритмізацію таким чином, щоб визначення однієї функції займало, як правило, не більше одного екрану тексту.

Ці правила потрібно використовувати спільно з правилами іменування елементів програми і правил коментування тексту програми.

Завдання

Для завдання A - C за обраним варіантом:

a. проаналізувати індивідуально обрані завдання, визначити вхідні і вихідні дані;

b. підібрати математичні формули для вирішення завдання (якщо необхідно);

c. обрати і описати метод рішення;

d. розробити тести для перевірки рішення завдання;

e. скласти алгоритм рішення завдання (у вигляді блок-схеми та діаграми Нассі-Шнейдермана).

Вимоги до програм (завдання A-C):

- вхідні дані вводяться з клавіатури;
- результати роботи програми виводяться на екран.

Вимоги до тексту програми:

- коментарі щодо призначення програми, її вхідних і вихідних даних;
- коментарі щодо призначення кожного блоку програми;
- самодокументованість коду: всі ідентифікатори повинні мати назви, що відповідають суті змінних.

Завдання А:

Дано дві геометричні фігури, розміри яких дозволяють створити нову фігуру способом, який описаний в завданні. Обчисліть об'єм і площу поверхні отриманої фігури.

1) На верхню грань куба з довжиною ребра c поставили прямокутний паралелепіпед ($a \times b \times H$), поєднавши осі симетрії фігур. Ось симетрії у куба і у прямокутного паралелепіпеда проходить через центри нижньої і верхньої граней. Відповідні бокові грані куба і прямокутного паралелепіпеда паралельні.

2) У куба з довжиною ребра a на двох протилежних бічних гранях вирізали півкулі радіусу r, поєднавши осі симетрії фігур. Ось симетрії у куба проходить через центри двох протилежних бічних граней, в яких вирізані півкулі. Ось симетрії у півкулі збігається з його висотою.

3) У кубі з довжиною ребра a вирізали вертикальний отвір у вигляді циліндра з радіусом R та висотою a, поєднавши осі симетрії фігур. Ось симетрії у куба і у циліндра проходить через центри нижньої і верхньої граней.

4) На циліндр з радіусом R та висотою a поставили конус з радіусом r та висотою H, поєднавши осі симетрії фігур. Ось симетрії у циліндра проходить через центри нижньої і верхньої граней. Ось симетрії у конуса збігається з його висотою.

5) У кубі з довжиною ребра a вирізали вертикальний отвір у вигляді чотирикутної піраміди з довжиною сторін основи b і висотою a.

6) На куб з довжиною ребра a поставили циліндр з радіусом R та висотою H, поєднавши осі симетрії фігур. Ось симетрії у куба і у циліндра проходить через центри нижньої і верхньої граней.

7) На верхньої грані прямокутного паралелепіпеду ($a \times b \times H$) вирізали півкулю радіуса R, поєднавши осі симетрії фігур. Ось симетрії у прямокутного паралелепіпеда проходить через центри нижньої і верхньої граней. Ось симетрії у півкулі збігається з його висотою.

8) У циліндрі з радіусом R та висотою H вирізали конус з радіусом r та висотою H, поєднавши осі симетрії фігур. Ось симетрії у циліндра проходить через центри нижньої і верхньої граней. Ось симетрії у конуса збігається з його висотою.

9) На циліндр з радіусом R та висотою a поставили півкулю з радіусом r, поєднавши осі симетрії фігур. Ось симетрії у циліндра проходить через центри нижньої і верхньої граней. Ось симетрії у півкулі збігається з його висотою.

10) На основі конуса з радіусом R та висотою H розмістили півкулю радіусу r, поєднавши вісь симетрії фігур. У півкулі і у конуса ось симетрії збігається з висотою фігури.

11) У кубі з довжиною ребра a вирізали вертикальний отвір у вигляді конуса з радіусом R та висотою H, поєднавши осі симетрії фігур.

12) На циліндр з радіусом R та висотою a поставили кубі з довжиною ребра a, поєднавши осі симетрії фігур.

13) У циліндрі з радіусом R та висотою H вирізали півкулю з радіусом R, поєднавши осі симетрії фігур. У півкулі і у циліндра ось симетрії збігається з висотою фігури.

14) З конуса з радіусом R та висотою H вирізали півкулю радіусу R поєднавши вісь симетрії фігур. У півкулі і у конуса ось симетрії збігається з висотою фігури.

15) На куб з довжиною ребра a поставили півкулю з діаметром a, поєднавши осі симетрії фігур. Ось симетрії у куба і у півкулі проходить через центри нижньої і верхньої граней.

16) Конус з радіусом R та висотою H поєднали з кубом з довжиною ребра a. Грань куба вписана в основу конуса та ось симетрії у куба і конуса проходить через центри нижньої і верхньої граней.

17) Дві трикутні піраміди з ребром a поєднали за основою.

18) З трикутної піраміди з ребром a вирізали півкулю радіусом r. У півкулі і у піраміди ось симетрії збігається з висотою фігури.

19) З куба з ребром a вирізали трикутну піраміду із ребром c.

20) На трикутну піраміду з ребром a поставили півкулю радіусом r. У півкулі і у піраміди ось симетрії збігається.

21) На циліндр з радіусом R та висотою H поставили півкулю з радіусом R, поєднавши осі симетрії фігур.

Завдання В

1) Пасажир повинен був здати в камеру зберігання порожню валізу в формі паралелепіпеда розмірами a1, a2 і a3 і коробку розмірами b1, b2 і b3. Оплачувати потрібно кожен предмет. Визначити, чи зможе пасажир заощадити на оплаті, помістивши коробку у валізу.

2) Дано три числа: a, b, c. Визначити, чи можуть вони бути сторонами трикутника, і якщо так, то визначити, чи є він тупокутним, гострокутним чи прямокутним.

3) Дано три числа: a, b, c. Визначити, чи можуть вони бути сторонами трикутника, і якщо так, то визначити його тип: рівносторонній, рівнобедрений, різносторонній.

4) Дано натуральне число, яке визначає вік людини в роках. Дати для цього числа найменування «років», «рік», «роки»: наприклад, 1 рік, 2 роки, 5 років.

5) Визначити суму яку необхідно виплатити співробітникові на підприємстві, якщо відомо його оклад та стаж роботи. Якщо стаж співробітника від 3 до 10 років, то він отримує ще й премію у розмірі 50% від окладу, якщо стаж від 10 до 20 років, то премія – 65 % від окладу, якщо стаж перевищує 20 років, то премія становить 90% від окладу.

6) Чи можна на прямокутної ділянці забудови розміром a на b метрів розмістити два будинки розміром в плані p на q і r на s метрів? Будинки можна розташовувати тільки паралельно сторонам ділянки.

7) Дано дійсні додатні числа a, b, c, x, y. З`ясувати, чи пройде цеглина з ребрами a, b, c в прямокутний отвір зі сторонами x, y. Просовувати цеглину в отвір дозволяється тільки так, щоб кожне з її ребер було паралельно або перпендикулярно кожної зі сторін отвору.

8) Визначити суму податку на доходи, яку необхідно сплатити, якщо процент відрахувань залежить від суми доходу. Якщо доход не перевищує 500, то процент становить P1, якщо доход не перевищує 1600, то процент – P2, якщо сума доходу менша 5000, то процент – P3, якщо ж сума доходу перевищує 5000, то процент становить P4. Процентні ставки P1, P2, P3, P4 ввести з клавіатури.

9) Дано дійсні додатні числа a, b, c, d. З`ясувати, чи можна прямокутник зі сторонами a, b вмістити всередині прямокутника зі сторонами c, d так, щоб кожна зі сторін одного прямокутника була паралельна або перпендикулярна кожній стороні другого прямокутника.

10) Два відрізка на осі X задані координатами своїх кінців. Визначити, чи мають ці відрізки спільні точки.

11) Дано сторони трикутника a, b, c. Розрахувати площу трикутника за формулою Герона.

12) Дано координати точки x та y. Перевірити чи належать вони до 1 чверті Декартової системи координат.

13) Дано два числа a та c: Вирішити рівняння типу ax^2+c=0.

14) Дано координати точки x та y. Перевірити чи належать вони до 4 чверті Декартової системи координат.

15) Дано три числа a, b, c. Знайти добуток найбільшого і найменшого із трьох заданих чисел.

16) Створити алгоритм для розрахунку заробітної плати за місяць. Передбачити введення: кількості відпрацьованих днів; заробітної плати за один день та виконати дію - якщо працівник відпрацював 10 і менше 20 днів, обчислити

зарплату за формулою **zp=zpd*day+300** (zpd – заробітна плата за один день, day – кількість відпрацьованих днів);

17) Дано координати точки x та y. Перевірити чи належать вони до 2 чверті Декартової системи координат.

18) Дано два числа a та b: Вирішити рівняння типу ax^2+bx=0.

19) Дано координати точки x та y. Перевірити чи належать вони до 3 чверті Декартової системи координат.

20) Створити алгоритм для розрахунку заробітної плати за місяць. Передбачити введення: кількості відпрацьованих днів; заробітної плати за один день та виконати дію - якщо працівник відпрацював 20 і більше днів, обчислити зарплату за формулою **zp=zpd*day+700** (zpd – заробітна плата за один день, day – кількість відпрацьованих днів);

21) Створити алгоритм для розрахунку заробітної плати за місяць. Передбачити введення: кількості відпрацьованих днів; заробітної плати за один день та виконати дію - якщо працівник відпрацював менше 10 днів, обчислити зарплату за формулою **zp=zpd*day** (zpd – заробітна плата за один день, day – кількість відпрацьованих днів).

Завдання С

Реалізувати задачу трьома способами, використовуючи три виду циклу.

1) Одна штука деякого товару коштує a гр. Надрукувати таблицю вартості 2, 3, ..., N штук цього товару.

2) Надрукувати таблицю відповідності між вагою в фунтах і вагою в кілограмах для значень 1, 2, ..., N фунтів (1 фунт = 453 г).

3) Надрукувати таблицю значень функції $y(x)=ax^2+bx+c$ на проміжку від *x1* до *x2* з заданим кроком *dx*.

4) Надрукувати таблицю перекладу відстаней в дюймах в сантиметри для значень 10, 11, ..., N дюймів (1 дюйм = 25,4 мм).

5) Надрукувати таблицю перекладу 1, 2, ... N доларів США в гривні за поточним курсом.

6) Надрукувати таблицю вартості 50, 100, 150, ..., 1000 г сиру.

7) Обчислити значення функції $y(t)=2t^2+5.5t-2$, $t=x+2$ для значень *x* на проміжку від *a* до b з заданим кроком *dx*.

8) Вважаючи, що Земля – ідеальна сфера з радіусом R=6350 км, визначити відстань до лінії горизонту від точки з висотою над Землею, що дорівнює 1, 2, ..., 10 км.

9) Щільність повітря зменшується з висотою за законом $p=p0e^{-hz}$, де p – щільність на висоті h метрів, $p0$ =1,29 кг/м3, z= 1,25*10-4. Надрукувати таблицю залежності щільності від висоти для значень від 0 до 1000м через кожні 100 м.

10) Надрукувати таблицю вартості 100, 200, 300, ..., 2000 гр цукерок.

11) Надрукувати таблицю розрахунку суми n-перших членів арифметичної прогресії, де n = 1...25, dx - користувач вводить через клавіатуру.

12) Надрукувати таблицю вартості 25, 35, 45, ..., 200 гр срібла, якщо вартість 10 гр. срібла вартує 15 грн..

13) Обчислити значення функції *y(x)=sin (2x), x=x+dx* для значень *x* на проміжку від *a* до b з заданим кроком *dx.*

14) Обчислити значення функції *y(x)=tg(x)+1, x=x+dx* для значень *x* на проміжку від *a* до b з заданим кроком *dx.*

15) Надрукувати таблицю вартості 50, 75, 100, ..., 500 гр цукру, якщо вартість 10 гр. цукру вартує 0,32 грн..

16) Обчислити значення функції *y(x)=sin (2x)+1, x=x+dx* для значень *x* на проміжку від *a* до b з заданим кроком *dx.*

17) Надрукувати таблицю вартості 10, 15, 20 ... 100 гр золота, якщо вартість 10 гр. золота вартує 80 грн...

18) Надрукувати таблицю розрахунку суми n-перших членів герметричної прогресії, де n = 1...25, dx - користувач вводить через клавіатуру.

19) Надрукувати таблицю значення функції *y(x)=lg(x), x=x+dx* для значень *x* на проміжку від *0* до 10 з заданим кроком *dx=1*.

20) Надрукувати таблицю вартості 100, 150, 200, ..., 1500 гр халви у магазині, якщо вартість 100 гр. халви вартує 25 грн..

21) Обчислити значення функції *y(x)=sin (2x)*cos(x), x=x+dx* для значень *x* на проміжку від *a* до b з заданим кроком *dx.*

Контрольні питання

1. Перерахуйте етапи розв’язування задач на комп’ютері.
2. Що таке алгоритм?
3. Які властивості має алгоритм?
4. Які засоби опису алгоритму ви знаєте?
5. Які є графічні методи представлення алгоритмів?
6. Які основні конструкції представлення алгоритмів графічним засобами?
7. Що таке консольний додаток?
8. Опишіть структуру програми мовою C/C++.

Контрольні питання та завдання підвищеної складності

1. У чому полягає процес тестування алгоритму та які його основні етапи?
2. Надайте схему отримання виконавчого файлу.

ЛАБОРАТОРНА РОБОТА № 2. СТАТИЧНІ МАСИВИ. АЛГОРИТМИ ПОШУКУ І СОРТУВАННЯ.

Тема. Статичні масиви. Алгоритми пошуку і сортування.

Мета роботи.

1. Отримати навички роботи з одновимірними масивами в мові C++.
2. Навчитися розробляти для одновимірних масивів алгоритми:
   – обробки елементів;

– сортування.

Зміст роботи:

1) ознайомитися з теоретичними відомостями до лабораторної роботи;

2) проаналізувати завдання;

3) розробити набори тестів для перевірки правильності виконання завдання;

4) скласти алгоритми для розв’язання завдання;

5) розробити програму;

6) налагодити програму за допомогою набору тестів;

7) скласти звіт з лабораторної роботи.

Зміст звіту:

1) титульна сторінка звіту;

2) тема і мета лабораторної роботи;

3) постановка задач (загальне та індивідуальне завдання за номером варіанту, вимоги до програми і тексту програми);

4) зовнішні специфікації;

5) тести для перевірки рішення завдання;

6) для алгоритму сортування покрокове виконання алгоритму на конкретному прикладі;

7) алгоритми рішення завдання у вигляді блок-схеми або схеми Нассі-Шнейдермана;

8) текст програми;

9) результати тестування програми та їх аналіз.

10) аналіз результатів та висновки щодо роботи з одновимірними статичними масивами.

Теоретичні відомості

Масив як тип даних у мові C++ [8, 9, 10, 13]**.** Масив – це складений тип даних, що складається з фіксованого числа елементів одного й того ж типу даних, які зберігаються в послідовно розташованих комірках оперативної пам’яті і мають спільну назву, яку надає програмист.

Характерні риси масивів:

– тип елементів масива повинен бути конкретно описаний;
– масив – це структура з так званим прямим доступом: всі її компоненти можуть вибиратися довільно й однаково доступні;
– для позначення окремого елемента масива необхідно знати його порядковий номер у структурі – індекс елемента масива;
– число елементів масива задається при його описі й надалі не змінюється.

Індекс елементів масива починається з 0 (нуль). Останній індекс дорівнює N - 1, тобто число N елементів масива на 1 більше від індексу останнього елемента масива.

У мові програмування С++ всі елементи масиву займають суміжні комірки пам'яті. Іншими словами, елементи масива в пам'яті розташовані послідовно один за одним. Комірка з найменшою адресою належить до першого елемента масива, а з найбільшою – до останнього.

Кількість індексів визначає розмірність масиву.

1	2	3	4	5	8
a	b	c	d	e	f

Рис. 6. Одновимірний масив

Одновимірні масиви. Оголошення [4, 5, 6, 8, 9, 11]**.** Оголошення масиву повинно описувати три аспекти:

1. Тип значень кожного елементу.
2. Ім’я масиву.
3. Кількість елементів в масиві.

