МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Миколаївський національний університет імені В.О.Сухомлинського
Коледж МНУ імені В.О.Сухомлинського
Циклова комісія технічного напряму підготовки

МЕТОДИЧНІ ВКАЗІВКИ ДО ЛАБОРАТОРНИХ ЗАНЯТЬ З ДИСЦИПЛІНИ «АЛГОРИТМИ ТА СТРУКТУРИ ДАНИХ»

для студентів галузі знань 11 Математика та статистика

спеціальність 113 Прикладна математика

м. Миколаїв -2019

Укладач:
викладач вищої категорії циклової комісії технічного напряму підготовки Литвинчук О.В.

Методичні вказівки розглянуті і затверджені на засіданні циклової комісії технічного напряму підготовки, протокол №_6_ від «_12_»_листопада__20_19_р.

Методичні вказівки до лабораторних занять з дисципліни «Алгоритми та структури даних» для студентів ІІ курсу галузі знань 11 Математика та статистика спеціальності 113 Прикладна математика розроблені на основі навчальної програми для вищих навчальних закладів І-ІІ рівнів акредитації, які здійснюють підготовку молодших спеціалістів на основі базової загальної середньої освіти.

Зміст

Вступ 4
Порядок виконання лабораторних робіт 5
Вимоги до оформлення звітів про виконання лабораторних робіт 5
Лабораторна робота № 1. Побудова лінійних алгоритмів 6
Лабораторна робота № 2. Побудова розгалужених алгоритмів 10
Лабораторна робота № 3. Побудова циклічних алгоритмів 14
Лабораторна робота № 4. Аналіз методів сортування 18
Лабораторна робота №5. Проектування алгоритмів обробки одновимірних масивів 22
Лабораторна робота №6. Проектування алгоритмів обробки двовимірних масивів 30
Лабораторна робота №7. Проектування алгоритмів з використанням підпрограм 33
Лабораторна робота № 8. Алгоритми пошуку 39
Лабораторна робота № 9. Особливості використання рядків, множин та структур 43
Рекомендована література 50

Вступ

Дисципліна «Алгоритми та структури даних» є базовою дисципліною професійного циклу підготовки спеціалістів з комп'ютерних наук.

Метою вивчення даної дисципліни є формування у студентів знань, умінь і навичок в області методів представлення даних в пам'яті комп'ютера та основних алгоритмів їх опрацювання.

В результаті вивчення дисципліни повинні бути сформовані професійні компетенції:

- знання сучасних інструментаріїв ПК;
- знання теоретичних основ алгоритмізації і проектування програм;
- знання етапів обробки програм на комп'ютері, скалярні та структурні типи даних;
- знання методів структурного програмування;
- знання засобів використання статичних та динамічних структур даних;
- знання методів побудови та використання складних структур даних, нетрадиційні представлення даних;
- знання різних аспектів обробки цих структур даних;
- знання про складність алгоритмів і методах аналізу складності та ефективності;
- знання про способи подання стеків, черг, дерев у пам'яті ЕОМ;
- знання методів сортування (внутрішньої і зовнішньої);
- знання завдань пошуку і методів їх вирішення:
- знання різних методів розробки алгоритмів.
- вміння розв'язувати спектр проблем при рішенні практичних задач від проблеми формалізації задачі до проблем, які постають у час виконання закінченої програми;
- вміння застосовувати вибраний або розроблений алгоритм до конкретних вихідних даних задачі, яка вирішується.
- вміння використовувати рекурсивні структури даних, рекурсивні алгоритми;
- вміння при вирішенні конкретної задачі грамотно формулювати завдання програмування;
- вміння застосовувати методи побудови нових типів при проектуванні інформаційних моделей;
- вміння вибирати оптимальну для даної інформаційної моделі структуру даних;
- вміння для написання програми вміти вибирати у разі потреби одне з відомих рішень;
- вміння аналізувати трудомісткість методу сортування даних;

вміння вибрати оптимальний підхід для вирішення завдання.

Поняття алгоритму невід'ємно пов'язане із структурою даних, обраною для практичної реалізації поставленої задачі. Алгоритми опрацювання даних в комп'ютерних науках використовуються для опису методів розв'язання задачі, які можна реалізувати в середовищі програмування. Ретельний підхід до вивчення класичних алгоритмів та пошуки шляхів розробки нових алгоритмів є фундаментом ефективного розв'язання поставленої задачі за допомогою комп'ютера майбутніми фахівцями з програмування.

В даних методичних вказівках представлено 9 лабораторних робіт, кожна з яких містить необхідні короткі теоретичні відомості, приклад розв'язання типового завдання та зразки завдань для самостійного виконання, контрольні питання. В методичних вказівках також наведена рекомендована література.

Порядок виконання лабораторних робіт

Для виконання лабораторних робіт необхідно:

- використовуючи літературні джерела, конспект лекцій, методичні розробки з дисципліни, засвоїти теоретичний матеріал, пов'язаний з тематикою лабораторної роботи;
- відповісти керівнику лабораторних занять на поставлені запитання за темою лабораторної роботи;
- опрацювати наведений у лабораторній роботі приклад;
- отримати індивідуальне завдання;
- розробити схему алгоритму для розв'язування індивідуального завдання;
- за потреби написати відповідну програмну реалізацію;
- оформити та захистити звіт про виконання лабораторної роботи.

Вимоги до оформлення звітів про виконання лабораторних робіт

Звіти про виконання лабораторних робіт оформляються на аркушах формату А4 та подаються до захисту в електронному вигляді.

Структура звіту:

- титульний аркуш (зразок титульного аркуша наведено в додатку А);
- номер та тема роботи;
- мета виконання лабораторної роботи;
- індивідуальне завдання з детальним формулюванням задачі;
- графічна схема алгоритму розв'язування задачі з поясненням (за потреби);
- базові елементи тексту програми.

Звіт повинен бути написаний українською мовою, акуратно та грамотно, з дотриманням правил оформлення ділової документації. Назви розділів звіту повинні бути візуально виділені розміром, жирністю, курсивом шрифта або підкресленням. Лабораторна робота повинна бути здана у визначені викладачем терміни.

Лабораторна робота № 1. Побудова лінійних алгоритмів

Мета: навчитися складати лінійні алгоритми для розв'язку задач.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Алгоритм зветься *лінійним,* якщо всі його дії виконуються послідовно, одна за одною, від початку до кінця. У лінійному алгоритмі не може бути команди, яка б передбачала різну послідовність виконання алгоритму.

При побудові структурних схем (блок-схем) лінійних алгоритмів використовуються наступні блоки:

Рисунок 1 - Блоки, які використовуються при побудові схем лінійних алгоритмів.

Символ "Термінатор" починає і завершує схему алгоритму. В ньому пишеться відповідне слово: "ПОЧАТОК" або "КІНЕЦЬ". Від блоку "ПОЧАТОК" відходить тільки одна лінія.

Усередині символу "Ввод/вывод" записуються значення, які вводяться у програму або виводяться з неї з відповідним словом "ВВОД" або "ВИВОД". Усередині символу "Процес" записуються текстові вказівки або формули. З такого блоку може входити тільки одна лінія.

За допомогою ліній, якими з'єднуються блоки, позначається послідовність виконання алгоритму. Після виконання дій одного блоку переходять по лінії до виконання дій наступного блоку.

За основний напрямок виконання дій прийнято напрямок зверху вниз і зліва направо. В цьому випадку стрілки на кінцях ліній можна не ставити.

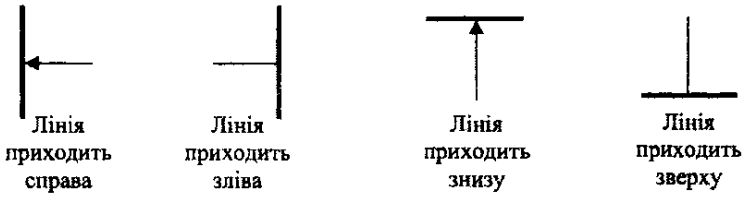


Рисунок 2 - Основні і побічні напрямки ліній з'єднань.

МЕТОДИЧНІ ВКАЗІВКИ

Завданням даної лабораторної роботи є складання лінійного алгоритму для розв'язку певної (арифметичної, геометричної, логічної) задачі. Результатом виконання роботи повинно бути зображення схеми алгоритму, виконане на папері або з використанням ЕОМ у будь-якому графічному редакторі (наприклад, Paint).

Лабораторна робота виконується у наступній послідовності.

1. Постановка задачі.

Постановка задачі - це чітке формулювання задачі, визначення вхідних даних для її розв'язування і точні вказівки відносно того, які результати і в якому вигляді повинні бути отримані.

Первинна постановка задачі - це завдання на лабораторну роботу. Однак перед виконанням наступних кроків роботи необхідно:

- уважно перечитати задачу (можливо декілька разів) до чіткого розуміння ЇЇ суті і вимог;

- визначити, які дані є вхідними, тобто такими, які задаються користувачем;
- визначити, які дані є вихідними, тобто такими, які треба отримати в результаті розв'язку задачі.

2. Математична формалізація задачі.

Математична формалізація задачі - це опис задачі у вигляді формул, рівнянь, співвідношень, обмежень.

Цей крок є найважливішим при виконанні даної роботи. Більша частина задач у даній та наступних лабораторних роботах потребують математичної формалізації.

Математична формалізація задачі вимагає від студента певного рівня знань, вмінь та навичок в області, до якої належить поставлена задача.

3. Вибір методу розв'язування задачі.

Вибір методу розв'язування задачі полягає у виборі того чи іншого способу розв'язування задачі. Вибір методу залежить як від самої задачі, так і від можливостей комп'ютера. При оцінюванні якості розв'язку задачі враховуються наступні показники:

- оригінальність розв'язку;
- об'єм пам'яті, який займає і використовує програма;
- трудомісткість обчислень, тобто ефективність алгоритму;
- лаконічність і наочність програми.

4. Побудова схеми алгоритму.

Побудова схеми алгоритму - це графічний запис алгоритму на основі вибраного методу.

При побудові алгоритму дотримуються правил, викладених у теоретичних відомостях до даної лабораторної роботи. Перед формуванням кінцевого варіанту схеми бажано розглянути і проаналізувати кілька варіантів схеми. Алгоритм більшою мірою визначається методом, хоча один і той же метод може бути реалізований за допомогою різних алгоритмів.

5. Перевірка алгоритму.

Перевірка алгоритму полягає в ручній перевірці окремих розв'язків задачі. Отриманий алгоритм необхідно обов'язково перевірити за допомогою *тестів.* Тест - сукупність вхідних даних для алгоритму з очікуваними результатами. Звичайно треба приготувати не один а декілька тестів, який допоможе охопити максимум ситуацій.

Набір тестів називається *повним,* якщо він дозволяє активізувати усі гілки алгоритму. В наборі тестів виділяють три групи:

- "тепличні" - перевіряють роботу алгоритму при коректних, нормальних вхідних даних найпростішого вигляду;
- "екстремальні" - на межі області визначення, в ситуаціях, які можуть відбутися і на які треба коректно реагувати;
- "позамежні" - за межами області визначення - ситуації, безглузді з точки зору постановки задачі, але які можуть відбутися через помилки користувача або алгоритмів, що надають вхідні дані для алгоритму, який тестується.