Загальний вигляд конструкції оголошення одновимірного масиву такий:

тип_елементу	ім’я_масиву [	[розмір_маси];

Наприклад, команда int mas[10] оголошує масив з іменем mas, який складається з 10 цілих чисел, float array[5] – масив array з 5 дійсних чисел, char name[15] – масив з 15 символів.

Розмір масиву – це кількість елементів масиву. При оголошенні масиву для позначення його розміру не можна використовувати змінну. Це може бути тільки *константа*, причому ціла. Значення змінної стає відомим тільки в процесі виконання програми. Крім того, змінна може міняти своє значення. Але перед тим, як почнеться виконання програми, компілятор повинен виділити область пам’яті для зберігання всіх змінних (включаючи елементи масиву). Щоб компілятор міг зарезервувати місце для всіх елементів масиву, їх кількість повинна бути відома заздалегіть. Коли при запуску програми змінним починають надавати їх значення, вся область пам’яті вже повинна бути розподілена. Саме тому розмір масиву задається константою.

Примітка. Слід знати, що за допомогою оператора new можна обійти це обмеження. Цей матеріал буде розглянутий в наступних лабораторних роботах.

Ім’я масиву у програмі змінювати не можна – це стала величина, яка містить адресу першого елемента. Тобто, назва масиву – це вказівник на перший елемент.

Звернутися до елементів масиву можна за допомогою його імені і індексу або використовуючі вказівники (буде розглянуто в наступних лабораторних роботах). Наприклад, змінна mas[3] є четвертим елементом масиву mas:

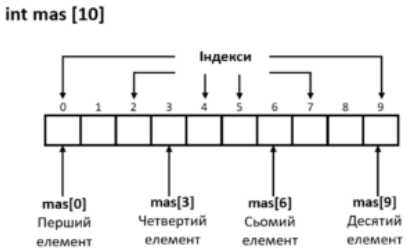


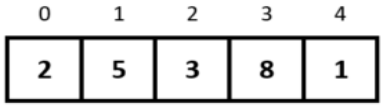
Рис. 7. Масив з десяти цілих елементів

Одновимірні масиви. Ініціалізація [4, 5, 6]**.** Проініціалізувати масив (надати значення елементам масиву) можна одним із способів:

- під час оголошення;
- за допомогою команди присвоєння;
- під час введення даних з клавіатури.

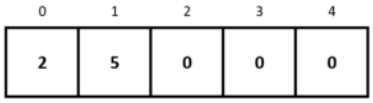
Масив можна ініціалізувати під час його оголошення, записуючи значення змінних через кому у фігурних дужках.

Наприклад, int array[5] = {2, 5, 3, 8, 1}; // масив з 5 цілих елементів:

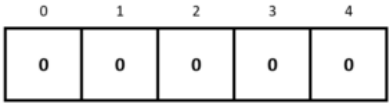


Масив можна ініціалізувати частково, вказавши не всі елементи, тоді компілятор присвоїть іншим елементам нульові значення.

Наприклад, int array[5] = {2, 5}; // масив з 5 цілих елементів:



Наприклад, int array[5] = {0}; встановить значення всіх елементів масиву рівними 0:



Якщо масив повністю ініціалізують під час оголошення, то його розмір зазначати не обов’язково. У цьому випадку компілятор сам визначає, скільки пам’яті необхідно зарезервувати.

Наприклад, int array[] = {2, 5, 3, 8, 1}; приведе до утворення масиву з 5 цілих чисел.

Одновимірні масиви [4, 5, 6]**.. *Приклад використання.*** Створити і заповнити з клавиатури масив з 10 цілих елементів. Вивести на екран парні елементи масиву та їх індекси.

```
#include <iostream>
using namespace std;
int main()
{
  const int SIZE = 10;
```

```
  int array[SIZE];
  // заповнення масиву з клавиатури
  cout << "Enter the values of the array elements :\n";
  for (int i = 0; i < SIZE; ++i) {
   cout << "[ " << i << " ] = ";
   cin >> array[i];
  }
  // виведення парних елементів та їх індексів
  cout << endl << "Paired elements and their indices :\n";
  for (int i = 0; i < SIZE; ++i) {
   if (array[i] % 2 == 0)
     cout << "[ " << i << "] = " << array[i] << endl;
  }
  return 0;
}
```

Алгоритми сортування одновимірних масивів [9, 11, 14]**.** Сортування масиву - це процес розташування всіх елементів масиву в певному порядку. Дуже часто це буває корисним. Наприклад, у вашій поштовій скриньці електронні листи відображаються в залежності від часу отримання; нові листи вважаються більш релевантними, ніж ті, які ви отримали півгодини, годину, два або день назад; коли ви переходите до свого списку контактів, імена зазвичай знаходяться в алфавітному порядку, тому що так легше щось знайти.

Сортування зазвичай виконується шляхом повторного порівняння пар елементів масиву і заміни значень, якщо вони відповідають заданим критеріям. Порядок, в якому ці елементи порівнюються, залежить від того, який алгоритм сортування використовується. Критерії визначають, як буде сортуватися масив (наприклад, в порядку зростання або в порядку убування).

Методи сортування розрізняють в залежності від порядку і правил порівняння елементів масиву. Розглянемо кілька методів сортування масиву:

1. Сортування бульбашкою (Bubble sort)
2. Сортування перемішуванням (Cocktail sort)
3. Сортування вставками (Insertion sort)
4. Сортування гнома (Gnome sort)
5. Сортування вибором (Selection sort)
6. Сортування Шелла (Shell sort)
7. Сортування за розрядами (Radix sort)
8. Сортування підрахунком (Counting sort)
9. Сортування парне-непарне
10. Сортування гребінцем (англ. Comb sort)

Завдання

Розробити алгоритм та програму для виконання індивідуального завдання.
Вимоги до програми:
– вхідні дані вводити з клавіатури;

– вхідні дані перевіряти на коректність (діапазон значень, захист від дурня);

– виконати сортування масиву методом бульбашки та методом вставками;

– результати роботи програми вивести на екран.

Вимоги до тексту програми:

– коментарі щодо призначення програми, її вхідних та вихідних даних;

– коментарі щодо призначення блоків програми;

– самодокументуємий код: назви (ідентифікатори) змінних і функцій мають назви, що відповідають їх суті.

Таблиця 3. Індивідуальні завдання до лабораторної роботи 2

№	Завдання
1.	Є масив даних про температуру тіла хворих на COVID - 19 інфекційного відділення лікарні. Сформувати два нових масиву: нормальної і підвищеної температури.
2.	Є масив даних про результати модульного контролю з дисципліни "Основи програмування" в групі ПЗ20хх. Оцінки виставлені по 100-бальній системі. Знайти середній бал по групі, кількість незаліків і оцінок "задовільно", "добре" і "відмінно" (кожну окремо).
3.	Є масив монет і купюр, що містяться в гаманці. Порахувати суму готівки.
4.	Є масив закладок в книзі. Визначити, чи є закладка на заданій сторінці.
5.	Є масив координат точок на осі X. Знайти координату точки, найближчої до заданої точки.
6.	У масив записали довжини слів тексту. Знайти кількість слів середньої довжини.
7.	Протягом декількох днів літнього періоду в м. Дніпро фіксували полуденну температуру повітря. Підрахувати кількість днів з мінімальною і максимальною температурою повітря (окремо).
8.	У масив записали результати гри в дартс групи учасників. Визначити кількість очок у кожного з трьох призерів.
9.	Задані координати набору точок на площині. Визначити точки, які потрапляють в коло заданого радіуса з центром в заданій точці.
10.	У масив записали розміри файлів папки Windows в кілобайтах. Визначити скільки файлів папки мають розмір, що відрізняється від середнього розміру файлів заданої папки не більше, ніж на 10%.
11.	Через станцію X протягом доби проходить різна кількість пасажирських поїздів. Сформувати масив та визначити середнє число проходження поїздів по станції за годину, годину з максимальним потоком поїздів.
12.	Бібліотеку протягом місяця відвідують читачі. Визначити найбільш та найменш відвідувані дні читачами бібліотеки.
13.	Банкомат містить купюри різного номіналу. Порахувати суму готівки, що може видати банкомат, якщо в банкоматі повинно залишитися не менше, ніж 15% від доступної суми.
14.	У таблицю після прийому школярів 1-го класу записали ріст кожної дитини. Визначити середній ріст та визначити трьох дітей, що мають зріст близький до середнього.

15.	На обчислювальному центрі проводити інвентаризацію та рахували кількість комп'ютерів у кожній залі занесли у таблицю. Визначити середню кількість машин для кімнат обчислювального центру.
16.	Протягом деякої кількості років фермер записував у таблицю свій врожай. Визначити найурожайніший рік у фермера та загальну кількість врожаю, яку зібрав фермер зі своїх полів.
17.	На уроці фізичної культури студенти здавали залік із стрибків у довжину, викладач фіксував їх результати. Вкажіть студента, який показав найкращий результат, та трьох його одногрупників, що показали трошки гірші результати.
18.	В масив записали розміри графічних файлів для використання у контенті на сайті. Визначити середній розмір файлу, що може бути використаний на сайті та сформувати новий масив масив файлів, розмір яких перевищує розмір середнього на 65%.
19.	Садівник у кінці осені підводив підсумки врожаю, що він зібрав з кожної яблуні протягом літа-осені поточного року. Сформувати два масиви з деревами, із врожаєм більше та менше середніх значень врожаю із дерев.
20.	Користувач інтернету протягом доби відвідує сторінки. Вказати годину з найбільшою та найменшою динамікою відвідування користувачем сайтів.

Контрольні питання

1. Що таке масив?
2. Чим характеризується одновимірний масив?
3. Як оголошується масив?
4. Що таке індексація елементів масиву?
5. Що таке лінійний пошук?
6. Розкажіть етапи методу сортування масиву бульбашкою.
7. Розкажіть етапи методу сортування масиву вставками.

Контрольні питання та завдання підвищеної складності

1. Які методи сортування масивів ви знаєте?
2. В чому суть методу Шелла?
3. В чому полягає метод швидкого сортування?

ЛАБОРАТОРНА РОБОТА № 3. ФУНКЦІЇ. МЕТОДИ ПЕРЕДАЧІ ПАРАМЕТРІВ. ДИНАМІЧНІ МАСИВИ.

Тема. Функції. Методи передачі параметрів. Динамічні масиви.

Мета. Отримати навички роботи:

1. із динамічними масивами.
2. із функціями. Навчитися передавати параметри у функцію різними методами.

Зміст роботи:

1) ознайомитися з теоретичними відомостями до лабораторної роботи;
2) проаналізувати завдання А та розробити концепцію виконання завдання В;
3) розробити набори тестів для перевірки правильності виконання завдання;

4) скласти алгоритми для розв’язання завдання;
5) розробити програм для двох індивідуальних завдань;
6) налагодити програми за допомогою набору тестів;
7) скласти звіт з лабораторної роботи.

Зміст звіту:

1) титульна сторінка звіту;
2) тема і мета лабораторної роботи;
3) для завдання А.
   3.А.1) постановка задач (загальне та індивідуальне завдання за номером варіанту, вимоги до програми і тексту програми);
   3.А.2) зовнішні специфікації;
   3.А.3) тести для перевірки рішення завдання;
   3.А.4) алгоритми рішення завдання у вигляді блок-схеми або схеми Нассі-Шнейдермана;
   3.А.5) текст програми;
   3.А.6) результати тестування програми та їх аналіз.
4) для завдання В.
   4.В.1.) постановка завдання;
   4.В.2) текст програми;
   4.В.3) результати виконання функцій і їх аналіз (повний опис механізму передачі параметрів і повертання значення з функції);
5) аналіз результатів та висновки щодо зручності та ефективності застосованих методів та використаних інструментів.

Теоретичні відомості

Динамічна пам'ять [8, 9, 11]**.** Динамічна пам'ять, яка називається також "купою", виділяється явно за запитом програми з ресурсів операційної системи і контролюється вказівником. Вона не ініціалізується автоматично і повинна бути явно звільнена. На відміну від статичної і автоматичної пам'яті динамічна пам'ять практично не обмежена (обмежена лише розміром оперативної пам'яті) і може динамічно змінюватися в процесі роботи програми.

Динамичні масиви [4, 5, 6, 8, 9, 11]**.** Динамічна пам’ять (купа) - це вільна пам’ять, у якій під час виконання програми можна виділяти місце залежно від потреб користувача. Доступ до виділених ділянок динамічної пам’яті, що називаються динамічними змінними, здійснюється тільки через вказівники. Час існування динамічних змінних - від початку створення до кінця програми або до явного звільнення пам’яті.

У мові С++ застосовують два способи роботи з динамічною пам’яттю. Перший з них дістався в спадщину від мови С і використовує сукупність функцій malloc(), другий - працює з операціями new та delete.

Оператор new виділяє пам’ять і повертає її адресу. За допомогою оператора delete відбувається звільнення пам’яті, на яку вказує змінна-вказівник.

Загальна форма запису оператора new:

змінна-вказівник = new тип_змінної;.

Оператор delete має вигляд:

delete [ ] змінна-вказівник; .

Динамічні масиви створюють за допомогою операції new, при цьому необхідно вказати їх тип і розмірність.

Наприклад, для одновимірного масиву дійсних чисел, що має 100 елементів, треба записати:

```
int n = 100;
float *p = new float[n];    //змінна-вказівник на float отримує адресу ділянки
```

у динамічній пам'яті для розміщення 100 елементів дійсного типу.

Слід пам'ятати, що динамічні масиви при створенні не можна ні ініціювати, ні обнуляти.

Матриці. Спосіб "вказівник на вказівник" [6, 8, 9]**.** Розглянемо перший спосіб виділення динамічної пам'яті під двовимірний масив, коли обидві його розмірності задаються на етапі виконання програми.

Наприклад, розподіл динамічної пам'яті для матриці, що має n рядків і m стовпців та елементи цілого типу, можна здійснити так:

```
int n, m;
cout << " Введите количество строк и столбцов: “;
cin >> n >> m;
//оголошення змінної тип «вказівник на вказівник на int» і виділення
// пам'яті для масиву вказівників на рядки матриці
int **a = new int *[n];
for (int i = 0; i < n; i++)
    a[i] = new int [m];   //кожному елементу масиву вказівників на рядки
```

присвоюється адреса початку ділянки пам'яті, виділеної для рядка матриці

Наочно це представлено рис.8.

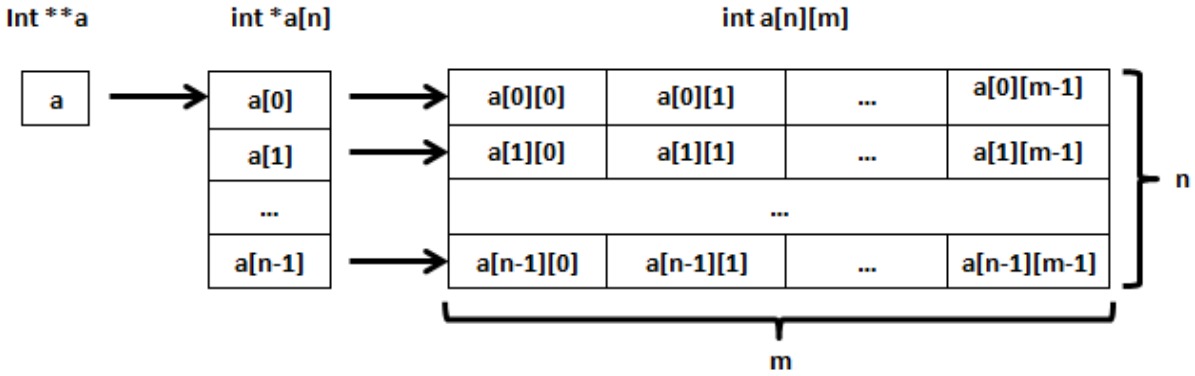


Рис. 8. Виділення пам'яті для двовимірного масиву

Звільнення пам'яті з-під масиву будь-якої кількості вимірів виконується за допомогою операції delete[].