Слід відмітити, що лінійні алгоритми мають дуже обмежені можливості щодо контролю коректності даних через те, що в них не можна використовувати блоки розгалуження.

Приклад. Уряд гарантує, що інфляція в новому році складатиме p % на місяць. Якого зростання цін за рік можна очікувати?

Розв'язок.

Вхідними даними в цій задачі є рівень інфляції, що задається у процентах, та інтервал часу, зростання цін на протязі якого треба підрахувати.

Вихідні дані - коефіцієнт зростання цін - будемо обчислювати як відношення ціни будь-якого товару в кінці року до ціни цього товару на початку року.

Проведемо математичну формалізацію задачі. Позначимо ціну деякого товару на початку року - c_1 ціну того ж товару в кінці року - c_{12}. Тоді ціна товару в кінці року

обчислюється наступним чином:

$$c_{12}=c_1 \times k \times 12,$$

де *k* - коефіцієнт збільшення ціни за 1 місяць.

Коефіцієнт збільшення ціни за 1 місяць звідси буде:

$$k= c_{12} / (c_1 \times 12),$$

а за 1 рік:

$$k= c_{12} / c_1 .$$

Якщо за 1 місяць ціна збільшується на р процентів, це значить, що до початкової ціни додається *p* її сотих частин, тобто:

$c_2= c_{1+} c_{1*}p/100= c_1*(1+p/100)$,

$c_3= c_{2+} c_{2*}p/100= c_2*(1+p/100)$,

…

$c_{12}= c_{11+} c_{11*}p/100= c_{11}*(1+p/100)/$

Звідси легко побачити:

$c_3= c_2*(1+p/100)= c_1*(1+p/100)^2$,

$c_4= c_3*(1+p/100)= c_2*(1+p/100)^2= c_1*(1+p/100)^3$,

…

$c_{12}= c_{11}*(1+p/100)=…= c_1*(1+p/100)^{12}$

Таким чином, коефіцієнт збільшення цін за рік складатиме:

$k=c_{12}/c_1=c_1*(1+p/100)12/c1=(1+p/100)^{12}$

Алгоритм розв'язку спочатку запишемо у вигляді інструкцій:

1. Ввести *p* - рівень інфляції за місяць у процентах.
2. Обчислити $k=(1+p/100)^{12}$ – коефіцієнт зростання цін за рік.
3. Вивести *k.*

Схема алгоритму представлена на наступному рисунку.

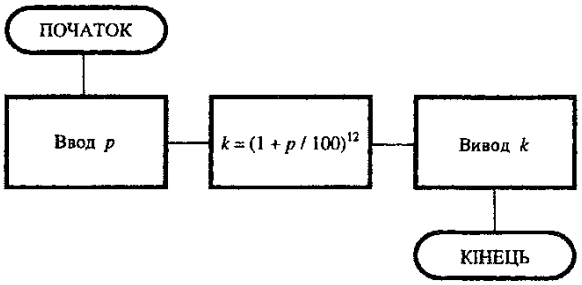


Рисунок 3 - Схема алгоритму розв'язку задачі про зростання цін.

Роботу алгоритму перевіримо на наступних контрольних значеннях рівня інфляції:

1.p=50%, $k=(1+50/100)^{12}\approx130$ – ціни зростуть у 130 разів.

2.p=0, $k=(1+0/100)^{12}\approx0$ – ціни не зростуть.

ЗАВДАННЯ

1. Вивчити теоретичні відомості і методичні вказівки до лабораторної роботи.
2. Вибрати завдання на лабораторну роботу згідно варіанту.
3. Розробити алгоритм розв'язку у текстовому вигляді.
4. Скласти схему алгоритму розв'язку задачі.
5. Розробити систему контрольних прикладів і перевірити роботу алгоритму,
6. Зробити висновки.

ВАРІАНТИ ЗАВДАНЬ

1. Кут α заданий в градусах, хвилинах та секундах. Знайти його величину в радіанах.
2. Довжина відрізку задана в дюймах (1 дюйм = 2,54 см). Перевести значення довжини в метричну систему, тобто виразити її в метрах, сантиметрах і міліметрах.
3. Задані моменти початку і кінця деякого проміжку часу в годинах, хвилинах і секундах (в межах доби). Знайти тривалість цього проміжку в кожній з перерахованих одиниць.
4. В таксі одночасно сіли три пасажири. Коли вийшов перший пасажир, на лічильнику було p_1 гривень; коли вийшов другий - p_2 гривень. Скільки повинен був заплатити кожен пасажир, якщо в кінці поїздки лічильник показав p_3 гривень? Плата за посадку складає p_0 гривень.
5. За перший рік продуктивність праці на підприємстві зросла на p_1 %, за другий і третій - відповідно на p_2 і p_3 %. Знайти середньорічний приріст продуктивності (в процентах).
6. Знайти корені квадратного рівняння, заданого своїми коефіцієнтами, з додатнім дискримінантом.
7. Задані координати точки підвіски математичного маятника A(x_0, y_0, z_0) і координати одної з точок його найвищого положення B(x_1, y_1, z_1). Знайти координати найнижчої точки траєкторії та іншої найвищої точки підйому.
8. Задані рівняння двох прямих на площині, які перетинаються: $y = k_1 x + b_1$ та $y = k_2 x + b_2$. Знайти кут між ними.
9. У квадрата *ABCD* на площині відомі координати двох протилежних вершин - точок *A* і *C*. Знайти координати точок *B* і *D*. Зауваження: розташування квадрату є довільним.
10. В рівнобедреному прямокутному трикутнику відома висота *h*, опущена на гіпотенузу. Знайти сторони трикутника.
11. Трикутник *ABC* заданий довжинами своїх сторін. Знайти довжину висоти, опущеної з вершини A.
12. З кола радіуса *r* вирізаний прямокутник, велика сторона якого дорівнює *a*. Знайти максимальний радіус кола, яке можна вирізати з отриманого прямокутника.
13. Трикутник *ABC* заданий координатами своїх вершин на площині. Знайти площу трикутника.
14. Знайти суму довжин медіан трикутника *ABC*, який заданий координатами своїх вершин на площині.
15. Тваринник на початку кожної зими підвищує відпускну ціну на молоко на *p* %, а кожного літа - знижує на стільки ж відсотків. Чи зміниться ціна на молоко і якщо так, то в яку сторону і на скільки через *n* років.
16. Знайти внутрішні кути трикутника *ABC*, який заданий координатами своїх вершин на площині.
17. Комерсант, який мав стартовий капітал *k* гривень, зайнявся торгівлею, яка щомісяця збільшує капітал на *p* %. Через скільки років він накопить суму *s*, достатню для купівлі власного магазину?
18. Знайти координати вершини параболи $y = ax^2 + bx + c$
19. Парабола $y = ax^2 + bx + c$ перетинається з прямою $y = px + q$ в одній точці. Знайти координати цієї точки.
20. Написати алгоритм побудови бісектриси кута за допомогою циркуля та лінійки.

КОНТРОЛЬНІ ПИТАННЯ

1. Дайте визначення поняття алгоритму і його основних властивостей.
2. Які ви знаєте етапи розв’язання задачі на ЕОМ?
3. Які основні типи задач вивчаються в курсі «Алгоритми та структури даних»?
4. Що означає скінченність (дискретність) алгоритму?
5. З чого складаються прості (лінійні) алгоритми?

Лабораторна робота № 2. Побудова розгалужених алгоритмів

Мета: навчитися складати розгалужені алгоритми для розв'язку задач.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Розгалужений алгоритм - це алгоритм, в якому виконуються ті або інші дії залежно від результату перевірки *умови.*

Умова - це логічний вираз, який може приймати два значення: "так" - якщо умова виконується і "ні" - якщо умова не виконується. Умови можуть мати вигляд як математичних співвідношень, так і текстових виразів і запитань.

При побудові структурних схем розгалужених алгоритмів використовується блок "Розгалуження" у різних модифікаціях:

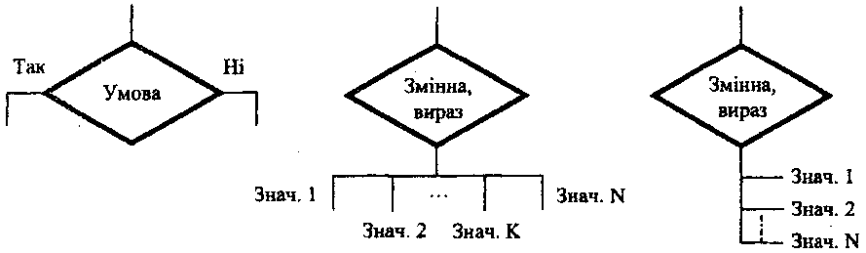


Рисунок 1 - Модифікації розгалуженого блоку.

МЕТОДИЧНІ ВКАЗІВКИ

Завданням даної лабораторної роботи є складання розгалуженого алгоритму для розв'язку певної задачі. Результатом виконання роботи повинне бути зображення схеми алгоритму, виконане на папері або з використанням ЕОМ у будь-якому графічному редакторі (наприклад, Paint).

При виконанні даної роботи рекомендується використовувати також методичні вказівки до лабораторної роботи № 1.

При побудові алгоритмів, складні умови рекомендується розбивати на більш прості. Наступний приклад ілюструє це.

Приклад 1. Вивести значення складної функції за заданим користувачем значенням аргументу:

$$y = \begin{cases} x^2, x \le 0 \\ x^3, 0 < x \le 1 \\ 4x, x > 1 \end{cases}.$$

Найпростіший розв'язок цієї задачі полягає у послідовній перевірці усіх трьох умов (маються на увазі умови належності аргументу x певному діапазону). Схема алгоритму, що відповідає такому розв'язку представлена на наступному рисунку.

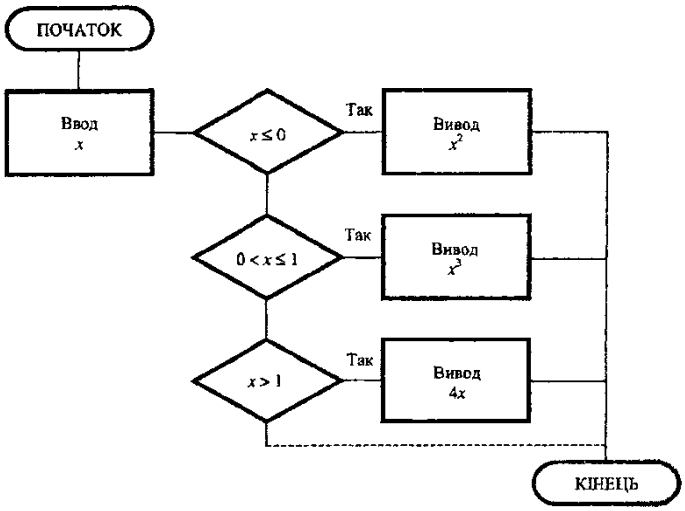


Рисунок 2 - Схема алгоритму розв'язку задачі про складну функцію.

Однак ще до завершення малювання такої схеми ми побачимо, що вона є некоректною, тому що гілка "Ні" блоку розгалуження "x > 1" не має сенсу. Дійсно, з нашої схеми випливає, що може виникнути ситуація, коли жодна умова не виконується і тоді дії алгоритму є невизначеними (позначено штриховою лінією на схемі). Чим викликана ця помилка? Усі три умови тут ($x \le 0$, $0 < x \le 1$, $x > 1$) є взаємозалежними. Пояснимо це на простому логічному міркуванні: нехай аргумент *не* менше і *не* дорівнює нулю. Це значить, що перша умова не виконується, але з цього також випливає, що виконується ліва частина другої умови ($0 < x$) і перевіряти її зайвий раз немає сенсу. Залишається перевірити праву частину другої умови ($x \le 1$). Припустимо, що не виконується і вона, тобто аргумент *не* менше і *не* дорівнює одиниці. Тоді з цього однозначно випливає, що $x > 1$ — виконується остання умова і її також немає сенсу перевіряти.

З проведених міркувань ми бачимо, що замість трьох перевірок можна цілком обійтися двома, позбавляючись при цьому некоректності алгоритму.

Далі приведена схема алгоритму, який використовує два блоки розгалуження замість трьох.

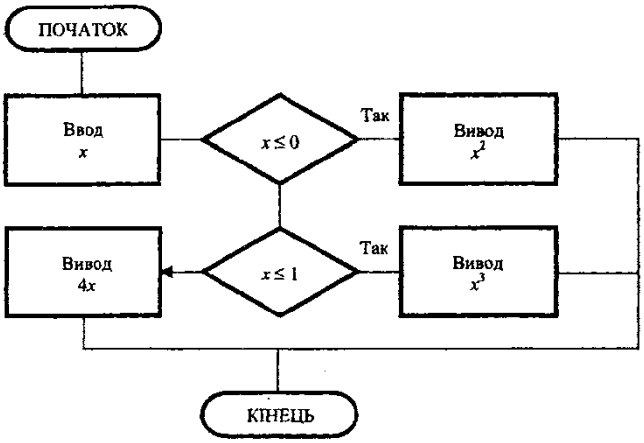


Рисунок 3 - Схема алгоритму розв'язку задачі про складну функцію.

Приклад 2. Відомо, що число ділиться на 3 тоді і тільки тоді, коли сума його цифр ділиться на 3. Перевірити цей признак на прикладі заданого трьохзначного числа.

Розв'язок.

Вхідними даними у цій задачі є цифри трьохзначного числа: *a, b, c.* Для простоти будемо вважати, що ці цифри вводяться кожна окремо, тобто немає необхідності виділяти їх з трьохзначного числа.

Вихідними даними є повідомлення про вірність або невірність признаку.

Перевірка признаку полягає у складанні цифр числа, діленні їх на 3 і перевірці

наявності дробової частини у частки. Якщо дробова частина присутня, сума не ділиться на 3 націло і, отже, признак є невірним. У протилежному випадку признак є вірним.

Запишемо алгоритм у вигляді набору інструкцій:

1. Ввести *a, b, c* - цифри довільного трьохзначного числа.
2. Порахувати $p = (a + b + c) / 3$ — частку від ділення суми цифр на 3.
3. Якщо $\{p\}>0$ — дробова частина частки більше нуля - вивести повідомлення "Невірно"; у протилежному випадку (дробова частина - нуль) вивести "Вірно".

Схема алгоритму представлена на наступному рисунку.

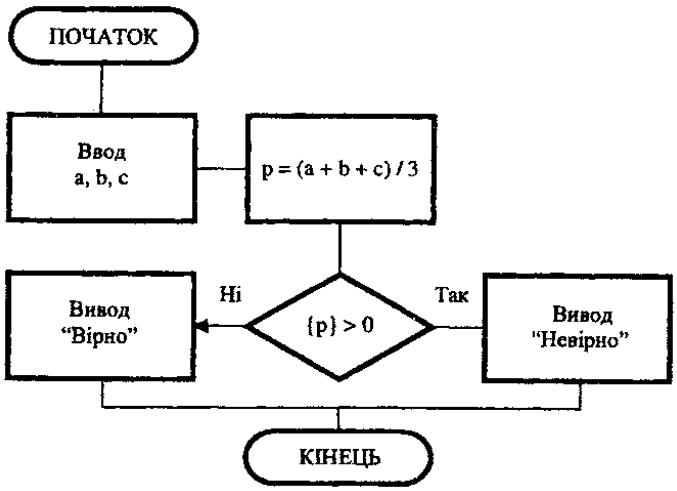


Рисунок 4 - Схема алгоритму розв'язку задачі про ділення на 3.

ЗАВДАННЯ

1. Вивчити теоретичні відомості і методичні вказівки до лабораторної роботи.
2. Вибрати завдання на лабораторну роботу згідно варіанту.
3. Розробити алгоритм розв'язку у текстовому вигляді.
4. Скласти схему алгоритму розв'язку задачі.
5. Розробити систему контрольних прикладів і перевірити роботу алгоритму.
6. Зробити висновки.

ВАРІАНТИ ЗАВДАНЬ

1. Задані три числа: *a, b* і *c*. Визначити, чи можуть вони бути сторонами трикутника, і якщо так, то визначити його тип: рівнобічний, рівнобедрений.
2. Чотирикутник *ABCD* заданий координатами своїх вершин на площині: $A(x_a , y_a)$, $B(x_b , y_b)$, $C(x_c , y_c)$, $D(x_d , y_d)$. Визначити тип чотирикутника: прямокутник, паралелограм, трапеція, довільний чотирикутник.
3. Чи лежить трикутник *ABC*, заданий координатами своїх вершин на площині в області перетину заданих кіл: $(x-a_1)^2+(y-b_1)^2=r_1^2$; $(x-a_2)^2+(y-b_2)^2=r_2^2$?
4. Чи пролізе цегла зі сторонами *a, b* і *c* через прямокутний отвір зі сторонами r і s, якщо сторони отвору повинні бути паралельними граням цегли?
5. Чи може куля з радіусом r пролізти через ромбовидний отвір з діагоналями
6. Чи можна коробку розміром $a \times b \times c$ запакувати в посилку розміром $r \times s \times t$, якщо кутом пакувати не можна?
7. Перевірити, чи лежить коло $(x-a_1)^2+(y-b_1)^2=r_1^2$ цілком у колі $(x-a_2)^2+(y-b_2)^2=r_2^2$ або навпаки.
8. Чи лежить точка $M(x_m, y_m)$ всередині трикутника, заданого координатами своїх вершин $A(x_a ,y_a)$, $B(x_b , y_b)$, $C(x_c , y_c)$ на площині?

9. Серед заданих цілих чисел *k, l, m* знайти всі пари кратних.

10. Задані координати вершин трикутника *ABC* на площині. Вивести їх в порядку обходу за годинниковою стрілкою.

11. Мандрівник рухався t_1 годин зі швидкістю v_1 потім t_2 годин - зі швидкістю v_2 і t_3 годин - зі швидкістю v_3. За який час він подолав першу половину шляху?

12. Можна їхати на таксі зі швидкістю v_1 км/год і оплатою p_1 грн/км або йти зі швидкістю v_2 безкоштовно. Як з найменшими витратами подолати шлях *s* за час *t*, якщо це є можливим? Які це витрати?

13. Раціон корови на добу складає *u* кг сіна, *v* кг силосу і *w* кг комбікорму. В господарстві, яке має стадо з *k* корів, залишилося *s* центнерів сіна, *t* тон силосу і *f* мішків комбікорму по 50 кг. Скільки ще днів господарство може годувати корів за повним раціоном? Який з кормів закінчиться найпершим?

14. На шаховій дошці стоять чорний король і три білі тури. Перевірити, чи не знаходиться король під биттям, а якщо є загроза, то від кого саме.

15. На шаховій дошці стоять чорний король і білі тура та слон. Перевірити, чи є загроза королю і якщо є, то від кого саме.

16. На шаховій дошці стоять три ферзі. Знайти ті пари з них, які загрожують один одному.

17. Банк пропонує три види термінових внесків: на 3 місяці під p_1 %, на 6 місяців під p_2 % і на рік під p_3 %. Який з внесків найбільш прибутковий для вкладника?

18. З пункту *A* в пункт *B* виїхав велосипедист зі швидкістю v_1 км/год. Одночасно назустріч йому з пункту *B* виїхав інший велосипедист зі швидкістю v_2 км/год. Знайти, через який час відбудеться їх зустріч, якщо відстань від пункту А до пункту *B* дорівнює *s* км.

19. Для заданого значення *x* вивести значення функції $y=\ln x+(x-5)^{-1}$

20. Чи можна з круглої заготовки радіуса *r* вирізати дві прямокутні пластини з розмірами $a \times b$ і $c \times d$?

КОНТРОЛЬНІ ПИТАННЯ

1. Які способи опису алгоритмів Ви знаєте?
2. Що таке формальність алгоритму?
3. Що означає масовість алгоритму?
4. Які є три головні базові структури?
5. Яке призначення команди розгалуження і як вона діє ?

Лабораторна робота № 3. Побудова циклічних алгоритмів

Мета: навчитися складати циклічні алгоритми для розв'язку задач.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Алгоритм називається *циклічним,* якщо одна і та ж послідовність дій в ньому виконується кілька разів, доки не виконається задана умова.

Організація циклу вимагає встановлення початкових значень тим змінним, які використовуються в циклі.

Команда або група команд, виконання яких повторюється при кожному проходженні циклу, називається *тілом циклу.*

Цикли бувають двох видів: цикл "ДО" (з післяумовою) і цикл "ПОКИ" (з передумовою). Особливість циклу "ДО" полягає в тому, що він виконується хоча б один раз.

При логічних помилках в циклічних алгоритмах може виникнути ситуація, коли тіло циклу виконується раз за разом, а умова припинення циклу не настає. Така ситуація називається "зациклювання". Тому, при написанні програм за циклічними алгоритмами треба бути особливо уважним.