Наприклад, звільнення динамічної пам'яті для матриці, що має n рядків і m стовпців, можна здійснити так:

```
for (int i = 0; i < n; i++)
  delete[] a[i];   //звільняється ділянки пам'яті, виділена для і-го рядка
delete[] a;        //звільняється ділянки пам'яті, виділена для масиву
```

// вказівників на рядки матриці

Завдання

Завдання А. (Динамічний масив)

Розробити програму для виконання індивідуального завдання.

Вимоги до програми:

1) обробка некоректного введення користувачем розмірності матриці;
2) виділення і звільнення пам'яті для матриць виконується операторами *new* і *delete*;
3) заповнення обох матриц паралельно випадковими значеннями відповідно до умов завдання (в результаті отримуються матриці, елементи яких однакові);
4) використання для кожної частини завдання динамічної матриці такої структури, яка є більш ефективною для її виконання (економія часу, пам'яті, короткий код);
5) виведення результатів роботи програми на екран;
6) реалізація частин завдання у вигляді функцій.

Індивідуальні завдання.

1. Є фруктовий сад. Восени з кожного дерева зібрали врожай і результати записали в таблицю відповідно до розташування дерев в саду.
   1) знайти, де росте дерево, з якого зібрали найбільший врожай;
   2) відсортувати рядки таблиці по першому елементу по зростанню.
2. В м. Дніпро протягом серпня кожен час доби заміряли температуру повітря. Результати заносили в таблицю, рядки якої відповідають дням місяця.
   1) визначити день і час, коли в останній раз була зафіксована найнижча температура повітря;
   2) вилучити з таблиці всі рядки з найнижчою температурою повітря.
3. В таблицю занесли результати модульного контролю першого курсу ДНУЗТ, набраного за спеціальністю 121. Для кожного студента в таблиці завели окремий рядок.
   1) визначити скільки студентів отримали за модуль оцінку «відмінно»;
   2) відсортувати рядки таблиці по результатам першого модуля по спадаючій.
4. Група молодиків грала в Дартс. Кожен з учасників зробив серію кидків дротика. Результати кидків кожного учасника занесли в таблицю.
   1) визначити скільки раз за змагання дротик влучив «в молоко»;
   2) вилучити з таблиці результати учасників, які влучили «в молоко» більше трьох разів.
5. В м. Дніпро протягом вересня кожен час доби заміряли швидкість повітря. Результати заносили в таблицю, рядки якої відповідають дням місяця.
   1) визначити день і час, коли в перший раз була зафіксована найбільша швидкість повітря;
   2) відсортувати рядки таблиці за полуденною швидкістю повітря по зростанню.

6. Провели серії підкидання кількох монет у повітря. Результати випробувань (1 – «орел», 0 – «решка») по кожної монеті занесли в таблицю.

1) визначити номер першої монети з найбільшою кількістю «орлів»;
2) вилучити з таблиці рядки, які починаються з трьох «решок».

7. Протягом семестру фіксували присутність на заняттях студентів першого курсу ДНУЗТ, набраного за спеціальністю 121. Результати моніторингу занесли в таблицю у вигляді суми пропусків занять за тиждень для кожного студента.

1) визначити номер останнього студента, кількість пропусків занять якого є найменшою;
2) відсортувати рядки таблиці за результатами останнього тижня по спадаючій.

8. Є таблиця чемпіонату з футболу. Результати матчів – перемога, поразка або нічия для кожної команди записані в окремий рядок.

1) визначити номер першої команди, яка не мала поразок у чемпіонаті;
2) вилучити з таблиці команди, які мають більше двох поразок.

9. Протягом семестру викладач в журнал заносив оцінки, які студенти першого курсу ДНУЗТ, набраного за спеціальністю 121, отримували за лабораторні роботи з курсу «Основи програмування».

1) визначити кількість студентів, які здали всі лабораторні роботи;
2) відсортувати рядки таблиці за середнім балом по зростанню.

10. Є фруктовий сад, посаджений по принципу шахової дошки: по контуру висаджені яблуні, в середині чергуються – сливи і груші. Восени з кожного дерева зібрали врожай і результати записали в таблицю відповідно до розташування дерев в саду.

1) визначити кількість дерев в саду, що засохли;
2) вилучити з таблиці рядки таблиці, в яких є записи про засохлі дерева.

11. Протягом грудня проводилось вимірювання вологості повітря.

1) визначити кількість днів коли вологість повітря дорівнювалась 45% та визначити останній день у грудні, коли вологість повітря була 45%;
2) відсортувати рядки таблиці за середнім значення вологості повітря по спадаючій.

12. Поле розбите на квадратні ділянки, на яких висаджено різні сорти помідорів. Кількість врожаю з кожного ділянки було записано у таблицю.

1) на якому рядку було зібрано найбільша кількість помідорів;
2) вилучити з таблиці рядки з найменшою кількістю помідорів.

13. У м. Дніпро кожен місяць протягом року збираються статистичні дані щодо до кількісті хворих людей у місті. Можлива однакова кількість хворих у різні місяці. Дані записали у таблицю.

1) знайти місяць з найбільшою кількістю хворих
2) відсортувати дані за зростанням хворих у першому тижні.

14. Кожен місяць протягом року було підраховано кількість пасажирів, які виходили на залізничні станції Дніпро-Головний.

1) визначити відношення найбільшої кількісті пасажирів до найменшої кількість пасажирів, що виходили на залізничні станції Дніпро-Головний;

2) відсортувати дані за спаданням пасажирів

15. На стоянці знаходяться автомобілі різних марок Ford, BMW, Kia. Вони розташовані у кілько рядів.

1) визначити кількість автомобілів марки Ford та їх розташування;
2) вилучити із таблиці рядки, в яких знаходяться біль ніж два автомобіля марки Ford.

16. ДІІТ зробив набір студентів на першій курс за різніми спеціальностями. На кожну спеціальність було зараховано різна кількість студентів.

1) визначити спеціальність, на яку кількість зарахованих студентів дорівнює числу К;
2) відсортувати дані за зростанням кількісті студентів

17. В університеті є статистика, щодо кількості студентів третього курсу з різних факультетів. З кожного факультету було відрахована різна кількість студентів.

1) знайти факультет на якому відсоток відрахованих студентів був найменшим;
2) побудувати таблицю даних, яка містить дані: факультет та кількість студентів на факультеті після відрахування. Відсортувати ці дані за зростанням

18. У дитячому садочку існує декілька груп малюків. В кожній групі виховуються хлопчики та дівчата

1) визначити групу з найбільшим відсотком дівчат;
2) вилучити з таблиці рядки, які містять кількість дівчат, яка дорівнює числу К.

19. В олімпійських іграх приймають участь команди з різних держав. Кожна держава виставляє різну кількість спортсменів.

1) визначити команду з найменшою та найбільшою кількістю спортсменів;
2) відсортувати рядки таблиці за кількістю спортсменів по зростанню

20. У кондитерському відділу магазина виставлено на продаж цукерки різних сортів. Кожен сорт цукерок користується різним попитом.

1) визначити два сорти цукерок, які користуються найбільшим попитом
2) відсортувати рядки таблиці за спаданням попиту на цукерки.

Завдання В.

Розробити програму для демонстрації механізму передачі параметрів згідно індивідуального завдання.

Вимоги до програми:

1) введення вхідих даних з клавіатури

2) в програмі розробити три варіанти заданої в індивідуальному завданні функції; всі параметри і значення, що повертає функція, передаються:

1-й варіант – за значенням;
2-й варіант – за посиланням;
3-й варіант – за вказівником.

3) для кожного варіанту функції передбачити виведення на консоль:

– назви способу передачі параметрів;
– значень параметрів, що передаються у функцію, до і після її виклику;
– значення, яке повертає функція.

Примітка. Результати виконання де-яких функцій можуть не відповідати очікуваним через використаний спосіб передачі параметрів.

Вимоги до тексту програми:

– коментарі щодо призначення програми, її вхідних і вихідних даних;
– коментарі щодо призначення функцій програми, їх вхідних і вихідних даних;
– коментарі щодо призначення блоків програми, дій окремих операторів для пояснення алгоритму;
– самодокументованість коду: ідентифікатори повинні мати назви, що відповідають суті змінних.

Індивідуальні завдання

1) Є діапазон цілих чисел [a, b]. Знайти суму всіх парних чисел цього діапазону та підрахувати їх кількість.
2) Є діапазон цілих чисел [a, b]. Знайти кількість всіх кратних 10 чисел цього діапазону та підрахувати суму їх квадратів.
3) Є діапазон цілих чисел [a, b]. Знайти кількість всіх кратних 5 чисел цього діапазону та підрахувати їх відсоток серед всього діапазону.
4) Є діапазон цілих чисел [a, b]. Знайти суму всіх кратних 3 чисел цього діапазону та підрахувати їх середнє арифметичне.
5) Є діапазон цілих чисел [a, b]. Знайти суму всіх не парних чисел цього діапазону та підрахувати їх відсоток серед всього діапазону.
6) Є діапазон цілих чисел [a, b]. Знайти суму всіх не парних чисел цього діапазону та підрахувати їх середнє арифметичне.
7) Є діапазон цілих чисел [a, b]. Знайти суму всіх парних чисел цього діапазону та підрахувати їх середнє арифметичне.
8) Є діапазон цілих чисел [a, b]. Знайти суму всіх кратних 7 чисел цього діапазону та підрахувати їх кількість.
9) Є діапазон цілих чисел [a, b]. Знайти кількість всіх кратних 3 чисел цього діапазону та підрахувати їх добуток.
10) Є діапазон цілих чисел [a, b]. Знайти суму всіх кратних 5 чисел цього діапазону та підрахувати їх кількість.
11) Є діапазон цілих чисел [a, b]. Знайти добуток елементів кратних 11 з цього діапазону та підрахувати кількість додатніх елементів кратних 11.
12) Є діапазон цілих чисел [a, b]. Знайти суму елементі з цього діапазону, які кратні 5 та менше ніж 100 та підрахувати їх кількість.
13) Є діапазон цілих чисел [a, b]. Знайти добуток від'ємних елементів цього діапазону, які кратні 13 та підрахувати їх суму.
14) Є діапазон цілих чисел [a, b]. Знайти суму елементів цього діапазону, які знаходяться на парних місцях та підрахувати їх кількість.
15) Є діапазон цілих чисел [a, b]. Знайти суму елементів цього діапазону, які знаходяться на непарних місцях та підрахувати їх добуток.

16) Є діапазон цілих чисел [a, b]. Знайти добуток елементів цього діапазону, які знаходяться на парних місцях та підрахувати їх суму

17) Є діапазон цілих чисел [a, b]. Знайти кількість елементів цього діапазону, які мають індекс (місцерозташування) кратний 3 та підрахувати їх кількість.

18) Є діапазон цілих чисел [a, b]. Знайти суму елементів цього діапазону, які кратні 3 та менше числа введеного числа із цього цього діапазону та підрахувати їх кількість.

19) Є діапазон цілих чисел [a, b]. Знайти суму непарних додатних елементів цього діапазону та підрахувати їх кількість.

20) Є діапазон цілих чисел [a, b]. Знайти суму непарних від'ємних елементів цього діапазону та підрахувати їх добуток.

Контрольні питання

1. Що таке динамічний масив?
2. Які функції використовують при роботі з динамічною пам'яттю?
3. Розкажіть схему виділення динамічної пам'яті у динамічній матриці.
4. Що таке функція?
5. Розкажіть про метод передачі параметрів – за значенням.
6. Розкажіть про метод передачі параметрів – за посиланням.
7. Розкажіть про метод передачі параметрів – за вказівником.

Контрольні питання та завдання підвищеної складності

1. У чому полягають переваги роботи з динамічною пам'яттю?
2. У чому полягають недоліки роботи з динамічною пам'яттю?

ЛАБОРАТОРНА РОБОТА №4. СТРУКТУРНИЙ ТИП ДАНИХ В МОВІ С++

Тема. Структурний тип даних в мові С++

Мета.

1) засвоїти поняття складеного типу даних.
2) отримати практичні навички роботи з інтегрованими типами даних – структурами і масивами структур – мови С++.
3) навчитися створювати багатофайлові проєкти.

Зміст роботи:

1) ознайомитися з теоретичними відомостями до лабораторної роботи;
2) проаналізувати індивідуальне завдання;
3) розробити набори тестів для перевірки правильності виконання завдання;
4) скласти алгоритми для розв'язання завдання;
5) розробити програм у вигляді багатофайлового проєкту;
6) налагодити програму за допомогою набору тестів;
7) скласти звіт з лабораторної роботи.

Зміст звіту:

1) вид та номер роботи, дисципліна, П.І.Б. та група виконавця;
2) тема і мета лабораторної роботи;
3) постановка завдання (завдання до лабораторної роботи, індивідуальне завдання за обраним варіантом, вимоги до програми);
4) зовнішні специфікації програми;
5) метод рішення завдання;
6) алгоритм розв'язання задачі у вигляді діаграм Н-Ш (алгоритм для реалізації завдання в цілому (меню користувача, укрупнені блоки для всіх функцій програми), окремі алгоритми виконання задач обробки масиву структур);
7) набір тестів для перевірки роботи меню користувача;
8) набори тестів для перевірки виконання задач обробки масиву структур;
9) текст програми;
10) результати тестування програми та їх аналіз;
    аналіз результатів та висновки щодо розробки програм з використанням структур і масивів.

Теоретичні відомості

Структурний тип даних [4, 5, 6, 14]**.** Структура [14] – це об'єднання однієї або більше змінних в одній області пам'яті, яка має одне ім'я.

Змінні, що входять в структуру називаються полями структури. Такі змінні можуть мати різні типи. Імена полів (змінних) в одному шаблоні мають бути унікальними. Однак, в різних шаблонах імена можуть повторюватись.

Синтаксис оголошення структури:

```
struct им'я структури
{
   тип ім'я_поля структури1;
   тип ім'я_поля структури2;
   ….
};
```

Одне єдине ім'я структури може об'єднувати різні змінні, вони можуть відрізнятися навіть типами даних, це можуть бути як масиви, рядки так і звичайні змінні.

Наприклад:

```
struct inflatable
{
     char name[20];
     float volume;
     double price;
};
```

Оголошення структурної змінної та ініціалізація структур [14]**.** Структурну змінну можна оголосити такими способами.

Спосіб 1. Оголошення змінної на основі раніше створеного шаблону

```
struct Worker
{
  int code;  // код працівника
  char name[50]; // ім'я працівника
  int hours; // кількість відроблених годин
  float cost; // розцінка за 1 годину
};
Worker w;
w.code = 334;
strcpy(w.name, "Johnson B.");
w.hours = 23;
w.cost = 12.85f;
```

Спосіб 2. Оголошення змінної w1 при оголошенні шаблону

```
struct Worker
{
  int code;  // код працівника
  char name[50]; // ім'я працівника
  int hours; // кількість відроблених годин
  float cost; // розцінка за 1 годину
} w1;
```

Доступ до полів змінної w1 з іншого програмного коду

```
// заповнення значеннями полів змінної w1
w1.code = 123;
strcpy(w1.name, "Williams D.");
w1.hours = 33;
w1.cost = 10.58f;
```

Спосіб 3. Оголошення змінної на основі типу, створеного з допомогою кваліфікатора typedef.

```
typedef struct Wrk
{
  int code;
  char name[50];
  int hours;
  float cost;
} WORKER;
```

Тоді, щоб використати змінну з іменем w2 типу WORKER, можна написати такий код:

```
// оголошення змінної на основі типу,
// заданого з допомогою ключового слова typedef
WORKER w2;
w2.code = 100;
strcpy(w2.name, "Typedef");
```

```
w2.hours = 33;
w2.cost = 21.33f;
```

У цьому способі задавати ім’я шаблону структури Wrk не є обов’язковим. Тобто, наступний запис також є коректний

```
// після слова struct назва шаблону відсутня
typedef struct
{
 int code;
 char name[50];
 int hours;
 float cost;
} WORKER;
```

Масив структур та його ініціалізація **[**14**].**

Оголошення масива структур.

```
struct inflatable
{
char name[20 ];
float volume;
double price;
};
```

inflatable gifts[100];

Це оголошення робить gifts масивом структур inflatable.