Циклічні алгоритми можна складати на основі як блоку "Розгалуження" так і блоку "Цикл":

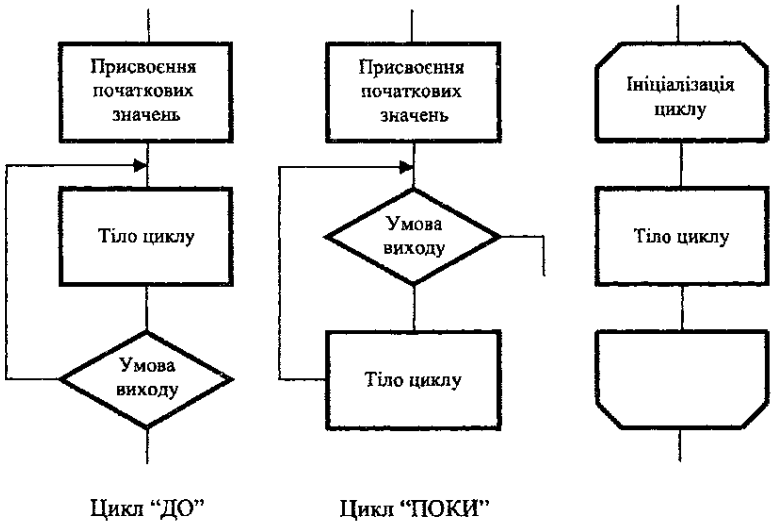


Рисунок 1 - Схеми циклів "ДО" і "ПОКИ".

МЕТОДИЧНІ ВКАЗІВКИ

Завданням даної лабораторної роботи є складання циклічного алгоритму для розв'язку певної задачі. Результатом виконання роботи повинне бути зображення схеми алгоритму, виконане на папері або з використанням ЕОМ у будь-якому графічному редакторі (наприклад, Paint).

При виконанні даної роботи рекомендується використовувати також методичні вказівки до лабораторних робіт №1,2.

В задачах цієї роботи необхідно реалізувати той або інший циклічний процес, який виконується або за заздалегідь відому кількість кроків, або до досягнення деякої умови (ітераційні алгоритми). В останньому випадку корисно підстраховатися від зациклювання, яке може виникнути через некоректні вхідні дані. Для цього достатньо установити ліміт кроків з видачею повідомлення у разі його вичерпання.

В ітераційних алгоритмах задана похибка використовується для перевірки модуля

різниці знайденого наближення і точного значення, однак якщо останнє є невідомим, можна оцінювати різницю між сусідніми ітераціями.

Часто в задачах для обчислення наступного доданку зручно використовувати попередній доданок, а не організовувати додатковий цикл.

Приклад. Перевірити чисельно справедливість наступного розкладу експоненти:

$\exp(x)=1+x/1!+x^2/2!+x^3/3!+\ldots+x^n/n!+\ldots$.

і оцінити "швидкість збігання пошуком кількості доданків, необхідних для досягнення заданої похибки ε.

Розв'язок.

Вхідними даними у цій задачі є аргумент x і похибка ε.

Вихідними даними є значення експоненти на кожній ітерації - *exp* і кількість ітерацій, необхідна для досягнення заданої точності - *ki.*

Проміжні дані: поточний номер елементу розкладення - *i*, елемент розкладення $y=x^i / i!$, факторіал - *f*, сума елементів розкладення – *s.*

Суть розв'язку полягає у послідовному обчисленні в циклі елементів розкладення $x^i / i!$ і порівнянні їх з заданою похибкою ε. Тоді умовою виходу з циклу буде наступна умова:

$$x^i / i! \le \varepsilon$$

Визначимо закономірності у заданому розкладенні: початкове значення степеню аргументу і знаменника - 1, кінцеве обмежується похибкою; степінь і знаменник змінюється з кроком 1 в бік зростання.

Обчислення факторіалу будемо виконувати рекурентно, тобто:

n!=(n-1)!*n=(n-2)!*(n-1)*n=…=1*2*…*n.

Алгоритм у вигляді набору інструкцій:

1. Ввести x - аргумент і ε - похибку.

2. Покласти $i = 1$, $s = 1$, $f = 1$ - початкові значення відповідно номеру елементу розкладення, суми елементів розкладення і факторіалу.

3. Обчислити $y=x^i / f$ - поточний елемент розкладення.

4. Якщо $y <= \varepsilon$ - перейти до пункту 8.

5. Додати до суми поточний елемент розкладення: $s = s + y$.

6. Збільшити номер елементу і перерахувати факторіал: $i = i + 1$, $f=f\times i$.

7. Перейти до пункту 3.

8. Вивести $ki = i$ — кількість витрачених кроків і $exp = s$ - значення експоненти. Схема алгоритму представлена на наступному рисунку.

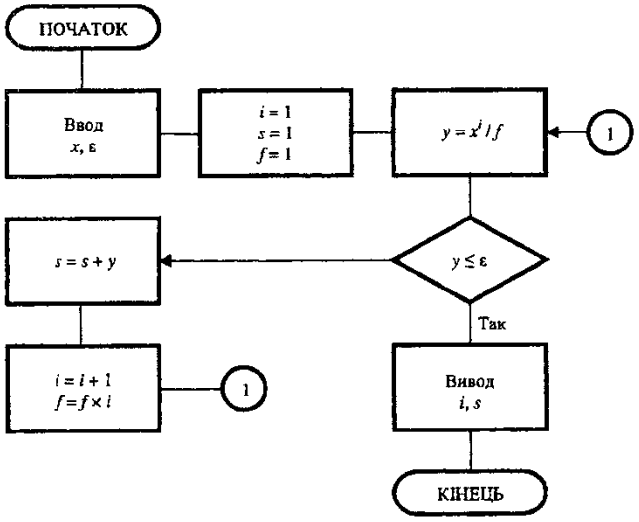


Рисунок 2 - Схема алгоритму розв'язку задачі про розкладення експоненти.

Тестування алгоритму краще провести за допомогою калькулятора. Звіт з тестування приведено далі:

$x = 2$, $\varepsilon = 0{,}01$. $\exp(x) = 7{,}38$.
$i = 0$, $s = 1$;
$i = 1$, $s = s + 2^1 / 1! = 1 + 2 / 1 = 1 + 2 = 3$;
$i = 2$, $s = s + 2^2 / 2! = 3 + 4 / 2 = 3 + 2 = 5$;
$i = 3$, $s = s + 2^3 / 3! = 5 + 8 / 6 = 5 + 1{,}333333 = 6{,}333333$;
$i = 4$, $s = s + 2^4 / 4! = 6{,}333333 + 16 / 24 = 6{,}333333 + 0{,}666666 = 6{,}999999$;
$i = 5$, $s = s + 2^5 / 5! = 6{,}999999 + 32 / 120 = 6{,}999999 + 0{,}266667 = 7{,}266667$;
$i = 5$, $s = s + 2^6 / 6! = 7{,}266667 + 64 / 720 = 7{,}266667 + 0{,}088889 = 7{,}355556$;
$i = 6$, $s = s + 2^7 / 7! = 7{,}355556 + 128 / 5040 = 7{,}355556 + 0{,}025397 = 7{,}380953$;

Відповідь: необхідна кількість доданків для точності $\varepsilon = 0{,}01$ $ki = 6$. Результат розкладення $exp = 7{,}380953$.

ЗАВДАННЯ

1. Вивчити теоретичні відомості і методичні вказівки до лабораторної роботи.
2. Вибрати завдання на лабораторну роботу згідно варіанту.
3. Розробити алгоритм розв'язку у текстовому вигляді.
4. Скласти схему алгоритму розв'язку задачі.
5. Розробити систему контрольних прикладів і перевірити роботу алгоритму.
6. Зробити висновки.

ВАРІАНТИ ЗАВДАНЬ

1. Чисельно впевнитися, чи є задана функція $y = f(x)$ парною або непарною на заданому відрізку $-a \le x \le a$. Врахувати похибку обчислень і можливі точки розриву функції. Перевірити, наприклад, для функцій $y = x^4$, $y = tg\ x$, $y = e^x$, виконуючи їх обчислення на відрізку [-5; 5] з кроком 0,1.

2. Для заданих a і b знайти всі точки з цілочисельними координатами, які знаходяться всередині еліпса $x^2 / a^2 + y^2 / b^2 = 1$.

3. Для заданих a і b знайти всі точки з цілочисельними координатами, які знаходяться в середині еліпса $x^2/a^2 + y^2/b^2 = 1$.

4. Матеріальна точка кидається на горизонтальну площину під кутом α до неї зі швидкістю v_0. При кожному ударі по площині кінетична енергія точки зменшується в β разів. Знайти абсциси перших n точок дотику. Опором повітря знехтувати.

5. Отримати таблицю перерахунку міль в кілометри і навпаки (1 миля = 1,609344 км) для відстаней, які не перевищують k км, в наступному вигляді:

милі	км
0,6214	1,0000
1,0000	1,6093
1,2428	2,0000
1,8641	3,0000
2,0000	3,2187

6. Для заданих m та x обчислити біном Ньютона $(1 + x)^m$ безпосередньо за формулою розкладення в ряд:

$$(1+x)^m = \sum_{i=0}^{m} C_m^i x^i .$$

Для обчислення C_m^i використовувати рекурентну формулу:

$$C_m^{i+1} = C_m^i \frac{m-i}{i+1};\ C_m^0 = 1.$$

7. Порівняти швидкість збігання (кількість доданків для досягнення заданої точності ε) наступних розкладень числа π:

$$\pi = 4\left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \ldots\right);$$

$$\pi = 3 + 4\left(\frac{1}{2 \cdot 3 \cdot 4} - \frac{1}{4 \cdot 5 \cdot 6} + \frac{1}{6 \cdot 7 \cdot 8} - \ldots\right).$$

8. Скільки співмножників треба взяти в добутку:

$$\prod_{k=1}^{n}\left(1 + \frac{(-1)^k}{2k+1}\right) = \frac{\sqrt{2}}{2},$$

щоб рівність виконувалась до шостої значущої цифри, тобто з похибкою не більше 10^{-6}?

9. Для заданих *a* і *p* обчислити $x = \sqrt[p]{a}$ за рекурентним співвідношенням Ньютона:

$$x_{n+1} = \frac{1}{p}\left[(p-1)x_n + \frac{a}{x_n^{p-1}}\right]; x_0 = a.$$

10. Числа $x_1, x_2, \ldots$ послідовно надходять з пристрою вводу. Всі числа зберігати в пам'яті не треба; після початку вводу кожного числа треба обчислити і надрукувати середнє значення всіх введених чисел.

11. В учбовому закладі задається початок учбового дня, тривалість "пари" або уроку, тривалість звичайної і великої перерви (та їх "місце" в розкладі), кількість пар (уроків). Отримати розклад дзвінків на весь учбовий день.

12. Для заданих значень *n* та *x* обчислити вираз:

$$s = \sin x + \sin \sin x + \ldots + \sin \sin \sin \ldots \sin x \ (n \text{ разів}).$$

13. Упевнитися в справедливості рівності:

$$a^x = 1 + \frac{x \ln a}{1!} + \frac{(x \ln a)^2}{2!} + \ldots + \frac{(x \ln a)^n}{n!}.$$

14. Підприємець розпочав справу і взяв кредит розміром k гривень під *p* процентів річних. За прогнозом, його справа повинна давати прибуток r гривень на рік. Чи зможе він накопити суму, достатню для погашення кредиту, і якщо так, то через скільки років?

15. Для заданого $x > 1$ обчислити $y = \sqrt{x}$ за ітераційною формулою:

$$y_i = \frac{1}{2}\left(y_{i-1} + \frac{x}{y_{i-1}}\right)$$

із заданою похибкою є і початковим наближенням $y_0 = x$. Скільки ітерацій треба виконати?

16. Для заданого є знайти найменше n таке, що $2^n / n! < \varepsilon$. Вивести всі члени послідовності від 1-го до *n*-го.

17. Стверджується, що функція $y = f(x)$ періодична з періодом *T*. Перевірити це чисельно, обчисленням функції з постійним кроком на відрізку [0; 5T]. Врахувати похибку обчислень і можливі точки розриву функції.

18. Відомий час початку і кінця роботи деякого автобусного маршруту з одним автобусом на лінії, а також довжина маршруту у хвилинах і час відпочинку на кінцевих зупинках. Скласти добовий розклад цього маршруту (моменти відправлення з кінцевих пунктів).

19. Перевірити чисельно другу чудову границю:

$$\lim_{n \to \infty}\left(1 + \frac{1}{n}\right)^n = e,$$

для $n = 1, 2, 3, \ldots$ При якому n досліджуваний вираз відрізняється від *e* менш ніж на задану похибку ε?

20. Прямокутник на площині задається чотирма числами: *a, b,* c, *d* - координатами його вершин. Послідовно вводяться габарити *n* прямокутників. В процесі вводу знаходити площину їх перетину, не запам'ятовуючи власне габаритів.

КОНТРОЛЬНІ ПИТАННЯ

1. Яке призначення команди циклу і як вона діє ?
2. Які є правила складання блок-схем?
3. Які види блоків використовують у блок-схемах?
4. Які є етапи створення програми?
5. Що таке цикл з параметром, формат його використання?
6. Що таке вкладені цикли?

Лабораторна робота № 4. Аналіз методів сортування

Мета: навчитися досліджувати алгоритми сортування.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Алгоритми сортування

Алгоритмом сортування називається алгоритм для впорядкування деяких елементів множини. Зазвичай алгоритмом сортування вважають алгоритм упорядкування елементів за зростанням або спаданням.

Практично кожен алгоритм сортування можна розбити на 3 етапи:

порівняння, що визначає впорядкованість пари елементів;

перестановку, міняє місцями пару елементів;

власне алгоритм що сортує, який здійснює порівняння і перестановку елементів до тих пір, поки всі елементи множини не будуть впорядковані.

Алгоритми сортування мають велике практичне застосування. Їх можна зустріти там, де мова йде про обробку та зберігання великих обсягів інформації. Деякі завдання обробки даних вирішуються простіше, якщо дані заздалегідь упорядкувати.

Існує величезна кількість різних алгоритмів сортування. Універсального, найкращого алгоритму сортування на даний момент не існує. Однак, маючи приблизні характеристики вхідних даних, можна підібрати метод, який працює оптимальним чином при даних умовах. Для цього необхідно знати параметри, за якими буде проводитися оцінка алгоритмів.

Основні параметри алгоритмів сортування

Час сортування - основний параметр, що характеризує швидкодію алгоритму.

Необхідна пам'ять - один з параметрів, який характеризується тим, що ряд алгоритмів сортування вимагають виділення додаткової пам'яті для тимчасового зберігання даних. При оцінці пам'яті що використовується не буде враховуватися місце, яке займає вихідний масив даних та витрати які не залежать від вхідної послідовності, наприклад, на зберігання коду програми.

Стійкість - це параметр, який відповідає за те, що сортування не змінює взаємного розташування рівних елементів.

Природність поведінки - параметр, якій вказує на ефективність методу при обробці вже відсортованих, або частково відсортованих даних. Алгоритм поводиться природно, якщо враховує цю характеристику вхідної послідовності і працює краще.

Сортування вибором

Алгоритм сортування за зростанням:

1. Знайти мінімальне значення в поточному списку.

2. Знайдене мінімальне значення міняється місцем з елементом на першій позиції.

3. Повторюємо сортування, виключивши з розгляду вже відсортований перший елемент, тобто, починаючи з другої позиції.

Послідовність кроків при n=7 показана на рисунку нижче.

Рисунок 1 - Схема алгоритму сортування вибором

Алгоритм не використовує додаткової пам'яті: всі операції відбуваються "на місці".

Метод простого обміну (метод бульбашки).

Алгоритм полягає в повторюваних проходах по сортованому масиву. За кожен прохід елементи послідовно порівнюються попарно і, якщо порядок у парі невірний, виконується обмін елементів. Проходи по масиву повторюються до тих пір, поки на черговому проході не виявиться, що обміни більше не потрібні, що означає - масив відсортований. При проході алгоритму, елемент, що стоїть не на своєму місці, «спливає» до потрібної позиції як бульбашка у воді, звідси і назва алгоритму.

Алгоритм сортування методом бульбашки:

1. Порівнюємо перший і другий елементи масиву. Якщо перший елемент більший, ніж другий, то міняємо їх місцями.
2. Порівнюємо другий і третій елементи масиву. Якщо другий елемент більший, ніж третій, то міняємо їх місцями.
3. ...
4. Порівнюємо передостанній (N-1) і останній (N) елементи масиву. Якщо передостанній елемент більший, ніж останній, то міняємо їх місцями.

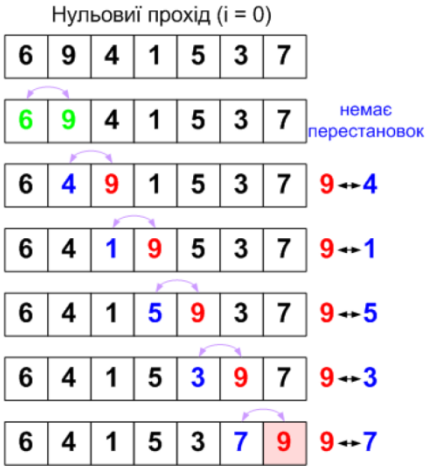


Рисунок 2 - Схема алгоритму сортування обміном

В результаті останнім елементом в масиві у нас виявиться найбільший елемент.

Другий крок сортування методом бульбашки - повторюємо вищевказані дії для частини масиву, починаючи з першої позиції до N-1:

1. Порівнюємо перший і другий елементи масиву. Якщо перший елемент більший, ніж другий, то міняємо їх місцями.

2. Порівнюємо другий і третій елементи масиву. Якщо другий елемент більший, ніж третій, то міняємо їх місцями.

3. ...

4. Порівнюємо елемент (N-2) і елемент (N-1) масиву. Якщо (N-2)-й елемент більший, ніж елемент (N-1), то міняємо їх місцями.

В результаті передостанній елемент в масиві у нас теж буде на своєму, "відсортованому" місці.

На наступних кроках сортування методом бульбашки вищевказані дії повторюються для частини масиву, починаючи з 1 позиції до (N-2) (крок 3), а потім для діапазону 1 .. (N-3) і так далі до діапазону 1 .. 2.

Після завершення останнього кроку масив буде відсортований за зростанням.

Рисунок 3 - Схема алгоритму сортування обміном

По-перше, розглянемо ситуацію, коли на якому-небудь з проходів не відбулося жодного обміну. Це означає, що всі пари розташовані в правильному порядку, так що масив вже відсортований. Одна з можливостей поліпшення алгоритму полягає в запам'ятовуванні, чи проводився на даному проході обмін. Якщо ні - алгоритм закінчує роботу.

Процес поліпшення можна продовжити, якщо запам'ятовувати не тільки сам факт обміну, але і індекс останнього обміну k. Дійсно: всі пари сусідніх елементів з індексами, більшими за k, вже розташовані в потрібному порядку. Подальші проходи можна закінчувати на індексі k, замість того щоб рухатися до встановленої заздалегідь верхньої межі.

ЗАВДАННЯ

1. Вивчити теоретичні відомості до лабораторної роботи.
2. Вибрати завдання на лабораторну роботу згідно варіанту.
3. Розробити алгоритм розв'язку у текстовому вигляді.
4. Скласти схему алгоритму розв'язку задачі.
5. Розробити систему контрольних прикладів і перевірити роботу алгоритму.
6. Зробити висновки.

ВАРІАНТИ ЗАВДАНЬ

1. Написати алгоритм сортування методом бульбашки у порядку зростання та дослідити його складність.

2. Написати алгоритм сортування методом бульбашки у порядку спадання та

дослідити його складність.

3. Написати алгоритм сортування методом перестановки у порядку зростання та дослідити його складність.

4. Написати алгоритм сортування методом перестановки у порядку спадання та дослідити його складність.

5. Написати алгоритм сортування методом підрахунку у порядку зростання та дослідити його складність.

6. Написати алгоритм сортування методом підрахунку у порядку спадання та дослідити його складність.

7. Написати алгоритм сортування методом Шелла у порядку зростання та дослідити його складність.

8. Написати алгоритм сортування методом Шелла у порядку спадання та дослідити його складність.

9. Написати алгоритм сортування методом швидкого сортування у порядку зростання та дослідити його складність.

10. Написати алгоритм сортування методом швидкого сортування у порядку спадання та дослідити його складність.

11. Написати алгоритм сортування методом швидкого сортування з розподілом за алгоритмом Ломуто у порядку зростання та дослідити його складність.

12. Написати алгоритм сортування методом швидкого сортування з розподілом за алгоритмом Ломуто у порядку спадання та дослідити його складність.

13. Написати алгоритм сортування методом вибірки у порядку зростання та дослідити його складність.

14. Написати алгоритм сортування методом вибірки у порядку спадання та дослідити його складність.

15. Написати алгоритм сортування методом порозрядного цифрового сортування у порядку зростання та дослідити його складність.

16. Написати алгоритм сортування методом порозрядного цифрового сортування у порядку спадання та дослідити його складність.

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Для чого потрібні алгоритми сортування?
2. З яких основних частин складається будь-який алгоритм сортування?
3. Які основні параметри алгоритмів сортування ви знаєте?
4. Поясніть принцип роботи сортування вибором.
5. Поясніть принцип роботи сортування методом простого обміну.

Лабораторна робота №5. Проектування алгоритмів обробки одновимірних масивів

Мета роботи: вивчення методики проектування алгоритмів обробки одновимірних масивів, організації їхнього введення і виведення, знаходження суми, добутку скінченого числа елементів одновимірних масивів.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Масив - це впорядкований набір однорідних елементів, що мають ім’я, для зберігання яких наділяється послідовно розташоване поле пам'яті. Масив характеризується ідентифікатором (ім'я), числом вимірів (індекси) і числом змінних у кожному вимірі (верхня межа кожного індексу). Існують одновимірні масиви (вектори, лінійні таблиці), двовимірні масиви (матриці, прямокутні таблиці), а також багатовимірні масиви.

Введення - виведення елементів масиву виконується поелементно. Можливі два способи.

Спосіб 1. Безпосереднє введення-виведення елементів масиву, яке позначається на блок-схемі алгоритму при введенні значень елементів блоками “введення” або “процес” та при виведенні блоком “виведення”.

Такий спосіб доцільно застосовувати, коли масиви мають малі розміри.