В результаті кожен елемент масиву, такий як gifts[0] або gifts[99], є об'єктом типуin fla t a b le і може бути використаний з операцією членства:

```
cin >> gifts[0].volume;
cout << gifts [99] . price << endl;
```

gifts є масивом, а не структурою, тому конструкція така як gifts.price є некоректною.

Для ініціалізації масиву структур комбінується правило ініціалізації масивів

Оскільки кожен елемент масиву є структурою, його значення є ініціалізацією структури.

```
inflatable guests [2] =
{
{"Bambi", 0.5, 21.99},
{"Godzilla", 2000, 565.99}
};
inflatable guests [20] =
{
{"Bambi", 0.5, 21.99},
{"Godzilla", 2000, 565.99}};
```

Завдання

Розробіть програму роботи з даними для обраної предметної області.

Вимоги до програми:

– три перши елементи масиву необхідно ініціалізувати при оголошенні масиву;

– вхідні дані вводяться з клавіатури;

– перевірка вхідних даних на відповідність діапазону значень і некоректні символи;

– введені дані зберігаються в масиві структур (масив складається з 20 елементів);

– символьні поля структури оголошуються як масив символів;

– інформація, яка зберігається у символьному полі структури, може складатися з декількох слів;

– керування виконанням програми здійснюється на основі текстового меню користувача;

– дії програми реалізувати у вигляді функцій;

– програму розробити як багатофайловий проект (в першому файлі заголовків розмістити опис структури елементів масиву, у другому - прототипи функцій програми, в файлі реалізації - опис функцій програми, функцію main() оформити окремим файлом);

– результати роботи програми виводяться на консоль (відсортований масив структур).

Вимоги до тексту програми:

– коментарі щодо призначення програми, її вхідних і вихідних даних;

– коментарі щодо призначення функцій програми, їх вхідних і вихідних даних;

– коментарі щодо призначення блоків програми, дій окремих операторів для пояснення алгоритму;

– самодокументованість коду: ідентифікатори повинні мати назви, що відповідають суті змінних.

Індивідуальні завдання.

Зверніть увагу:

1) в завданнях виводу списку виводиться **повна** інформація про елементи списку;

2) в завданнях пошуку елемента масиву виводиться **повна** інформація тільки про **перший** елемент, що задовольняє зазначеним умовам пошуку.

Таблиця 4. Індивідуальні завдання до лабораторної роботи 4.

№	Предметна область	Дані	Дії
1	«День народження»	• прізвище • ім’я	• додавання особи в довідник • виведення списку осіб • виведення списку осіб з вказаним роком народження

		• дата народження (структура з трьох полів – день, місяць, рік)	• пошук наймолодшої особи за роком народження
2	Співробітники	• прізвище та ініціали • назва компанії • виплати в гривнях (структура з двох полів – оклад і премія)	• додавання співробітника в список • виведення списку співробітників • виведення списку співробітників з вказаним окладом • пошук співробітника з найвищою сумою виплат
3	Авіарейси (1)	• номер рейсу • тип літака • час польоту (структура з двох полів – години, хвилини)	• додавання рейсів в список • виведення списку рейсів • виведення списку рейсів з вказаним типом літака • пошук рейсу з мінімальним часом польоту
4	IT компанії	• назва компанії • діяльність (структура з двох полів – основні напрямки діяльності) • кількість співробітників	• додавання компанії в список • виведення списку компаній • виведення списку компаній з вказаною сферою діяльності • пошук компанії з максимальною кількістю співробітників
5	Студенти	• прізвище та ініціали • повний номер групи • успішність (структура з трьох полів – оцінки за останній модуль)	• додавання студента в список • виведення списку студентів • виведення списку студентів вказаної групи • пошук студента з максимальною середньою оцінкою
6	Деталі комп’ютера	• тип деталі • виробник • строк гарантійного обслуговування (структура з двох полів – роки і місяці)	• додавання деталей в список • виведення списку деталей • виведення списку деталей з вказаною ціною • пошук деталі з мінімальним строком гарантійного обслуговування
7	Авіарейси (2)	• номер рейсу • маршрут (структура з двох полів - пункти вильоту і призначення) • ціна квітка у гривнях	• додавання рейсів в список • виведення списку рейсів • виведення списку рейсів з вказаним пунктом призначення • пошук рейсу з найвищою ціною квітка
8	Держава	• назва • столиця • характеристики (структура з двох полів - чисельність населення і площа)	• додавання держави в список • виведення списку держав • виведення списку держав зі столицею, яка починається на задану літеру • пошук держави з найменшою територією
9	«Музична шкатулка»	• назва альбому • рік випуску альбому • тривалість звучання (структура з двох полів – хвилини і секунди)	• додавання альбому в список • виведення списку альбомів • виведення списку альбомів з вказаним роком випуску • пошук альбому з найдовшою тривалістю звучання
10	Програміст	• прізвище та ініціали • стаж роботи в роках	• додавання програміста в список • виведення списку програмістів

		• знання (структура з двох полів – мова програмування і БД)	• виведення списку програмістів з вказаною мовою програмування • пошук програміста з найбільшим стажем роботи
11	Бібліотека	• назва книги, автор • кількість сторінок • видання (структура з двох полів – рік видання та назва видавництва)	• додавання книги з автором у список • виведення списку книг з автором • виведення списку книг з вказаним роком видання • пошук книги з найменшій кількістю сторінок
12	Склад	• назва товару •ціна товару •рік випуску товару (структура з двох полів –рік та місяць випуску)	• додавання товару в список • виведення списку товару • виведення списку товару з вказаним роком випуску • пошук товару з найвищою вартістю
13	Аптека	• назва ліків • держава виготовлення ліків • термін придатності (структура з двох полів –рік та місяць закінчення терміну придатності))	• додавання ліків у список • виведення списку ліків • виведення списку ліків за державою виготовлення • пошук ліків за мінімальним терміном придатності.
14	Каса продажу залізничних квитків	• номер поїзду • маршрут (структура з двох полів - пункти відправлення і призначення) • ціна квітка у гривнях	• додавання поїздів у список • виведення списку поїздів • виведення списку поїздів з вказаним пунктом відправлення • пошук рейсу з найменшою ціною квітка
15	Радіо деталі	• тип деталі • виробник • строк гарантійного обслуговування (структура з двох полів – роки і місяці)	• додавання деталей в список • виведення списку деталей • виведення списку деталей з вказаною ціною • пошук деталі з мінімальним строком гарантійного обслуговування
16	Фірма	• назва фірми • діяльність (структура з двох полів – основні напрямки діяльності) • кількість співробітників	• додавання фірми в список • виведення списку фірм • виведення списку фірм з вказаною сферою діяльності • пошук фірми з мінімальною кількістю співробітників
17	Дитячий садочок	• прізвище та ініціали дитини • номер групи • інформація (структура з трьох полів – рік народження, вага дитини, зріст дитини)	• додавання дитини в список • виведення списку дітей • виведення списку дітей з вказаної групи • пошук найстаршої дитини
18	Довідка "Міста жителях"	• прізвище • ім’я	• додавання особи в довідник • виведення списку осіб

		• дата народження (структура з трьох полів – день, місяць, рік)	• виведення списку осіб з вказаним роком народження • пошук наймолодшої особи за роком народження
19	Автостанція	• номер автобусу • тип автобуса • час поїздки (структура з двох полів – години, хвилини)	• додавання автобусу в список • виведення списку автобусу • виведення списку автобусу з вказаним номером автобусу • пошук рейсу з максимальним часом поїздки
20	Кафедра	• прізвище та ініціали • назва кафедри • виплати в гривнях (структура з двох полів – оклад і премія)	• додавання співробітника в список • виведення списку співробітників • виведення списку співробітників з вказаним окладом • пошук співробітника з найвищою сумою виплат

Контрольні питання

1. Що таке структурний тип даних?
2. Що таке поле структури?
3. Як ініціалізувати структурну змінну?
4. Як звернутися до поля структури?

Контрольні питання та завдання підвищеної складності

1. Що таке масив структур?
2. Як звернутися до елемента поля структури?
3. Яке звернутися до поля структури, якщо структурна змінна оголошена як вказівник на структуру?
4. Опишіть процес сортування масиву структур.

ЛАБОРАТОРНА РОБОТА № 5. РОБОТА З ФАЙЛАМИ.

Тема. Робота з файлами.

Мета:

1) сформувати навички роботи з символьними рядками і функціями обробки символьних рядків засобами стандартної бібліотеки;
2) вивчити засоби роботи з файлами в стилі C та стилі C++;
3) засвоїти навички роботи з файлами;
4) навчитися використовувати аргументи командного рядка.

Зміст роботи:

1) ознайомитися з теоретичними відомостями про роботу із текстовими та бінарними файлами, та функціями обробки рядків;
2) проаналізувати завдання до роботи;
3) розробити набори тестів для перевірки правильності виконання завдання;
4) скласти алгоритми для розв’язання завдання;
5) розробити програм для двох індивідуальних завдань;
6) налагодити програми за допомогою набору тестів;

7) скласти звіт з лабораторної роботи.

Зміст звіту:

1) титульний аркуш (вид та номер роботи, дисципліна, П.І.Б. та група виконавця);
2) тема і мета лабораторної роботи;

Завдання А.

А.1) постановка завдання;

А.2) алгоритм обробки рядка у вигляді діаграми Нассі-Шнейдермана;

А.3) набори тестів;

А.4) текст програми;

А.5) результати тестування програми та їх аналіз;

А.6) висновки щодо обробки символьних рядків, роботи з текстовими файлами і командним рядком .

Завдання В.

В.1) постановка завдання;

В.2) зовнішні специфікації програми;

В.3) алгоритм для розв’язання завдання;

В.3) текст програми;

В.4) результати виконання програми;

В.5) скріншот змісту вихідного файлу у hex-редакторі та його аналіз;

В.6) висновки щодо роботи з бінарними файлами.

Теоретичні відомості

Поняття рядка [4, 5, 6, 14]**.** Масив символів або рядок – один з різновидів одновимірних масивів. Рядок у мовах програмування – це скінчена послідовність символів, які сприймаються і обробляються як єдине ціле. Алгоритми обробки рядків застосовуються для вирішення широкого кола задач: редагування та перекладу текстів, алгебраїчних перетворень формул, криптоаналізу, в інформаційно-пошукових системах, електронних словниках, браузерах тощо. Рядки, як різновиди даних, призначені для введення, обробки і виведення символьної інформації.

Кожен рядок характеризується загальною довжиною, яка визначається при оголошенні, і поточною довжиною – кількістю символів у конкретний момент виконання програми. Довжина рядка може змінюватися в процесі роботи програми, але не може перевищувати загальної довжини. У C/C++ довжина рядка обмежується лише кількістю пам’яті у комп'ютері.

У C/C++ не визначено вбудованого рядкового типу даних. Рядки в C/C++ представлені у вигляді масиву символів, але масив символів – не завжди рядок. Рядком в C/C++ (C-рядком) вважається символьний масив, який завершується символом кінця рядка – нуль-символом. Нуль-символ – це символ з кодом 0, який записується у вигляді керуючої послідовності '\0'. За його розташуванням визначається фактична довжина рядка.

При розміщенні у пам’яті C-рядок займає зв’язну область і зберігається як безперервна послідовність символів. Для C-рядка резервується ділянка пам’яті

на 1 байт більша за загальну довжину рядка (для зберігання символу кінця рядка).

Опис С-рядків [4, 5, 6, 14]**.** Пам'ять під розміщення рядків, як і для будь-яких масивів, може виділятися як компілятором, так і при виконуванні програми (динамічна пам'ять). Оголошений рядок повинен мати достатній розмір, щоб мати змогу зберігати найдовший рядок та символ кінця рядка.

Змінна С-рядка оголошується, як звичайний масив символів. Наприклад,

```
1) char str15[16];       // оголошення рядка str15 з 15-ти символів,
                         // пам'ять в об'ємі 16 байтів виділяється компілятором
2) const int n = 10;  char str[n];
                         // оголошення рядка str з n - 1 (тобто 9ти) символів,
                         // пам'ять в об'ємі 10 байтів виділяється компілятором
```

Як і у випадку масиву, ім'я С-рядка є вказівником на нього. Отже, С-рядок можна оголосити як вказівник на тип char. Наприклад,

```
1) char *ptr;     // оголошення вказівника ptr на перший символ рядка,
                  // пам'ять під сам рядок не виділяється
2) int n = 10; char *str = new char[n];
                  //оголошення вказівника str на рядок з n - 1
                  //символів, пам'ять виділяється при виконанні
```

С-рядок можна як оголошувати, так і ініціалізувати. Ініціалізація рядка здійснюється під час його визначення за допомогою рядкових літералів або як перелік окремих символів. Якщо рядок при оголошенні ініціалізується, його розмірність можна опускати (компілятор сам виділить потрібну кількість байтів). Рядки у лапках завжди неявно містять нуль-символ, тому при ініціалізації прописувати його немає потреби. Окрім того, засоби введення символьних масивів автоматично долучають нуль-символ у кінець рядка. При оголошенні й ініціалізації рядка слід бути впевненим, що розмір рядка є достатній, щоб умістити всі символи рядка з нуль-символом. Річ у тім, що функції, які опрацьовують рядки, керуються позицією нуль-символу, а не розміром рядка.

```
Наприклад,          char hello[] = "Hello!";
                    // довжина рядка визначається автоматично,
                    // виділено й заповнено 7 байтів, // '\0' заноситься автоматично
   char abc[] = {'a', 'b', 'c', '\0'};    // '\0' заноситься явно
```

Не слід плутати символьну константу з рядком, що містить один символ: 'X' – це окремий символ (символьна константа), що займає 1 байт пам'яті; "X" – це рядок (рядкова константа), що складається з одного символу (букви X) та символу кінця рядка і займає 2 байти пам'яті.

Поширеним засобом доступу до символів рядка є константні вказівники типу char*. У прикладі const char *str = "Приклад рядка";

компілятор записує всі символи рядка і нуль-символ до масиву без імені та присвоює змінній str адресу першого елемента масиву.

Рядки можуть бути порожніми. Рядок вважається порожнім у двох випадках: якщо вказівник на рядок має нульове значення (рядка взагалі немає) чи

коли вказівник вказує на масив, який складається з одного нуль-символу (не містить жодного значущого символу). Наприклад, char* empty1 = nullptr; // empty1 не адресує жодного рядку char* empty2 = ""; // empty2 адресує нуль-символ

Обробка рядків [14]. С-рядки можуть оброблятися як цілісний об'єкт, а також посимвольно.

При посимвольній обробці С-рядків доступ до конкретного символу рядка здійснюється за індексом або за вказівником. Можна вказати індекс рядка і поза його поточною довжиною. У цьому випадку зчитані символи будуть випадковими.

С-рядки, як цілісні об'єкти, можна присвоювати, порівнювати і обробляти застосовуючи функції обробки рядків, які містяться у стандартної бібліотеці С++, яка підключається в заголовковому файлі <cstring>.

Дії по обробці рядків можна згрупувати наступним чином:

- визначення довжини рядка;
- копіювання рядка;
- об'єднання (конкатенація) рядків;
- порівняння рядків;
- аналіз символів рядка;
- пошук у рядку;
- перетворення рядка.

Визначення довжини рядка. Визначити поточну довжину рядка можна тільки після його ініціалізації. Для визначення поточної довжини рядка служить функція strlen. Параметром даної функції є рядок, значення довжини якого вона повертає.

Прототип задання функції **strlen: size_t strlen(char* str),**
де size_t – стандартний беззнаковий цілий тип. Значення довжини рядка, що повертає функція strlen, не включає нуль-символ. Значення довжини рядка дозволяє обробляти символи рядка за допомогою циклу for.

Копіювання рядка. У С/С++ можна копіювати вміст одного рядка в інший повністю або частково с початку.