Спосіб 2. При великих розмірах масивів, що оброблюються, оператори введення-виведення розміщують усередині циклічної ділянки програми і при кожному виконанні циклу здійснюється введення-виведення одного елемента.

При складанні циклічних програм обробки масивів доцільно сполучати операції введення і безпосередньо обробки масивів, що дозволяє в ряді випадків скоротити обсяг програми.

ЗАВДАННЯ ТА ПОРЯДОК ВИКОНАННЯ

1 Вивчити навчальний матеріал та підготувати відповіді на контрольні питання.

2 Скласти схему алгоритму рішення задачі за варіантом завдання.

3. Оформити звіт.

Приклад 1: Вивести на друк елементи цілочисельного масиву N=[n1, n2, n3,... , n50], які кратні трьом. Сполучимо операції введення поточного значення елемента масиву і його опрацювання.

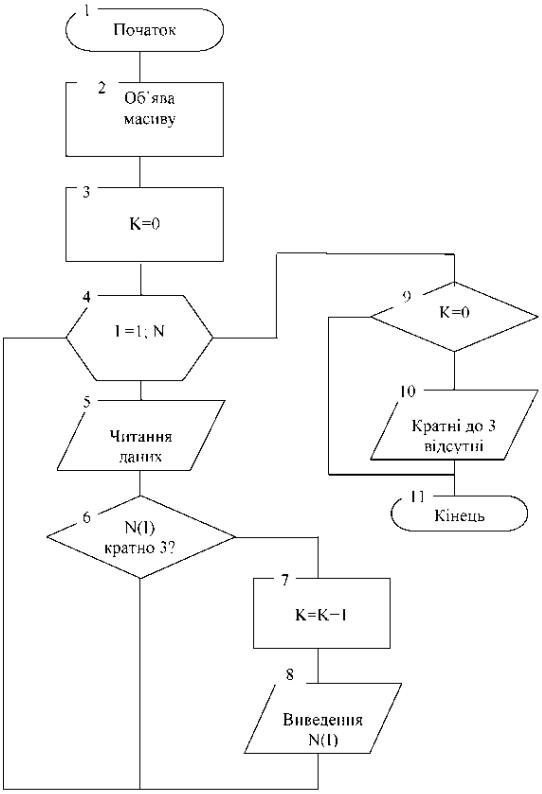


Рисунок 1 – Алгоритм визначення елементів масиву, кратних 3.

Якщо ж потрібно запам'ятати всі значення результатів у пам'яті, то необхідно виділити масив для результатів, і обчислити результат як змінну з індексом.

Приклад 2: Переписати підряд у масив Y додатні елементи масиву (x1, x2, x3,... , x20), а в масив Z - від'ємні.

Особливість рішення полягає в тому, що елементи, які записуються в масиви Z і Y, повинні розташовуватися підряд без пропусків. Це призведе до того, що індекси змінних y_k та z_j набудуть значень, відмінні від значень індексів змінної x_i. Тому в циклі необхідно змінювати значення k та j кожний раз перед записом відповідного елементу x, тобто необхідно використати спосіб організації циклу з кількома параметрами, які одночасно змінюються (рис. 2.).

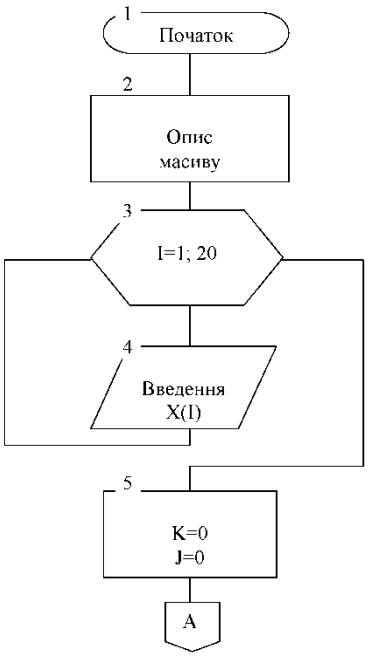


Рисунок 2 – Алгоритм визначення додатних та від'ємних елементів масиву

Знаходження суми $\sum_{i-1}^{n} y_i$ зводиться до її накопичення у вигляді значення змінної в циклі, в який вводяться відповідні додатки. При цьому наступний додаток y_i додається до суми попередніх додатків z_{i-1}, тобто в циклі послідовно обчислюються всі проміжні суми:

$z_1 = z_0 + y_1 = y_1$

$z_2 = z_1 + y_2 = (y_1) + y_2$

.....

$z_i = z_{i-1} + y_i = (y_1 + y_2) + y_i$

.....

$z_n = z_{n-1} + y_n$

$$z_{in} = \sum_{i=1}^{n} y_i$$

Обчислення добутку зводиться до його накопичення в циклі у вигляді значення змінної. Аналогічно накопиченню суми в циклі обчислюються послідовно всі проміжні добутки:

$z_1 = z_0 * y_1 = y_1$

$z_2 = z_1 * y_2 = (y_1) * y_2$

.....

$z_i = z_{i-1} * y_i = (y_1 * y_2) * y_i$

.....

$z_n = z_{n-1} * y_n$

$$z_n = \prod_{i=1}^{n} y_i$$

Якщо не потрібно зберігати у пам'яті комп'ютера додатки і проміжні добутки, вони представляються простими змінними; і тоді рекурентна формула для накопичення добутку має вигляд:

z = z * y. де **y** - множник; **z** - проміжний добуток.

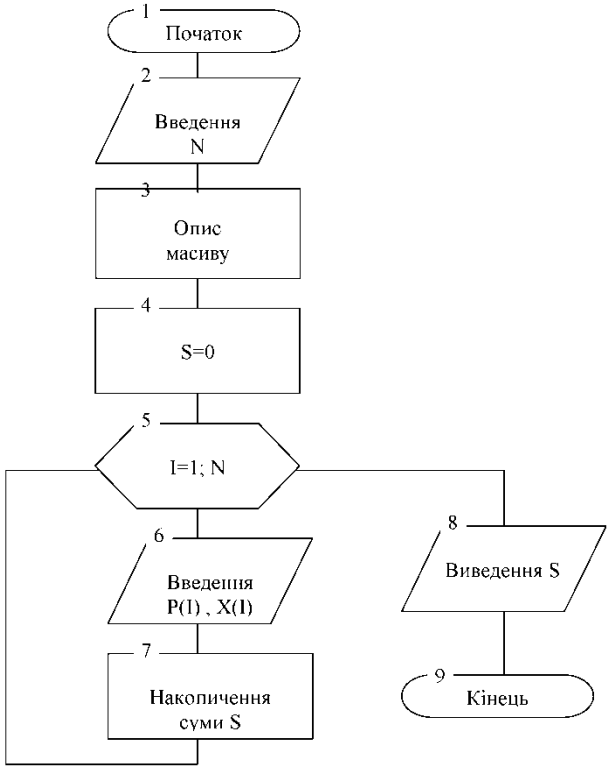
Перед циклом за початкове значення добутку, як правило, приймається одиниця: $z_0 = 1$.

Приклад 3: Скласти алгоритм обчислення математичного сподівання m_x випадкової

величини x, використовуючи розрахункову формулу

$$m_x = \sum_{i=1}^{n} p_i x_i$$

для початкових даних: n = 5; p_i = {0.3; 0.25; 0.2; 0.1; 0.15};x_i = {90; 80; 120; 95; 110}. Початковими даними для обчислення є проста змінна N та два масиви: P і X. Результат обчислення збережемо у простій змінній S. Для накопичення суми організується цикл, параметром якого є індекс I, який змінюється від 1 до N. (рис.3.)



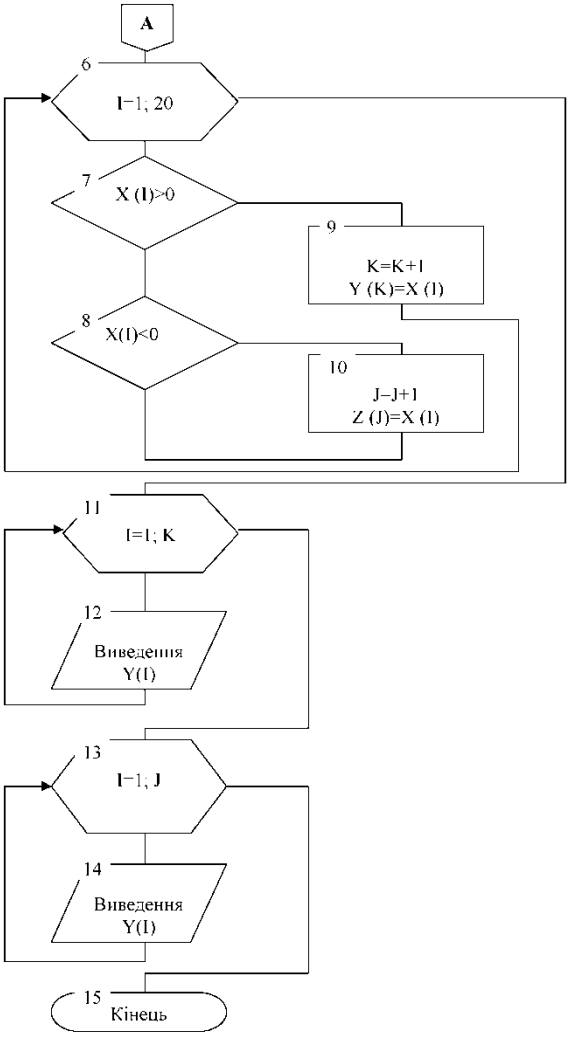


Рисунок 3 – Алгоритм обчислення математичного сподівання

Приклад 4: Скласти алгоритм обчислення середнього геометричного додатних елементів масиву. Схема алгоритму наведена на рис.4.

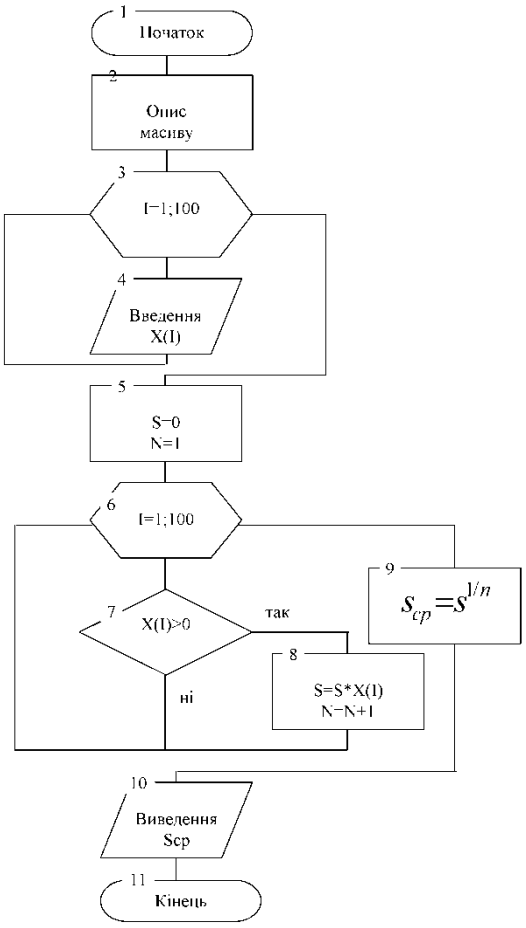


Рисунок 4 – Алгоритм обчислення середнього геометричного

Знаходження найбільшого та (або) найменшого з множень значень виконується у циклі шляхом порівняння деякого поточного значення з найбільшим (найменшим) з усіх попередніх.