Таблиця 5. Перелік основних функцій копіювання рядків у С/С++

Прототип функції	Опис
char* strcpy(char *str1, char *str2);	Копіює вміст рядка str2 у рядок str1
char* strncpy(char *str1, char *str2, size_t n);	Копіює n перших символів рядка str2 у рядок str1

Масив, який використовують для зберігання рядка str1, повинен бути достатньо великим, щоб в нього можна було помістити рядок з масиву str2. Інакше масив str1 переповниться, тобто відбудеться вихід за його межі, що може призвести до руйнування програми. Саме функція strcpy використовується для ініціалізації рядказмінної.

Об'єднання рядків. Над рядковими змінними означена операція об'єднання (конкатенації), яка дозволяє дописати один рядок в кінець іншого. Поточна довжина рядка, до якого приєднується інший рядок, збільшується на довжину

рядка, що додається. Збільшення довжини об'єднаного рядка здійснюється тільки у межах розміру рядка.

Таблиця 6. Перелік основних функцій об'єднання рядків у C/C++

Прототип функції	Опис
char *strcat(char *str1, char *str2);	Приєднує рядок str2 в кінець рядка str1
char *strncat(char *str1, char *str2, size_t n);	Приєднує n перших символів рядка str2 в кінець рядка str1

Порівняння рядків. Для рядків визначено операції порівняння «менше», «дорівнює», «більше». Принцип дії цих операцій базується на правилі порівняння даних символьного типу: з двох символів більшим вважається той, код якого має більше значення в кодовій таблиці символів (ASCII, UNICOD). Порівняння рядків виконується як серія послідовних порівнянь символів рядків, що мають однакові індекси, починаючи з першого. Порівняння здійснюється до першого незбіжного символу, і той рядок вважається більшим, у якому перший незбіжний символ має більше значення. Якщо рядки мають різну довжину, але в загальній частині символи збігаються, то вважається, що коротший рядок менше більш довгого (див. табл. 7).

Дані функції повертають значення 0, якщо рядки однакові (з урахуванням або без урахування регістра). Якщо рядок str1 відповідно до алфавітного порядку більший від рядка str2, то повертається додатне число; якщо ж менший від рядка str2 – то повертається від'ємне число.

Таблиця 7. Перелік основних функцій порівняння рядків у C/C++

Прототип функції	Опис
int strcmp(char *str1, char *str2);	Порівнює рядки str1 і str2
int stricmp(char *str1, char *str2);	Порівнює рядки str1 і str2 без урахування регістра
int strncmp(char *str1, char *str2, size_t n);	Порівнює перші n символів рядків str1 і str2
int strnicmp(char *str1, char *str2, size_t n);	Порівнює перші n символів рядків str1 і str2 без урахування регістра

Аналіз символів рядка. Іноді виникає необхідність у перевірці приналежності символів до певної класифікаційної групи: літера, цифра, знак пунктуації тощо.

Таблиця 8. Перелік основних функцій C/C++ для аналізу символів

Прототип функції	Перевіряє, чи є аргумент ch
int islower(int ch);	рядковою літерою
int isalnum(int ch);	алфавітно-цифровим символом
int isalpha(int ch);	літерою
int isupper(int ch);	прописною літерою
int isdigit (int ch);	цифровим символом
int ispunct(int ch);	знаком пунктуації
int isspace(int ch);	пробілом

Якщо символ належить до відповідної класифікаційної групи, то відповідна повертає ненульове значення, інакше – 0.

Пошук у рядку. Часто у програмах доводиться виконувати пошук у рядках окремих символів або підрядків. У C/C++ передбачені наступні можливості пошуку у рядках:

– пошук символів;

– пошук підрядків;

– пошук лексем.

Функція strchr повертає вказівник на символ ch в рядку, що адресується вказівником str. Якщо такий символ не знайдений, функція strchr повертає NULL. Також можна застосувати функцію strchr для присвоювання вказівника на підрядок, що починається з заданого символу.

Таблиця 9. Перелік основних функцій C/C++ для пошуку символів у рядках

Прототип функції	Опис
char *strchr(char *str, int ch);	Шукає перше входження символу ch у рядок str
char *strrchr(char *str, int ch);	Шукає останнє входження символу ch у рядок str
char *strpbrk(char *str1, char *str2);	Шукає перше входження будь-якого символу з рядка str2 у рядок str1

Можна також організувати пошук підрядків вказаного рядка. Подібно strchr, функція strstr повертає адресу підрядка або NULL, якщо шуканий рядок не знайдений. Якщо заданий підрядок знайдений, вказівник стане рівним його адресі.

Таблиця 10. Перелік основних функцій C/C++ для пошуку підрядків у рядках

Прототип функції	Опис
char *strstr(char *str1, char *str2);	Шукає перше входження підрядка str2 у рядок str1
size_t strcspn(char *str1, char *str2);	Визначає довжину початкового інтервалу рядка str1, що не містить символів з рядка str2; тобто, повертає індекс першого символу рядка str1, який збігається з будь-яким із символів в рядку str2
size_t strspn(char *str1, char *str2);	Визначає довжину початкового інтервалу рядка str1, що містить символи з рядка str2

Іноді виникає необхідність у розкладанні рядка на лексеми. Якщо лексеми рядка відокремлені одна від одної комами, пробілами або іншими роздільниками, то для цього можна використати функцію strtok. Прототип даної функції: char *strtok(char *str1, char *str2); Ця функція дозволяє перетворити вихідний рядок str1 в послідовність лексем, розділених символами, що містяться у рядку роздільників str2. Якщо лексема знайдена, то повертається вказівник на неї, інакше – NULL. Якщо у рядку декілька лексем, то дана функція викликається повторно: при першому виклику їй передається адреса вихідного рядка str1, потім – NULL. Пошук продовжується, доки strtok не поверне NULL.

Після першого виклику функція strtok повертає адресу першої лексеми, відокремленої від наступних одним із заданих роздільників і вставляє '\0' в позицію першого заданого обмежувача. Тим самим створюється підрядок, адреса якого повертається функцією strtok. Якщо задані роздільники не виявлені, функція strtok повертає NULL.

Окрім того, ця функція налаштовує свій внутрішній вказівник на символ, наступний за кінцем останнього сформованого підрядка, щоб наступний виклик strtok міг продовжити розкладання рядка. Для цього слід передати як перший аргумент значення NULL – це послужить сигналом для функції strtok використовувати її внутрішній вказівник як стартову адресу для пошуку іншого роздільника. Таким чином, щоб виділити інші лексеми рядка, потрібно просто викликати функцію strtok кілька разів, і кожного разу першим її аргументом має бути NULL.

Перетворення рядків.

Таблиця 11. Перелік основних функції перетворення рядків у C/C++

Прототип функції	Опис
char *strlwr(char *str);	Приводить символи рядка до нижнього регістра
char *strupr(char *str);	Приводить символи рядка до верхнього регістра
char *strrev(char *str);	Інвертує рядок

До функцій перетворення рядків також відносять функції перетворення рядка у відповідне числове значення і функції перетворення числових значень в рядкові.

Основні такі функції наведені у табл. 8.

Таблиця 12. Перелік основних функції перетворення рядків і чисел у C/C++

Прототип функції	Опис
double atof(const char *str);	Перетворює рядок str в число з плаваючою крапкою типу double; якщо рядок str не є коректним текстовим поданням числа double, то повертається NULL (заголовний файл – math.h)
int atoi(const char *str);	Перетворює рядок str в число типу int (заголовний файл – stdlib.h)
long atol(const char *str);	Перетворює рядок str в число типу long (заголовний файл – stdlib.h)
char *itoa(int value, char *str, int radix);	Перетворює значення цілого типу value в рядок str з урахуванням зазначеної системи числення radix
char *gcvt(double value, int ndig, char *buf);	Перетворює число з плаваючою крапкою value типу double в рядок довжиною ndig цифр, який адресується аргументом buf.

Загальні відомості про файли [4, 5, 6, 10, 11, 13]**.** Файл – це іменований блок інформації, який зберігається на носії інформації – будь-якому фізичному пристрої зчитування/запису. Введення даних у програму може здійснюватися з різних пристроїв – консолі (клавіатури), різноманітних накопичувачів; виведення даних – на консоль (монітор), принтер, дискові накопичувачі тощо. Тобто, як джерелами, так і приймачами інформації при такому обміні можуть бути пристрої, які на апаратному рівні суттєво відрізняються один від одного.

Текстові файли призначено для зберігання текстів – сукупності символьних рядків змінної довжини. У текстовому файлі текстові рядки закінчуються керуючою послідовністю. В Windows кінець рядка кодується послідовністю двох символів – '\r' (код 0D – «повернення каретки») і '\n' (код 0A – «перехід на новий

рядок»). Інформація в текстових файлах має такий самий вигляд, як і на екрані, тобто її можна читати. Відповідно текстові файли можна переглядати і змінювати вручну текстовим редактором.

Бінарні файли використовуються для збереження даних у внутрішньому поданні. При роботі з бінарним файлом на диск записуються точні копії даних з оперативної пам'яті і в оперативну пам'ять зчитуються точні копії даних з диска. При перегляді такого файлу неможливо зрозуміти, що в ньому записано – можна побачити лише послідовність байт. Щоб зрозуміти, що записано в бінарному файлі, потрібно знати його структуру. Такий файл не можна створювати або виправляти у текстовому редакторі. Однак всі ці незручності компенсуються швидкістю роботи з даними.

Бінарні файли поділяють на типізовані та нетипізовані. Типізовані файли складаються з компонент строго визначеного типу. Компоненти такого файлу можуть бути будь-якого типу, окрім файлового. Нетипізовані файли не мають жорстко встановленої одиниці читання/запису і можуть розглядатися як сукупність байтів.

Хоча дані у файл записуються послідовно, у порядку їхнього надходження, оброблятися компоненти файлів можуть двома способами:

– послідовний доступ, коли черговий елемент можна прочитати/записати тільки після аналогічної операції з попереднім елементом;

– прямий доступ, коли читання/запис довільного елемента здійснюється безпосередньо за його номером (адресою).

Файл, байти якого можуть бути зчитані тільки послідовно, називають файлом послідовного доступу; файл, байти якого можуть бути зчитані в довільному порядку є файлом прямого доступу. Послідовний доступ можливий для всіх типів файлів, прямий – тільки для бінарних.

Доступ до компонент файлу здійснюється за допомогою файлового вказівника (file pointer) – спеціальної змінної, яка неявно зв'язується з кожним файлом. На початку роботи з файлом файловий вказівник автоматично встановлюється на початок файлу (елемент з фізичним номером 0). При обробці компонентів файлу він послідовно переміщується до потрібного елемента і робить його доступним для обробки.

Технологія роботи з файлами даних [10, 11, 13]**.** Будь-який файл повинен бути спеціальним чином підготовлений до роботи. Також певним чином потрібно завершити роботу з ним.

Типовий сценарій роботи з файлами даних включає наступні кроки: - зв'язування файлової змінної (потоку) з фізичним файлом;

– відкриття файлу;

– його обробка (читання/запис);

– закриття файл.

Перед початком роботи будь-який фізичний файл необхідно зв'язати з ідентифікатором потоку, який буде представляти цей файл у програмі.

Відкриття файлу означає, що потік з'єднують з файлом. Програма "захоплює" заданий фізичний файл і повідомляє ОС, що далі вона буде працювати з

цим файлом. Такий крок потрібен, щоб не виникало конфліктів при спробі одночасного запису інформації в один і той же файл різними програмами. При відкритті файлу виконуються наступні дії:

– пошук фізичного файлу на зовнішньому носії;

– утворення файлового буферу для обміну з фізичним файлом; - встановлення файлового вказівника на початок потоку.

В операції відкриття файлу уточнюється, в якому режимі файл відкривається: "читання", "запису" або "читання і запису". При відкритті файлу також вказується його тип: текстовий або бінарний.

Обробка файлу полягає у реалізації операцій зчитування даних із файлу або запису даних у файл. Операції читання та запису завжди виконуються, починаючи з поточної позиції у файлі, яка встановлюється при відкритті файлу. При виконанні кожної операції введення/виведення вказівник поточної позиції файлу зміщується на одну позицію в сторону кінця файлу.

При роботі з файлами даних слід відрізняти власне обробку даних від їх зчитування з файлу або запису у файл. Обробка даних є внутрішньою справою програми і не має відношення до роботи з файлами. Робота ж з файлом полягає саме у реалізації файлового введення/виведення. Тому основними діями при роботі з файлами є операція зчитування – копіювання значення доступного елемента файлу у змінну програми, та операція запису – копіювання значення змінної у поточний елемент файлу.

Логічним завершенням роботи з відкритим файлом є його від’єднання від потоку. Цю операцію називають закриттям файлу. При закритті логічного файлу, відкритого для запису, всі дані, що накопичилися у буфері, записуються («скидаються») у фізичний файл, а сам буфер, утворений при відкритті файлу, ліквідується. Цей процес гарантує, що ніяка інформація випадково не залишиться в буфері. Після цього потік можна зв'язати з іншим файлом, а закритий файл стає доступним для обробки іншими програмами. Якщо програма завершує свою роботу нормально, то всі файли закриваються автоматично; у разі аварійного завершення роботи програми, файли не закриваються.

Файлове введення-виведення у C/C++, як правило, здійснюється або за допомогою бібліотечних функцій (в стилі C), або з використанням бібліотеки класів C++ (в стилі C++).

Робота з файлами у стилі С. [4]

Оголошення файлової змінної. Для роботи з файлами використовують змінну-вказівник типу FILE*. Цей тип оголошує вказівник потоку, який використовується надалі у всіх операціях з цим файлом.

Тип FILE – це структура, яка містить відомості про файл: його ім'я, місце розташування, режим роботи, вказівник поточної позиції тощо. Тип FILE означено у бібліотеці <stdio.h>. Якщо в програмі передбачається робота з файлами, цей заголовний файл слід долучити до програми за допомогою директиви #include.

Відкриття файлу Для з’єднання потоку з файлом використовується функція fopen, яка має прототип:

FILE* fopen(const char* filename, const char* mode);

Параметр filename – це рядок, що визначає повне ім’я фізичного файлу. Параметр mode – це рядок символів, що визначає режим відкриття файлу (табл. 9).

До зазначених специфікаторів наприкінці чи перед знаком "+" може дописуватись специфікатор типу файлу: "t" – для текстових файлів, "b" – для бінарних файлів.

Наприклад,

```
FILE* f; f = fopen("D:\file.txt", "w+t");
                    //створення текстового файлу в режимі читання/запису
```

Відкрити файл можна і при створенні відповідного файлового потоку. Наприклад,

Таблиця 13. Специфікатори режиму відкриття файлів

Режим	Призначення
r	відкрити існуючий файл тільки для зчитування даних (читання)
w	створити файл тільки для запису даних (якщо відповідний фізичний файл існує, то його вміст втрачається)
a	створити новий файл або відкрити існуючий файл для запису даних в кінець файлу (дозапис)
r+	відкрити існуючий файл для зчитування і запису даних
w+	створити новий файл для зчитування і запису даних (якщо відповідний фізичний файл існує, то його вміст втрачається)
a+	створити новий файл або відкрити існуючий файл для зчитування і записування даних в кінець файлу (дозапис)

```
FILE* f = fopen("h:\file.txt", "r");
```

У разі вдалого відкриття файлу функція fopen повертає вказівник на структуру типу FILE, що описує параметри відкритого файлу. Цей вказівник потім використовується для звертання до файлу у всіх файлових операціях.

Для уникання помилок після відкриття файлу слід перевіряти, чи насправді файл відкрився. Якщо файл відкрити не вдалося, наприклад, при спробі відкрити для читання неіснуючий файл, то повертається нульовий вказівник NULL. При цьому глобальна системна змінна errno містить код помилки. За необхідності цей код можна вивести:

```
FILE* f;
if ((f = fopen("h:\file.txt", "r"))== NULL)
{
      cout << "Error opening file with code " << errno << endl;
      exit(1);
};
```

Обробка файлу. Обробка файлу полягає у реалізації операцій зчитуванню даних із файлу або запису даних у файл. Операції читання/запису завжди виконуються, починаючи з поточної позиції у файлі. Початкова позиція встановлюється при відкритті файлу і може відповідати початковому або кінцевому байту файла в залежності від режиму відкриття файлу. При відкритті файлу в режимах читання/запису вказівник поточної позиції встановлюється на початок файлу,

при відкритті в режимі дозапису – в кінець файлу. При виконанні кожної операції читання або запису вказівник переміщується на нову поточну позицію у відповідності із числом записаних або прочитаних байтів.

У мові С передбачено декілька різновидів засобів файлового введення-виведення:

– символьні (функції fgetc і fputc);
– рядкові (функції fgets і fputs);
– блокові (функція fread і fwrite);
– форматовані (fscanf і fprintf).

При виконанні операції введення/виведення вказівник поточної позиції файлу зміщується в сторону кінця файлу. В якийсь момент він досягає кінця файлу. Для перевірки цього факту використовується функція feof: int feof(FILE* stream);

Функція повертає значення, відмінне від нуля (true), якщо всі елементи файлу прочитані і вказівник файлу вказує на кінець файлу або файл після відкриття виявився порожнім; у всіх інших випадках функція повертає значення 0 (false). Використовуючи функцію у циклі, можна читати вміст файлу, поки не буде досягнутий його кінець:

```
while (! feof(f))
{ … };
```

Закриття файлу. Логічним закінченням роботи з відкритим файлом є його закриття. Для закриття файлу у мові С служить функція fclose:

```
int fclose(FILE *filename);
```

Вона повертає 0 при успішному закритті файлу і EOF (-1) – якщо при закритті файлу виникла помилка. Змінна errno буде містити код помилки.

Вказівник позиції файлу. Для кожного файлу, після його відкриття, визначається вказівник позиції, який вказує на зміщення від початку файлу у байтах.

Для роботи з вказівником позиції призначені функції fseek, fsetpos, ftell і fgetpos.

Робота з файлами у стилі С++ [4, 5, 6]**.** В бібліотеці С++ для введення/виведення файлів є класи ifstream (для файлових вхідних потоків), ofstream (для файлових вихідних потоків) і fstream (для файлових двонаправлених потоків). Класи файлового введення/виведення визначені у файлі заголовків fstream.h.

Класи файлового введення/виведення містять спеціальні методи роботи з файлами – відкриття, перевірки стану, роботи з поточної позицією, закриття файлів, а також введення/виведення і форматування інформації.

В програмі файловий потік створюється (оголошується) як об’єкт одного з класів файлового введення/виведення, наприклад: ifstream in; //створення потоку in для файлового введення ofstream out; //створення потоку out для файлового виведення.

Зв’язування потоку з конкретним файлом відбувається одним зі способів:

1) при створенні потоку, наприклад:

ofstream out("d:\\test.bin"); // створення потоку out для файлового виведення,

// відкриття і зв'язування його з файлом

2) прі відкритті вже створеного потоку методом open():

out.open("d:\\test.bin"); // відкриття раніше створеного потоку out

// і зв'язування його з файлом

Закриття файлу відбувається за допомогою виконання методу close(), наприклад: in.close();

Або автоматично при знищенні потоку при завершенні програми. Метод close() не має параметрів і не повертає ніякого значення.

Режими відкриття файлів [4, 5, 6, 10]**.** При відкритті файлу можна вказати не тільки його ім'я, а і встановити режими відкриття файлів (таблиця 1).

Наприклад, відкриття файлу для читання у бінарному режимі:

ifstream in;

in.open("test.dat", ios::in | ios::binary);

Без явного вказання режиму відкриття файлу для потоків ifstream файли відкриваються у режимі ios::in, для ofstream – ios::out | ios::trunc, для fstream – ios::in | ios::out.

Якщо стандартне значення режиму відкриття файлу не влаштовує програміста, можна вказати інше значення, склавши його з одного або кількох значень з таблиці 1, з'єднаних операцією побітове АБО (|).

Перевірка стану файлу і потоку [4, 5, 6]**.** Відкриття файлу і пов'язаного з ним потоку може виявитися не успішним, наприклад, через відсутність вказаного файлу або неможливість його створення. Існує кілька методів, які перевіряють стан файлу і потоку.

Успішність виконання відкриття потоку можна перевірити за допомогою методу is_open, наприклад, if (! in.is_open()) cout << "Input file can't be opened.\n";

При невдалому відкритті файлу пов'язаний з ним потік отримує нульове значення, тому ця перевірка може бути записана і так:

if (!in) cout << "Input file can't be opened.\n";

Таблиця 14. Специфікатори режиму відкривання файлів

Параметр	Опис
ios::app	відкрити файл в режимі додавання, встановлюючи файловий вказівник на кінець файлу.
ios::ate	встановити файловий вказівник на кінець файлу
ios::in	відрити файл для читання
ios::nocreate	якщо й файл не існує, не створювати файл і повернути помилку
ios::noreplace	якщо файл існує, операція відкриття повинна бути перервана і повинна повернути помилку
ios::out	відкрити файл для запису
ios::trunc	перезаписати вміст існуючого файлу
ios::binary	відкрити файл у бінарному режимі

Перевірити, чи досягнене кінець вхідного файлу, можна за допомогою методу eof(). При його використанні слід пам’ятати, що він повертає значення true тільки після спроби читати за границею файлу.

Методи введення/виведення [4, 5, 6]**.** Для введення і виведення даних при роботі з файлами використовують:

1) перевизначені операції << і >>;
2) методи get() і put() для посимвольної (побайтової) передачі даних;
3) метод getline() з двома або з трьома параметрами для введення рядків;
4) методи read() і write():

ifstream& read(char* buf, int nCount);

ofstream& write(const char* buf, int nCount);

Метод read вводить вказану в nCount кількість байтів в пам’ять, починаючи з адреси buf. Метод write виводить вказану в nCount кількість байт, які розташовані в пам’яті, починаючи з адреси buf.

Якщо дані, що вводяться або виводяться, не є символьними, наприклад, це числа, масиви чисел, структури, необхідно виконати перетворення адрес даних, що передаються методам read() і write() до типу char*. В цьому випадку дані, починаючи з вказаної адреси, будуть розглядатися як коди символів і їх введення/виведення відбудеться без будь якого проміжного перетворення.

Наприклад,

```
int nums[10];
for ( int i = 0; i < 10; ++i )
   nums[i] = i + 1;
ofstream out( "test.dat", ios::out | ios::trunc | ios::binary );
out.write( (char*) nums, sizeof nums );
```

Спочатку з масивом nums програма працює як з масивом цілих чисел, а після перетворення до типу char* він використовується як масив байтів і записується у файл на диску. Для визначення розміру масиву використовується оператор sizeof.

Використовування поточної позиції файлу [4, 5, 6, 10, 11, 13] Система введення/виведення C++ дозволяє здійснювати довільний доступ з використанням функції seekg () і seekp (): istream & seekg (streamoff offset, seek_dir origin); ostream & seekp (streamoff offset, seek_dir origin);

Тип streamoff визначено в заголовку iostream.h за допомогою функції typedef як long. Тип streamoff визначає область значень, які може приймати величина offset.

Система введення/виведення C++ обробляє два вказівника, асоційовані з кожним файлом. Один з них, *get pointer*, визначає, де саме в файлі буде проводитися наступна операція введення. Інший вказівник *put pointer* вказує, де саме в файлі буде проводитися наступна операція виведення. Всякий раз, коли здійснюються операції введення або виведення, відповідний вказівник автоматично

переміщується. За допомогою функцій seekg () і seekp () можна отримати доступ до даних у файлі в довільному місці.

Функція seekg () переміщає вказівник *get pointer* на число байт offset від заданого origin, що приймає одне з наступних трьох значень: ios::beg – початок файлу; ios::cur – поточна позиція файлу; ios::end – кінець файлу.

Інший варіант seekg – з одним параметром, який задає зсув позиції. У якості точки зсуву використовується початок файлу.

Функція seekp () переміщає вказівник *put pointer* на offset байт від заданого origin. Наприклад,

```
fstream io( "d:\\temp\\test.dat", ios::in | ios::out );
char ch;
io.seekg( -5, ios::end ); // зсув на 5 байтів з кінця файлу позиції вказівника
io.get( ch );             // читання п’ятого від кінця файлу байту
io.seekp ( 10 );          // зсув на 10 байтів від початку файлу вказівника put pointer
io.put( '*' );        // запис символу '*' в одинадцятий байт файлу
```

Поточну позицію у файлі визначають методами tellg() при *введенні* і tellp() – при *виведенні*. Методи tellg() і tellp() не мають параметрів і повертають поточні позиції відповідних вказівників пов’язаних з потоком файлу. Використання цих методів дозволяє зберегти поточне значення позиції файлу, виконати якісь операції введення/виведення, а потім відновити попереднє значення позиції у файлі: long p = io.tellg();

```
…
io.seekg(p);
```

Завдання

Розробіть програми для вирішення задач 1 та 2 згідно індивідуального завдання.

Завдання А.

Розробити програму для обробки текстового файлу для заданого індивідуального завдання. Результати роботи програми записати в текстовий файл.

Вимоги до програми:

– для обробки рядків текстового файлу використовуються рядки в стилі С;

– для роботи з рядками максимально використовуються функції стандартної бібліотеки обробки рядків (забороняється дублювати дії функцій стандартної бібліотеки власними функціями);

– імена вхідного і вихідного файлів вводяться з клавіатури або задаються як аргументи командного рядку;

– файлове введення-виведення здійснюється за допомогою бібліотечних функцій (в стилі С);

– читання/запис проводяться виключно за допомогою рядків;

– операцій роботи з файлами перевіряються на помилку.

Завдання В.

Розробити програму роботи з бінарними файлами:

1) ініціалізувати в програмі масив записів для предметної області з лабораторної роботи №4 «Структурний тип даних в мові С++»;

2) записати дані з масиву в бінарний файл;

3) обробити отриманий бінарній файл за індивідуальним завданням;

4) вивести оброблений файл на консоль.

Вимоги до програми:

– імена файлів задавати в командному рядку;

– дані для запису у файл описати в масиві структур з 8 – 10 елементів;

– зчитані з файлу дані забороняється зберігати в масиві;

– забороняється зберігати в оперативної пам'яті одночасно більше двох зчитаних з файлу блоків;

– перевіряти всі операції роботи з файлами (відкриття, читання, запису).

Індивідуальні завдання

1) У вхідному файлі циклічно зсунути в бік початку файлу перші три блоки.

2) Переписати з вхідного файлу у вихідний парні блоки.

3) Переписати з вхідного файлу у вихідний три перших і два останніх блоки.

4) Переставити в вхідному файли пари блоків – перший і другий, третій і четвертий, і т. д. Блок, у якого немає пари, залишити без змін.

5) У вхідному файлі циклічно зсунути в бік кінця файлу перші три блоки.

6) Переписати з вхідного файлу у вихідний кожен третій блок.

7) Переписати з вхідного файлу у вихідний кожен другий блок, починаючи з третього.

8) Переписати з вхідного файлу у вихідний два перших і три останніх блоки.

9) Переписати у вхідному файлі блоки у зворотному порядку.

10) У вхідному файлі циклічно зсунути в бік початку файлу останні три блоки.

11) У вхідному файлі записати два перші та два останні блоки.

12) Переписати з вхідного файлу у вихідний непарні блоки.

13) Переписати з вхідного файлу у вихідний два перших і три останні блоки.

14) Переставити в вхідному файли пари блоків – другий і третій, п'ятий і четвертий, і т. д. Блок, у якого немає пари, залишити без змін.

15) У вхідному файлі циклічно зсунути в бік кінця файлу перші чотири блоки.

16) Переписати з вхідного файлу у вихідний кожен непарний блок.

17) Переписати з вхідного файлу у вихідний кожен другий блок, починаючи з четвертого.

18) Переписати з вхідного файлу у вихідний два перших блоки і один останній.

19) Переписати у вхідному файлі спочатку два остання блоки, а потім два перші блоки.

20) Переписати з вхідного файлу у вихідний кожний третій блок починаючі із третього.

Контрольні питання

1. Що таке файл?
2. Які є особливості роботи з текстовими файлами?
3. Які є особливості роботи з бінарними файлами?
4. Які функції роботи із файлами у підході С ви знаєте? Надайте алгоритм застосування цих функцій із файлами.
5. Які функції роботи із файлами у підході С++ ви знаєте? Надайте алгоритм застосування цих функцій із файлами.

Контрольні питання та завдання підвищеної складності

1. Поясніть відмінності між текстовими та бінарними файлами.
2. Опишіть основні функції роботи з рядками.

ЛАБОРАТОРНА РОБОТА № 6. РЕКУРСІЯ ТА ІТЕРАЦІЯ.

Тема. Рекурсія та ітерація.

Мета. Навчитися описувати повторюванні обчислювальні процеси у рекурсивної та ітераційної формах.

Зміст роботи:

1) ознайомитися з теоретичними відомостями до лабораторної роботи;
2) проаналізувати завдання і розробити зовнішні специфікації до програми (вхідні та вихідні дані, функціональні вимоги до програми);
3) розробити набори тестів для перевірки правильності алгоритму;
4) розробити рекурсивний та ітераційний алгоритми обчислення функції;
5) розробити і налагодити програму, переконатися в правильності реалізації завдання за допомогою набору тестів;
6) скласти звіт з лабораторної роботи.

Зміст звіту:

1) титульний аркуш (вид та номер роботи, дисципліна, П.І.Б. та група виконавця);
2) тема і мета лабораторної роботи;
3) постановка завдання (завдання до лабораторної роботи, індивідуальне завдання);
4) зовнішні специфікації програми (вхідні та вихідні дані, функціональні вимоги до програми);
5) рекурсивний та ітераційний алгоритми розв’язання завдання (схема Нассі-Шнайдермана);
6) набор тестів для перевірки правильності алгоритму обчислення функції;

7) текст програми;

8) результати тестування програми та їх аналіз;

9) висновки щодо використання рекурсії та ітерації при обчисленні функції.

Теоретичні відомості

Рекурсивний метод у програмуванні. Рекурсивний метод у програмуванні передбачає розв'язання задачі, ґрунтуючись на властивостях рекурсивності самої задачі: вихідна задача зводиться до вирішення аналогічних підзадач, які є простішими. При виконанні рекурсивного алгоритму здійснюється перехід від початкового рівня послідовно через поточні до нижнього рівня доти, поки, не буде отримано тривіальне рішення завдання, після чого здійснюється послідовний перехід в зворотному напрямку.

Розробці рекурсивних алгоритмів передують етапи моделювання завдання, на яких визначається набір параметрів і співвідношень між ними: параметризація, виділення бази і декомпозиція.

На етапі параметризації з постановки задачі виділяються параметри, які описують вихідні дані. При цьому рекурсивний алгоритм може вимагати введення додаткових параметрів, яких немає в умовах задачі. Необхідність в додаткових параметрах часто виникає також при вирішенні задач оптимізації рекурсивних алгоритмів, в ході яких скорочується їхня часова складність.

Виділення бази рекурсії передбачає виокремлення в розв'язувальної задачі тривіальних випадків, результат для яких є очевидним і не потребує проведення розрахунків. Вірно визначена база рекурсії забезпечує завершеність рекурсивних звернень.

Декомпозиція задачі полягає в тому, що задача послідовно розбивається на підзадачі двох типів: ті, рішення яких відомо, і ті, які аналогічні до вихідної задачі. Декомпозицію треба здійснювати так, щоб вона обов'язково призвела хоча б до одного з виділених у якості бази тривіального випадку, тобто до задачі з набором параметрів, що є індикатором завершення рекурсивних викликів. Кожна з отриманих задач другого типу повинна бути спрощеним варіантом попередньої задачі. Спосіб приведення задачі/підзадачі до спрощеного варіанту називають кроком рекурсії.

Рекурсивні функції та механізми їх реалізації **.**Рекурсивна функція - це функція, яка викликає саму себе. Це в разі прямої рекурсії. Існує і непряма рекурсія – коли дві або більше функції викликають одна одну.

Основні правила побудови рекурсивних функцій:

– визначити аргументи функції (кількість і призначення);

– визначити результат і спосіб його формування;

Рис. 9. Пряма і непряма рекурсії

– задати, щонайменше, одну умову закінчення рекурсії;

– описати формування нових значень аргументів для рекурсивного виклику.