Якщо поточне значення більше від найбільшого (MAX) з усіх попередніх, то MAX присвоюється значення поточного. У протилежному випадку зберігається попереднє значення. Процес можна описати наступною залежністю:

$$y_{max} = \begin{cases} y_i, & \text{якщо } y_i > y_{max} \\ y_{max}, & \text{якщо } y_i \le y_{max} \end{cases}$$

Після закінчення циклу значення y_{max} буде найбільшим з усіх розглянутих значень y_i.

Аналогічним чином знаходиться найменше (MIN) серед набору елементів, із використанням залежності:

$$y_{min} = \begin{cases} y_i, & \text{якщо } y_i < y_{min} \\ y_{min}, & \text{якщо } y_i \ge y_{min} \end{cases}$$

При знаходженні найбільшого (MAX) та (або) найменшого (MIN) елемента масиву за початкове значення найбільшого та найменшого у цьому випадку слід брати значення першого елемента масиву, а в циклі, пошук найбільшого та найменшого елементу виконувати, починаючи з другого елементу масиву.

Приклад 5: Скласти алгоритм знаходження найбільшого та найменшого елемента

масиву (x_1, x_2, ….x_k) та їх порядкових номерів.

Особливість пошуку полягає в тому, що необхідно знаходити номер найбільшого та найменшого елемента. Прийнявши за початкове значення і максимальний і мінімальний елемент, запам’ятаємо їх номери $i_{max} = 1$ і $i_{min} = 1$. У циклі при виконанні умов $x_i > x_{max}$ та $x_i < x_{min}$ присвоюють, відповідно, $x_{max} = x_i$, $i_{max} = i$ та $x_{min} = x_i$, $i_{min} = i$ (рис. 5.)

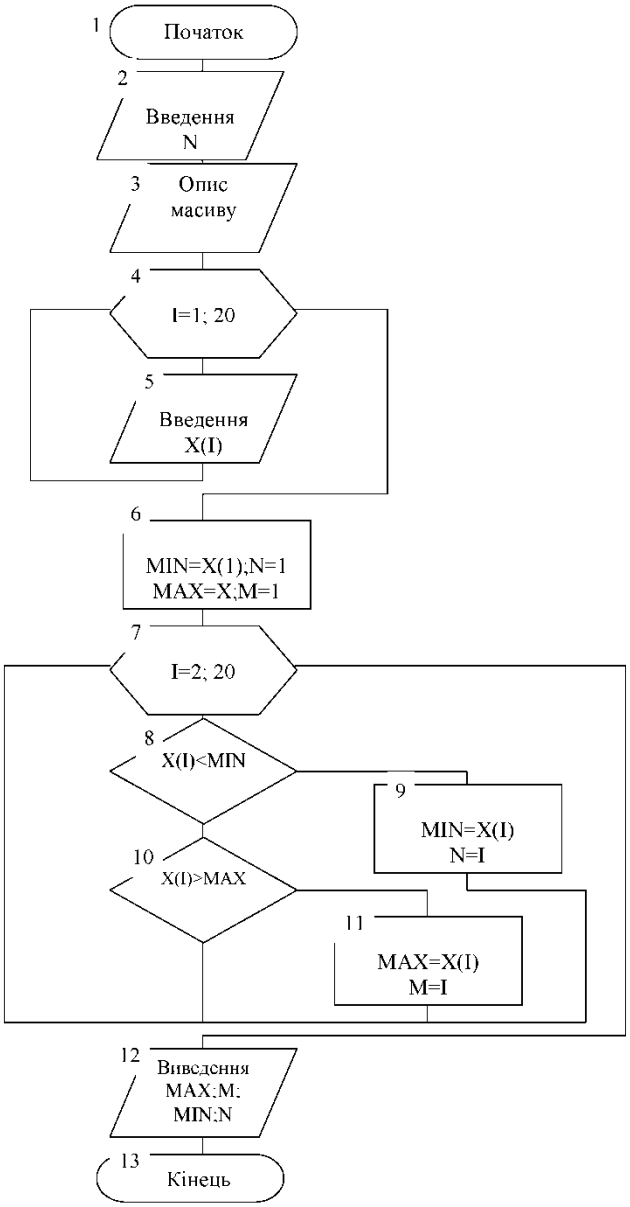


Рисунок 5 – Алгоритм знаходження найменшого та найбільшого елементів масиву та їх порядкових номерів

ВАРІАНТИ ІНДИВІДУАЛЬНИХ ЗАВДАНЬ

Розробити схему алгоритму обробки одновимірного масиву відповідно до заданого варіанта.

1. Обчислити суму і кількість елементів, більших нуля і менших нуля, масиву X(10).

2. Скласти програму для запису підряд у масив M додатних, а в масив H - від’ємних елементів масиву (x_1, x_2, …, x_{10}).

3. Обчислити відношення максимального елемента до середнього арифметичного

елементів масиву A(12).

4. Визначити максимальний елемент масиву B(15) і переставити його місцем з кінцевим елементом масиву.

5. Скласти програму для обчислення добутку і кількості додатних елементів масиву X(10).

6. Визначити мінімальний елемент масиву C(10) і переставити його місцем з першим елементом масиву.

7. Знайти максимальний і мінімальний елементи масиву H(12) і поміняти їх місцями.

8. Розташувати в масиві P спочатку додатні, а потім від'ємні елементи масиву A(15).

9. Визначити суму і кількість елементів масиву H(15), кратних трьом.

10. Обчислити значення функції:

$$H = \sum_{i=1}^{20} \frac{x_i}{i!},$$

де x_i - елементи масиву ($x_1, x_2, \ldots, x_{20}$).

11. Обчислити середнє геометричне додатних елементів масиву X(15) і записати його на місце від'ємних елементів цього масиву.

12. Для масиву X(10), що має додатні і від'ємні елементи, обчислити суму елементів, що стоять на парних місцях.

13. Для масиву X(15), що має додатні і від'ємні елементи, обчислити середнє арифметичне додатних елементів.

14. Для масиву X(15), що має додатні і від'ємні елементи, обчислити суму елементів, що стоять на непарних місцях.

15. Визначити суму і кількість елементів масиву H(15), кратних п'яти.

КОНТРОЛЬНІ ПИТАННЯ

1. Дайте визначення масиву.
2. Які характеристики мають масиви?
3. Опишіть особливості алгоритмів введення і виведення елементів масиву.
4. Опишіть реалізацію накопичення суми і добутку скінченого числа елементів масиву.
5. Як організувати рахунок кількості елементів?
6. Як організувати процес формування нових масивів у процесі розв'язання задач?
7. Як організувати знаходження найбільшого і найменшого елементів масиву?
8. Як організувати процес упорядкування елементів масиву?
9. У чому особливість організації циклу при обробці масивів?

Лабораторна робота №6. Проектування алгоритмів обробки двовимірних масивів

Мета роботи: вивчення методики проектування алгоритмів обробки двовимірних масивів, організації їх введення і виведення, знаходження суми, добутку скінченого числа елементів двовимірних масивів, обчислення координат положення елементів.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Двовимірний масив характеризується ідентифікатором, який має два виміри (індекси), і числом значень у кожному вимірі (верхньою межею кожного індексу). Положення елемента в двовимірному масиві визначається індексами. Один вказує на номер рядка, а другий - на номер стовпчика, на перетині яких розташовується елемент. Тобто елемент двовимірного масиву A представляється ідентифікатором A(i,j), де

i - номер рядка масиву A;

j - номер стовпчика масиву A.

Розмір масиву (кількість елементів масиву) – добуток числа рядків на число стовпчиків масиву, тобто для масиву A(10,5) його розмір складає 50.

Порядок введення елементів двовимірного масиву (за рядками чи за стовпчиками), якщо він не пов'язаний з обробкою масиву, байдужний і організується за допомогою вкладеного циклу за будь-яким порядком проходження параметрів зовнішнього і внутрішнього циклів по i та j . При обробці двовимірного масиву, а також при введенні, пов’язаним з обробкою, порядок проходження параметрів зовнішнього і внутрішнього циклів по i та j визначається умовами задачі.

При виведенні двовимірного масиву за параметр зовнішнього циклу приймається I (фіксується значення I та виводяться елементи рядків у внутрішньому циклі, змінюючи значення J).

ЗАВДАННЯ ТА ПОРЯДОК ВИКОНАННЯ

1 Вивчити навчальний матеріал та підготувати відповіді на контрольні питання.

2 Скласти схему алгоритму рішення задачі за варіантом завдання.

3. Оформити звіт.

Приклад: Із матриці A(10,5) вивести на друк додатні елементи.

Схема алгоритму наведена на рис.1.

Щоб скоротити обсяг алгоритму, введення елементів матриці A(I, J) та їх обробка об'єднані. Пропонується наступний порядок виконання вкладеного циклу. У зовнішньому циклі задається параметр циклу I, який дорівнює 1. Так як I<10, то виконуються всі операції, що входять в даний цикл. Першою такою операцією є заголовок внутрішнього циклу, де параметру J задається значення 1, а потім вводиться значення елементу масиву A(1,1). Далі виконується перевірка умови A(1,1)>0. Якщо умова виконується, то збільшується значення лічильника K на 1 і виводиться значення елемента A(1,1); якщо умова не виконується, то виконується перехід на продовження внутрішнього циклу J = 2,3,4,5. Після закінчення внутрішнього циклу у зовнішньому циклі I приймає значення 2 і знов повторюється внутрішній цикл п’ять разів. Після 10 - кратного виконання зовнішнього циклу виконується перехід на блок «кінець».