Правила побудови визначають, що рекурсивні функції складаються принаймні з двох гілок: термінальної і рекурсивної.

Термінальна гілка необхідна для закінчення обчислень. Без термінальної гілки рекурсивний виклик був би нескінченним. Термінальна гілка повертає результат, який є базою для обчислення результатів рекурсивних викликів. Після кожного виклику функцією самої себе, обчислення повинні наближатися до термінальної гілки.

В рекурсивній гілці відбувається самовиклик функції. Самовиклик функції нічим не відрізняється від виклику якоїсь іншої функції. Реалізація рекурсивних викликів функції спирається на типовий механізм реалізації виклику функції, заснований на збереженні контексту функції (адреси повернення, параметрів і локальних змінних) в стеку:

1) у точці виклику в стек поміщаються параметри, що передаються функції, і адреса повернення;

2) управління виконанням програми передається у функцію, яка в ході роботи розміщує в стеку власні локальні змінні;

3) по завершенні обчислень функція очищає стек від своїх локальних змінних, записує результат;

4) команда повернення з функції зчитує з стека адресу повернення і виконує перехід за цією адресою. Або безпосередньо перед, або відразу після повернення з функції стек очищається від параметрів.

Якщо функцію викликати повторно з іншої функції чи з неї самої, буде виконуватись той самий код, але працювати він буде з іншими значеннями параметрів і локальних змінних. Це і дає можливість рекурсії. Кожний черговий рекурсивний виклик приводить до створення нового набору її параметрів і локальних змінних, що відповідають ланцюжку незакінчених рекурсивних викликів, які існують незалежно один від одного.

Коли обчислювальний процес доходить в рекурсивної гілці до виклику функції самої себе, то виконання поточної функції призупиняється, для нової функції створюється новий контекст і ця нова (така ж саме) функція запускається з початку, але вже на новому рівні. Виконання перерваної функції буде продовжено після повернення з рекурсивного виклику нової функції. Нова функція, в свою чергу, теж може перервати своє виконання з тієї же причини і так далі. Таким чином, утворюється стек перерваних функцій, з яких виконується лише остання викликана функція. Після її завершення виконуватиметься попередня до неї функція. Цілком весь рекурсивний виклик буде виконано, якщо стек перерваних функцій буде порожній.

Формування стеку перерваних функцій при виклику рекурсивних функцій називають розгорткою рекурсії, а вилучення елементів з цього стеку – згорткою рекурсії.

Контекст рекурсивної функції обмежує максимальну глибину рекурсії (кількість одночасно виконуваних функцій) об'ємом стека. При надмірно великої глибині рекурсії може відбуватися переповнення стеку.

Рекурсія vs ітерація [10, 11, 13]**.** Програми, написані різними способами, за своїми властивостями помітно відрізняються одна від одної. Факторами, що впливають на розв'язок, можуть бути ефективність обчислень в плані часу виконання чи обсягу пам'яті, довжини програми, її прозорості і зрозумілості тощо.

Рекурсивний алгоритм завжди можна замінити ітераційним (який використовує цикли). Однак, для нетривіальних випадків, рекурсивна версія часто буває набагато простіше як для написання, так і для читання, і для зрозуміння. Наприклад, функцію обчислення n-го числа Фібоначчі чи швидкого сортування можна написати і за допомогою методу ітерацій, але це буде складніше.

Використання рекурсії не дає ніяких переваг з точки зору продуктивності. Ітеративні функції майже завжди більш ефективні, ніж їх рекурсивні аналоги. Наприклад, якщо рекурсивна функція містить великі обсяги локальних внутрішніх змінних і велику кількість параметрів, то використання рекурсії не є ефективним. Це пов'язано з тим, що для кожного рекурсивного виклику потрібно робити копії цих змінних і параметрів. При великій кількості рекурсивних викликів це призведе до надмірного використання пам'яті і часу. Ітеративні функції не витрачають цих ресурсів. Рекомендується використовувати ітерацію, замість рекурсії, але в тих випадках, коли це дійсно практичніше.

Це не означає, що ітеративні функції завжди є кращим варіантом. Іноді рекурсивна реалізація може бути чистіше і простіше, а додаткові витрати можуть бути виправдані, якщо вони зводять до мінімуму труднощі майбутньої підтримки коду.

Загалом, рекурсія є хорошим вибором, якщо:

– рекурсивний код набагато простіше зрозуміти і реалізувати;

– глибина рекурсії може бути обмежена;

– це не критична частина коду, яка безпосередньо впливає на продуктивність програми.

Рекурсія добре підходить для реалізації алгоритмів обходу списків, дерев, аналізаторів виразів, комбінаторних задач тощо.

Приклад розробки рекурсивної функції. Розробити програму обчислення значення функції

$$f(x,n)=\frac{x^n}{n!}$$

при будь-якому дійсному *x* і будь-якому натуральному *n*.

Рішення. В області невід'ємних чисел простішими, як правило, вважаються менші за величиною числа. Наприклад, для числа N більш простими є числа N - 1, N - 2 і т.д. Очевидно, таке зменшення числа можна зробити тільки кінцеву кількість разів – до 0.

Перш за все, обчислення $f(x, n)$ треба звести до підзадачі – до обчислення $x^k / k!$ при деякому $k < n$. Можна, наприклад, вибрати $k = n - 1$, тому що по $x^{n-1} / (n - 1)!$ легко отримати $x^n / n!$

Оскільки ми описуємо $f(x, n)$ в припущенні, що функція правильно обчислює $x^k / k!$ при будь-якому $k < n$, то в описі функції ми повинні для обчислення $x^{n-1} / (n - 1)!$ рекурсивно звернутися до $f(x, n - 1)$, а потім отриману величину помножити на x / n, щоб отримати значення $f(x, n)$. База рекурсії (нерекурсивний випадок) – $f(x, 0)$, тому що обчислення $x^0 / 0!$ не можна звести до обчислення $x^k / k!$ при $k < 0$. Схема виконання обчислення функції $f(2, 3)$ з використанням звичайної рекурсії наведена на рис. 10.

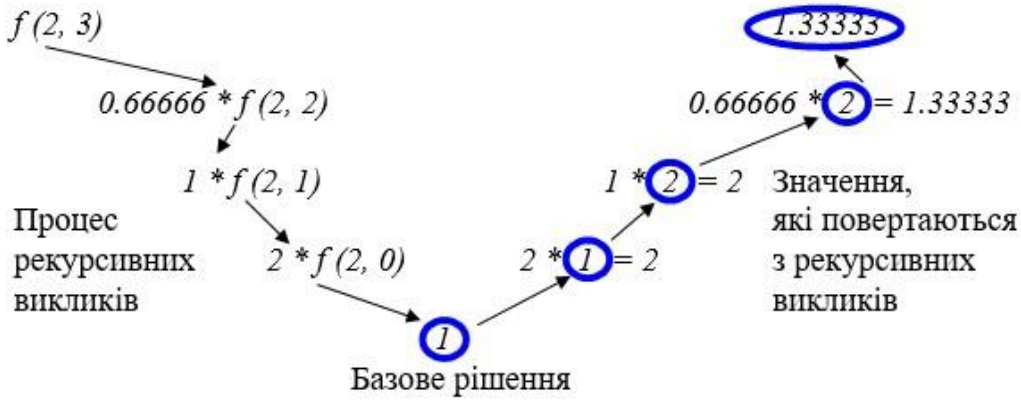


Рис. 10. Схема рекурсивних викликів та повернень з рекурсії

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

double recursion(float x, unsigned int n)
{
    // терминальна гілка
    if (n == 0) // базовий випадок
        return 1;
    // рекурсивна гілка
    return (x / n) * recursion(x, n - 1); // крок рекурсії
}

int main()
{
    cout << "(2^3 / 3! = "
        << recursion(2, 3) // виклик рекурсивної функции
        << endl;
    return 0;
}
```

Завдання

Для заданого індивідуального завдання розробити програму обчислення значення функції рекурсивним та ітераційним способом. Порівняти результати обчислення функції рекурсивним та ітераційним способом з результатом обчислення лівої частини виразу, якщо це можливо.

Вимоги до програми:

– дані для обробки ввести з консолі;
– за необхідністю перевірити вхідні дані на валідність;
– обчислення ітераційним методом оформити у вигляді функції;

– результати роботи програми вивести на консоль.

Таблиця 15. Індивідуальні завдання

Варіант	Завдання
1.	$sh\,(x)=\frac{x}{1!}+\frac{x^3}{3!}+\frac{x^5}{5!}+\frac{x^7}{7!}+\cdots+\frac{x^{2n+1}}{(2n+1)!}+\cdots=\sum_{n=0}^{\infty}\frac{x^{2n+1}}{(2n+1)!}$, $-\infty<x<+\infty$
2.	$ln(1-x)=-x-\frac{x^2}{2}-\frac{x^3}{3}-\frac{x^4}{4}-\cdots-\frac{x^n}{n}-\cdots=-\sum_{n=1}^{\infty}\frac{x^n}{n}$, $-1<x\leq 1$
3.	$\frac{1}{1+x}=1-x+x^2-\ldots+(-1)^n x^n+\cdots=\sum_{n=0}^{\infty}(-1)^n x^n$, $-1<x<1$
4.	$arcth(x)=\frac{1}{x}+\frac{1}{3x^3}+\frac{1}{5x^5}+\frac{1}{7x^7}+\cdots$, $\|x\|>1$
6.	$arcsin(x)=x+\frac{1}{2}\cdot\frac{x^3}{3}+\frac{1\cdot 3}{2^2\cdot 2!}\cdot\frac{x^5}{5}+\frac{1\cdot 3\cdot 5}{2^3\cdot 3!}\cdot\frac{x^7}{7}+\cdots$, $-1<x<1$
7.	$\frac{1}{(1-x)^2}=1+2x+3x^2+\ldots+(n-1)x^n+\cdots=\sum_{n=0}^{\infty}(n-1)x^n$, $-1<x<1$
8.	$arth(x)=x+\frac{x^3}{3}+\frac{x^5}{5}+\cdots+\frac{x^{2n-1}}{2n-1}+\cdots$, $-1<x<1$
9.	$\frac{1}{1-x}=1+x+x^2+\ldots+x^n+\cdots=\sum_{n=0}^{\infty}x^n$, $-1<x<1$
10.	$\sqrt{1+x}=1+\frac{1}{2}x-\frac{1}{2\cdot 4}x^2+\frac{1\cdot 3}{2\cdot 4\cdot 6}x^3-\frac{1\cdot 3\cdot 5}{2\cdot 4\cdot 6\cdot 8}x^4+\cdots$, $-1\leq x\leq 1$
11.	$cos(x)=1-\frac{x^2}{2!}+\frac{x^4}{4!}-\frac{x^6}{6!}+\cdots+(-1)^n\frac{x^{2n}}{(2n)!}+\cdots=\sum_{n=0}^{\infty}(-1)^n\frac{x^{2n}}{(2n)!}$, $-\infty<x<+\infty$
12.	$\frac{1}{\sqrt{1+x}}=1-\frac{1}{2}x+\frac{1\cdot 3}{2\cdot 4}x^2-\frac{1\cdot 3\cdot 5}{2\cdot 4\cdot 6}x^3+\frac{1\cdot 3\cdot 5\cdot 7}{2\cdot 4\cdot 6\cdot 8}x^4-\cdots$, $-1<x<1$
13.	$\frac{1}{1+x^2}=1-x^2+x^4-x^6+x^8-\ldots+(-1)^n nx^{n-1}+\cdots=\sum_{n=0}^{\infty}(-1)^n x^{2n}$, $-1<x<1$
14.	$e^x=1+\frac{x}{1!}+\frac{x^2}{2!}+\frac{x^3}{3!}+\cdots+\frac{x^n}{n!}+\cdots=\sum_{n=0}^{\infty}\frac{x^n}{n!}$, $-\infty<x<+\infty$

15.	$$\frac{1}{\sqrt{1-x^2}} = 1 + \frac{1}{1!\cdot 2}x^2 + \frac{1\cdot 3}{2!\cdot 2^2}x^4 + \frac{1\cdot 3\cdot 5}{3!\cdot 2^3}x^6 + \cdots,$$ $$-1 < x < 1$$
16.	$$ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \cdots + (-1)^{n+1}\frac{x^n}{n} + \cdots = \sum_{n=1}^{\infty}(-1)^{n+1}\frac{x^n}{n},$$ $$-1 < x \leq 1$$
17.	$$\frac{1}{(1+x)^2} = 1 - 2x + 3x^2 - 4x^3 + \ \ldots + (-1)^n nx^{n-1} + \cdots = \sum_{n=1}^{\infty}(-1)^{n-1}nx^{n-1},$$ $$-1 < x < 1$$
18.	$$ln\frac{1+x}{1-x} = 2\left(x + \frac{x^3}{3} + \frac{x^5}{5} + \cdots + \frac{x^{2n-1}}{2n-1} + \cdots\right),$$ $$-1 < x < 1$$
19.	$$sin(x) = \frac{x}{1!} - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots + (-1)^n\frac{x^{2n+1}}{(2n+1)!} + \cdots = \sum_{n=0}^{\infty}(-1)^n\frac{x^{2n+1}}{(2n+1)!},$$ $$-\infty < x < +\infty$$
20.	$$ch\,(x) = 1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \frac{x^6}{6!} + \cdots + \frac{x^{2n}}{(2n)!} + \cdots = \sum_{n=0}^{\infty}\frac{x^{2n}}{(2n)!},$$ $$-\infty < x < +\infty$$
21.	$$(1+x)^m = 1 + \frac{m}{1!}x + \frac{m(m-1)}{2!}x^2 + \frac{m(m-1)(m-2)}{3!}x^3 + \cdots + \frac{m(m-1)\ldots(m-n+1)}{n!}x^n + \cdots,$$ $$-1 < x < 1$$
22.	$$arsh(x) = x - \frac{1}{2}\cdot\frac{x^3}{3} + \frac{1\cdot 3}{2\cdot 4}\cdot\frac{x^5}{5} - \frac{1\cdot 3\cdot 5}{2\cdot 4\cdot 6}\cdot\frac{x^7}{7} + \cdots,$$ $$-1 \leq x \leq 1$$
23.	$$(1-x)^m = 1 - \frac{m}{1!}x + \frac{m(m-1)}{2!}x^2 - \frac{m(m-1)(m-2)}{3!}x^3 + \cdots + (-1)^n\frac{m(m-1)\ldots(m-n+1)}{n!}x^n + \cdots,$$ $$-1 < x < 1$$
24.	$$\frac{1}{(1+x)^3} = 1 - \frac{2\cdot 3}{2}x + \frac{3\cdot 4}{2}x^2 - \frac{4\cdot 5}{2}x^3 + \ \ldots + (-1)^n\frac{(n+1)(n+2)}{2}x^n + \cdots = \sum_{n=0}^{\infty}(-1)^n\frac{(n+1)(n+2)}{2}x^n,$$ $$-1 < x < 1$$
25.	$$cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \cdots + (-1)^n\frac{x^{2n}}{(2n)!} + \cdots = \sum_{n=0}^{\infty}(-1)^n\frac{x^{2n}}{(2n)!},$$ $$-\infty < x < +\infty$$
26.	$$ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \cdots + (-1)^{n+1}\frac{x^n}{n} + \cdots = \sum_{n=1}^{\infty}(-1)^{n+1}\frac{x^n}{n},$$ $$-1 < x \leq 1$$

Контрольні питання та завдання

1. Що таке рекурсія?
2. Що таке ітерація?

3. Що таке глибина рекурсії?
4. Що таке базис рекурсії?
5. Що таке складна рекурсія?
6. Чому варто слідкувати за стеком при роботі з рекрсією?

Контрольні питання та завдання підвищеної складності

1. У чому різниця між рекурсією та ітерацією?
2. В чому різниція між постфіксною та префіксною рекурсією?
3. В чому різниція між прямою та непрямою рекурсією?

БІБЛІОГРАФІЧНИЙ СПИСОК

1. Учасники проєктів Вікімедіа. Алгоритм. *Вікіпедія*. URL: https://uk.wikipedia.org/wiki/алгоритм (дата звернення: 10.12.2024).
2. Учасники проєктів Вікімедіа. Програмування. *Вікіпедія*. URL: https://uk.wikipedia.org/wiki/Програмування (дата звернення: 07.11.2024).
3. Етапи розв'язання задачі за допомогою персонального комп'ютера (част. 2). *Українське програмування*. URL: https://programming.in.ua/programming/basisprogramming/135-model-comp-2.html (дата звернення: 10.12.2024).
4. Мова програмування С++. *Основи програмування на С / С++ (Сі++) для початківців і новачків.* URL: http://cppstudio.com/cat/274/ (дата звернення: 04.11.2024).
5. Мова С++. *Бібліотека програміста*. URL: https://proglib.io/ (дата звернення: 10.12.2024).
6. Основи програмування на С++. *PureCodeCpp*. URL: https://purecodecpp.com/uk/ (дата звернення: 04.11.2024).
7. Microsoft Technical Documentation. *Microsoft Learn*. URL: https://learn.microsoft.com/uk-ua/docs/ (date of access: 04.11.2024).
8. Stroustrup B. A Tour of C++. 3rd ed. Boston : Addison-Wesley Professional, 2022. 299 p.
9. Stroustrup B. The C++ Programming Language. 4th ed. Boston : Addison-Wesley, 2013. 1376 p.
10. Lippman S., Lajoie J., Moo B. C++ Primer. 5th ed. Boston : Addison-Wesley Professional, 2012. 976 p.
11. Hunniger D. C++ Programming. *Wikibooks – Free Manuals*. URL: http://wikibooks.org (date of access: 10.12.2024).
12. Participants in Wikimedia projects. Nassi–Shneiderman diagram. *Wikipedia*. URL: https://en.wikipedia.org/wiki/Nassi–Shneiderman_diagram (date of access: 04.11.2024).
13. Lafore R. Object-Oriented Programming in C++. Subsequent ed. Indianapolis : SAMS Publishing, 2001. 1012 p.
14. Prata S. C++ Primer Plus. 6th ed. Boston : Addison-Wesley Professional, 2011. 1420 p.
15. Participants in Wikimedia projects. Visual Studio. *Wikipedia*. URL: https://en.wikipedia.org/wiki/Visual_Studio (date of access: 04.11.2024).

ДОДАТОК А.

Приклад розробки програми

1. Постановка завдання

Дано дійсне число *a*. Для функції *f(x)*, графік якої наведено на рис.1, розробити програму обчислення значення *f(a)*.

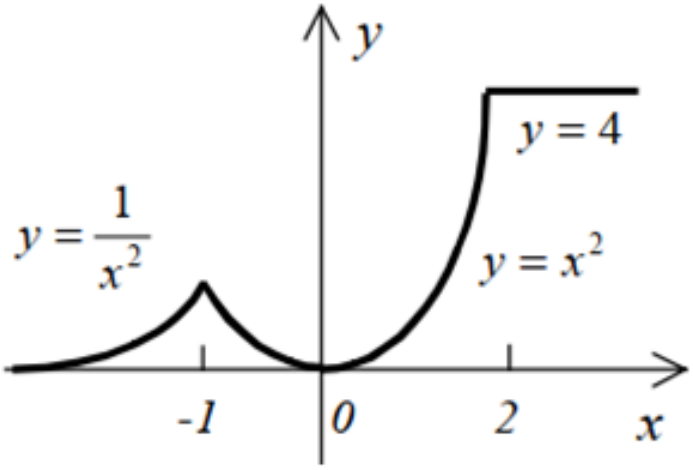


Рис.1. Графік функції *f(x)*

Вимоги до програми:
– вхідні дані вводяться з клавіатури;
– результати роботи програми виводяться на екран.
Вимоги до тексту програми:
– коментарі щодо призначення програми, її вхідних і вихідних даних;
– коментарі щодо призначення кожного блоку програми;
– самодокументованість коду: всі ідентифікатори повинні мати назви, що відповідають суті змінних.

2. Зовнішні специфікації

2.1. Формат вхідних даних
Вхідні дані: аргумент функції.
Вимоги до формату вхідних даних наведені у табл.1.

Таблиця 1

Формат вхідних даних

№ з/п	Найменування даних	Умовне позначення	Вимоги до даних	Приклад
1	Аргумент функції	a	Дійсне число	8 або -3,96

2.2. Формат вихідних даних
Вихідні дані: значення функції.
Вимоги до формату вихідних даних наведені у табл.2.

Таблиця 2

Формат вихідних даних

№	Найменування даних	Умовне позначення	Вимоги до даних	Приклад
1	Значення функції	f	Дійсне додатне число або нуль	-18 або 33,5

2.3. Функціональні вимоги до програми

Програма повинна реалізовувати такі дії:

– забезпечення вводу користувачем вхідних даних для обчислення;
– обчислення значення функції;
– вивід результатів обчислення.
–

3. Метод рішення задачі

Для обчислення значення функції скористаємося формулою

$$f(x) = \begin{cases} \frac{1}{x^2}, & x \le -1 \\ x^2, & -1 < x \le 2 \\ 4, & x > 2 \end{cases}$$

4. Тести для перевірки рішення завдання

Таблиця 3

Набор тестів

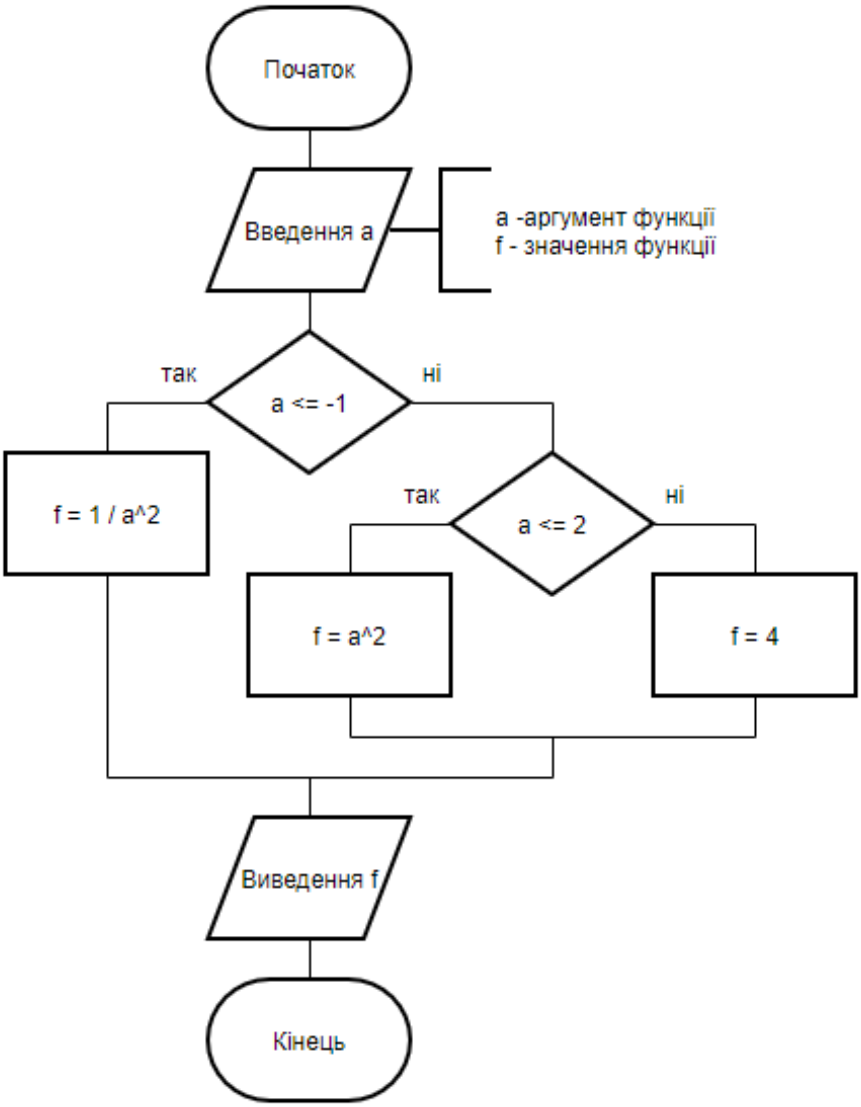
№	Назва	Вхідні дані	Очікувані результати
1	Функція f(x) = 1/x^2	-2	0.25
2	Граничне значення -1	-1	1
3	Функція f(x) = x^2	1	1
4	Граничне значення 2	2	4
5	Функція f(x) = 4	3	4

5. Алгоритм рішення завдання

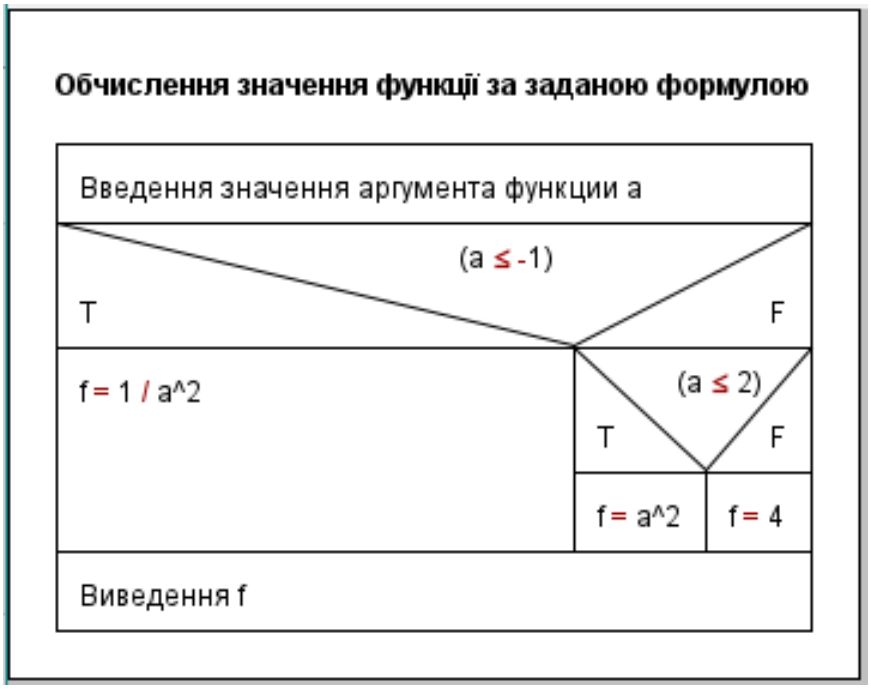
1.1. Словесний алгоритм:

1) Ввести значення аргументу функції *a*.
2) Якщо *a* ≤ *-1*, обчислити значення функції за формулою *f = 1 / a^2* і перейти до п.5
3) Якщо *a* ≤ *2*, обчислити значення функції за формулою *f = a^2* і перейти до п.5
4) Присвоїти f значення 4.
5) Вивести значення f на екран.

1.2. Блок-схема



1.3.Діаграма N-S

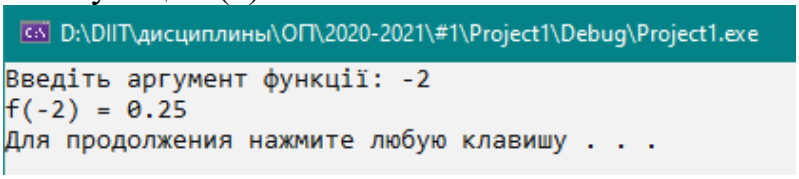


6. Текст програми

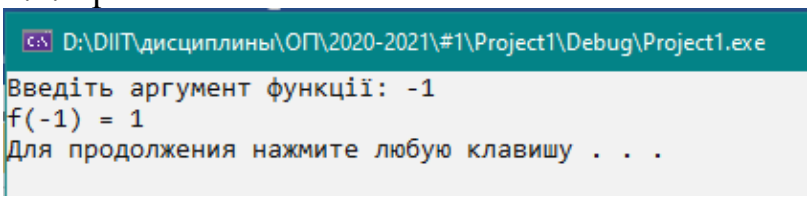
```
// програма обчислення значення функції
// вхідні дані: аргумент функції - дійсне число
// вихідні дані: значення функції - дійсне число
#include <iostream>       // директива препроцесора для підключення засобів
                          // введення/виведення
#include <Windows.h>  // директива препроцесора для підтримки кирилиці в консолі

using namespace std;      // використовувати простір імен std
                                        // (стандартної бібліотеки шаблонів)
int main()    // заголовок головної функції програми
{
        // підтримка кирилиці в консолі Windows
        SetConsoleCP(1251);
        SetConsoleOutputCP(1251);
        // введення даних
        float a;        // оголошення змінної для зберігання значення аргументу функції
        cout << "Введіть аргумент функції: ";
        cin >> a;
        // обчислення значення функції
        double f;      // оголошення змінної для зберігання значення функції
        if (a <= -1)
                f = 1. / (a * a);
        else
                if (a <= 2)
                        f = a * a;
                else
                        f = 4;
        // виведення результату
        cout << "f(" << a << ") = " << f << endl;
        system("pause"); // затримка для перегляду результату
        return 0; // вихід з функції main()
}
```

7. Результати тестування програми

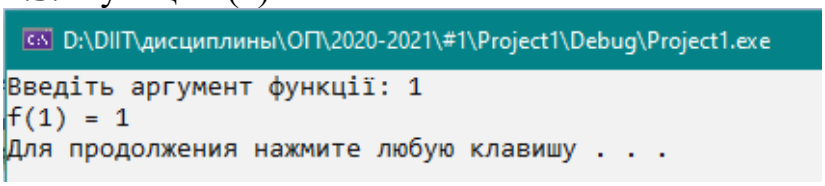
1.1.Функція f(x) = 1/x^2



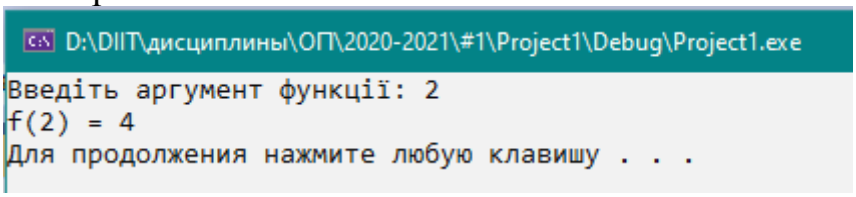
1.2. Граничне значення -1



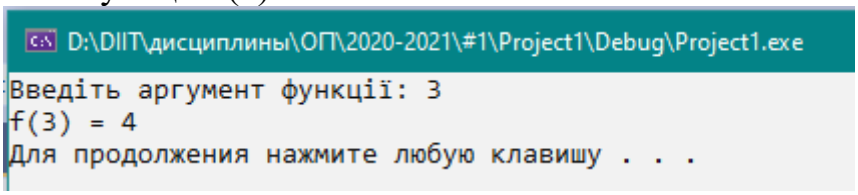
1.3. Функція f(x) = x^2



1.4. Граничне значення 2



1.5. Функція f(x) = 4



8. Аналіз результатів тестування програми

Всі фактичні результати виконання програми співпадають з очікуваними результатами з табл.3. В табл.3 наведені тести для всіх частин функції, а також для граничних значень аргументів функції – точок, де стикаються дві частини графіка функції. В алгоритмі й програмі не передбачена реакція на введення некоректних даних, тому тестування вхідних даних на «захист від дурня» не проводилось.

Навчально-методичне видання

Горбова Олександра Вікторівна

ОСНОВИ ПРОГРАМУВАННЯ

Навчально-методичні рекомендації до виконання лабораторних робіт

Електронне видання

Експертний висновок склав канд. техн. наук, доц. Вадим Андрющенко

Зареєстровано НМВ УДУНТ (№ 808 від 24.12.2024)

В авторській редакції
Комп’ютерна верстка О. В. Горбова

Формат 60х84 1/16. Ум. друк. арк. 3.95. Обл.-вид. арк. 4,0.

Зам. № 2

Видавець: Український державний університет науки і технологій
вул. Лазаряна, 2, ауд. 2216, м. Дніпро, 49010.
Свідоцтво суб’єкта видавничої справи ДК № 7709 від 14.12.2022