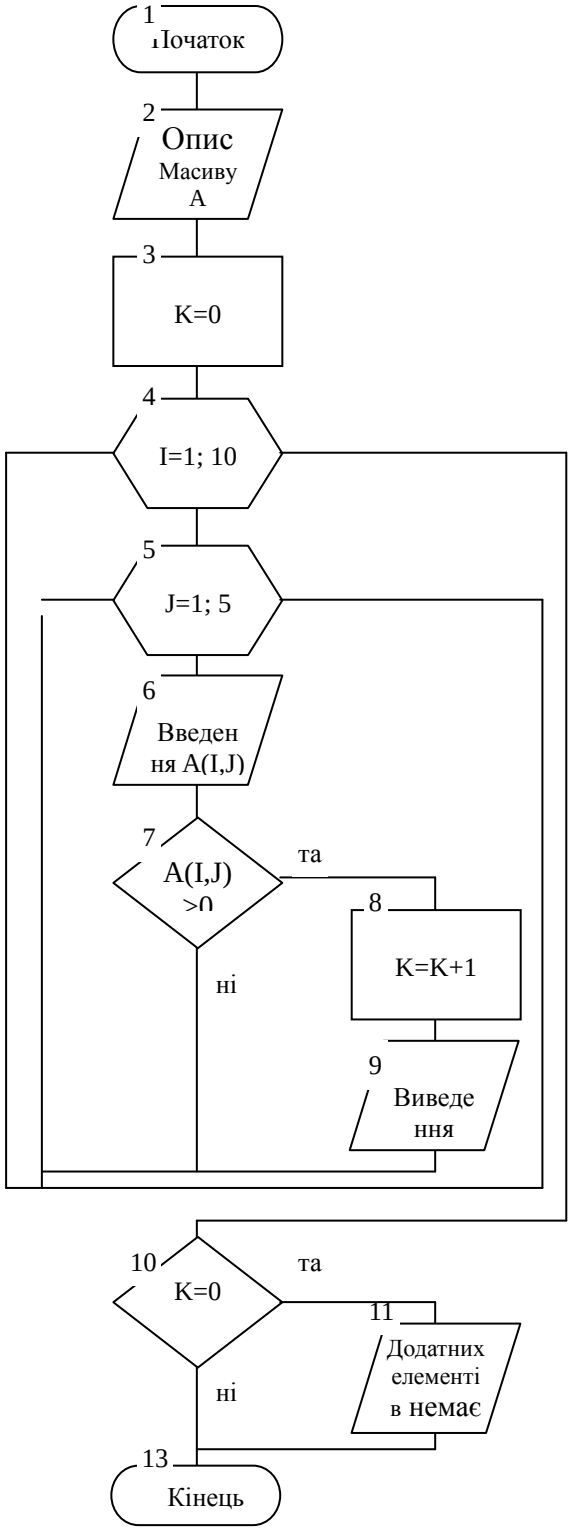


Рисунок 1 – Алгоритм виведення додатних елементів масиву.

ВАРІАНТИ ІНДИВІДУАЛЬНИХ ЗАВДАНЬ

Розробити схему алгоритму обробки двовимірного масиву відповідно до заданого варіанта.

1. Визначити суму і число додатних елементів кожного стовпчика матриці A(5,6).
2. Визначити добуток від'ємних елементів, розташованих у рядках з непарними

номерами і максимальний елемент рядка 3 для матриці A(7,5).

3. Визначити суму і число елементів матриці A(6,6), що знаходяться під головною діагоналлю.

4. Визначити суму і число додатних елементів матриці C(7,7), що знаходяться над головною діагоналлю.

5. У матриці A(7,7) змінити місцями елементи, розташовані у верхній і нижній чвертях, обмежених головною і побічною діагоналями (за винятком елементів, розташованих на діагоналях).

6. У матриці A(7,7) змінити місцями елементи, розташовані в лівій і правій чвертях, обмежених головною і побічною діагоналями (за винятком елементів, розташованих на діагоналях).

7. Знайти в кожному рядку матриці A(11,7) максимальний і мінімальній елементи і помістити їх на місце першого та кінцевого елементів рядку відповідно.

8. Для матриці A(6,7), складеної з цілих чисел, знайти для кожного рядку число елементів, кратних п'яти, і найбільший із отриманих результатів.

9. Знайти в кожному рядку матриці A(15,15) найбільший елемент і змінити його місцями з відповідним елементом головної діагоналі.

10. Для матриці розміром NxN заповнити одиницями нижню половину, включаючи середній рядок, якщо N-непарне, за винятком елементів, розташованих справа від головної діагоналі.

11. Визначити різницю між середнім арифметичним додатних і від'ємних елементів стовпчиків із непарними номерами матриці A(7,10).

12. Упорядкувати за зростанням елементи кожного рядка матриці A(10,7).

13. Для матриці A(14,7) обчислити суму елементів кожного рядка матриці. Результати розташувати за убуванням.

14. Для матриці розміром NxN заповнити одиницями ліву половину, за винятком елементів, розташованих справа від побічної головної діагоналі.

15. Обчислити добуток сум елементів головної і побічної діагоналей квадратної матриці розміром NxN.

16. Для матриці розміром NxN заповнити одиницями верхню половину, за винятком елементів, розташованих справа від побічної діагоналі.

17. Для матриці розміром NxN переставити місцями елементи головної і побічної діагоналі.

18. Для матриці розміром NxN заповнити одиницями верхню половину, за винятком елементів, розташованих зліва від головної діагоналі.

19. Задано матрицю A(10,10). Записати її в масив B(10,10) так, щоб стовпчики матриці A були рядками в матриці B.

20. Для матриці розміром NxN заповнити одиницями праву половину, включаючи середній стовпчик, якщо N - непарне, за винятком елементів, розташованих зліва від головної діагоналі.

21. Обчислити середнє арифметичне додатних елементів для матриці A(10,10).

22. Обчислити і запам'ятати кількість від'ємних елементів кожного стовпчика матриці A(10,10). Вивести на друк отримані данні.

23. Ввести початкові данні в перші 5 рядків і перші 4 стовпчики матриці A(6,5). Обчислити середнє арифметичне значення елементів кожного рядку і записати його в 6-ий рядок. Надрукувати отриману матрицю в узвичаєному вигляді.

24. Знайти найбільший елемент матриці A(10,10), що знаходиться над головною діагоналлю.

25. Для матриці A(10,10), складеної з цілих чисел, знайти для кожного стовпчика число елементів, кратних трьом, і найбільший з отриманих результатів.

26. Для матриці A(10,10) обчислити суму елементів кожного стовпчика матриці. Результати розташувати за убуванням.

27. Знайти в кожному стовпчику матриці A(15,15) найбільший і найменший елементи і змінити їх місцями. Отриману матрицю вивести на друк в узвичаєному вигляді.

28. Знайти в кожному стовпчику матриці A(10,10) найбільший елемент і змінити його місцями з елементом головної діагоналі.

29. Знайти стовпчики з найбільшою і найменшою сумою елементів матриці A(12,9).

30. Записати на місце від'ємних елементів квадратної матриці одиниці, а на місце додатних – нулі. Вивести на друк верхню трикутну матрицю в узвичаєному вигляді.

31. Упорядкувати за зростанням елементи кожного стовпчика матриці A(9,11).

32. Задано матрицю A(10,10). Записати в масив B елементи її головної діагоналі. Отриманий масив вивести на друк.

КОНТРОЛЬНІ ПИТАННЯ

1 Дайте визначення двовимірного масиву.

2 Опишіть алгоритми введення та виведення двовимірних масивів.

3 Які операції повинні бути у схемах обробки двовимірних масивів?

4 Які характеристики мають двовимірні масиви?

5 У чому складається особливість організації циклу при обробці двовимірних масивів?

6 Опишіть правила організації вкладеного циклу з урахуванням порядку перебору елементів матриці.

7 Як організувати виведення нижньої трикутної матриці в узвичаєному вигляді?

8 Як організувати виведення верхньої трикутної матриці в узвичаєному вигляді?

9 Як організувати виведення матриці розміром $H^{x}M$ елементів в узвичаєному вигляді?

10 Описати реалізацію прийомів накопичення суми та добутку, запам'ятовування результатів, знаходження найбільшого та найменшого елемента у двовимірному масиві.

Лабораторна робота №7. Проектування алгоритмів з використанням підпрограм

Мета роботи: набуття навичок проектування алгоритмів задач з використанням функцій, що визначаються користувачем, поняття процедури та параметрів підпрограм.

ТЕОРЕТИЧНІ ВІДОМОСТІ

При розв'язанні багатьох практичних задач виникає необхідність у багаторазовому обчисленні параметрів за одним алгоритмом, але при різних значеннях змінних. Щоб програма була компактною, доцільно оформити ці обчислення у вигляді програмних модулів, де описується процедура обчислень для формальних параметрів. При необхідності у відповідних місцях основної програми звертаються до підпрограми, задаючи необхідні значення фактичних параметрів, при яких повинні бути виконані обчислення. При цьому передача значень параметрів до підпрограми та запам'ятовування результатів її роботи виконується у головній програмі.

Розрізняють підпрограми двох типів: бібліотечні та користувача.

Бібліотечні підпрограми складаються раніше та зберігаються у пам'яті машини. Для того, щоб їх використовувати у головній програмі, необхідно звернутися до них, вказавши ім'я підпрограми та фактичні значення параметрів. У вигляді бібліотечних

підпрограм оформлюються найбільше уживані функції, обчислення інтегралів, розв'язання різноманітних рівнянь тощо.

Текст процедури обчислень підпрограми користувача розробляється програмістом для однієї конкретної задачі і підпрограма, як правило, не використовується при реалізації інших задач.

Схема алгоритму задачі при використанні підпрограм складається із декількох алгоритмів, один із яких є основним (головним) і обов'язково має наступні блоки: "початок", "кінець"; присвоєння формальним параметрам фактичних значень; "підпрограма", присвоєння фактичним змінним результатів обчислень підпрограм. В інших алгоритмах описуються дії, які виконуються у підпрограмах. Такі алгоритми складаються з блоків: "вхід", опису обчислювального процесу з використанням формальних параметрів, "вихід".

При використанні вкладених підпрограм схема алгоритму, що викликає програму повинна містити блочні компоненти основної програми: блоки присвоєння формальним параметрам програми, що викликається, блоки "підпрограма", блоки присвоєння фактичним змінним головної програми результатів обчислень підпрограми, що викликається.

При проектуванні схем алгоритмів з використанням підпрограм необхідно забезпечити узгодження формальних та фактичних параметрів за кількістю, порядком, типом.

На момент звертання до підпрограми усі вхідні фактичні параметри повинні бути визначені, тобто їх значення повинні бути відомими.

ЗАВДАННЯ ТА ПОРЯДОК ВИКОНАННЯ

1 Вивчити навчальний матеріал та підготувати відповіді на контрольні питання.

2 Скласти схему алгоритму рішення задачі за варіантом завдання.

3. Оформити звіт.

Приклад 1: У дисплейному залі в обігу знаходяться n дисплеїв, кожен з яких може відмовити з імовірністю p. Імовірність того, що за день відмовлять не менше m дисплеїв, можна оцінити за формулою

$$W=\sum_{k=m}^{n}\binom{k}{n}p^k(1-p)^{n-k},$$

де $\binom{k}{n}=\frac{n!}{k!(n-k)!}$ – число сполучень з n по k.

Скласти програму розрахунку імовірності w=f(n, m, p) для заданих значень n=15; p = 0,05. Значення m ввести у діалоговому режимі.

При визначенні числа сполучень необхідно три рази обчислювати факторіал від різних аргументів. Відомо, що факторіал від аргументу $x \geq 0$ визначається за формулою

$$F = x! = \begin{cases} 1, & \text{якщо } x = 0 \\ \prod_{i=1}^{x} i & \text{якщо } x > 0 \end{cases}$$

Для обчислення факторіалу використовується підпрограма, рис. 1.

Головна програма

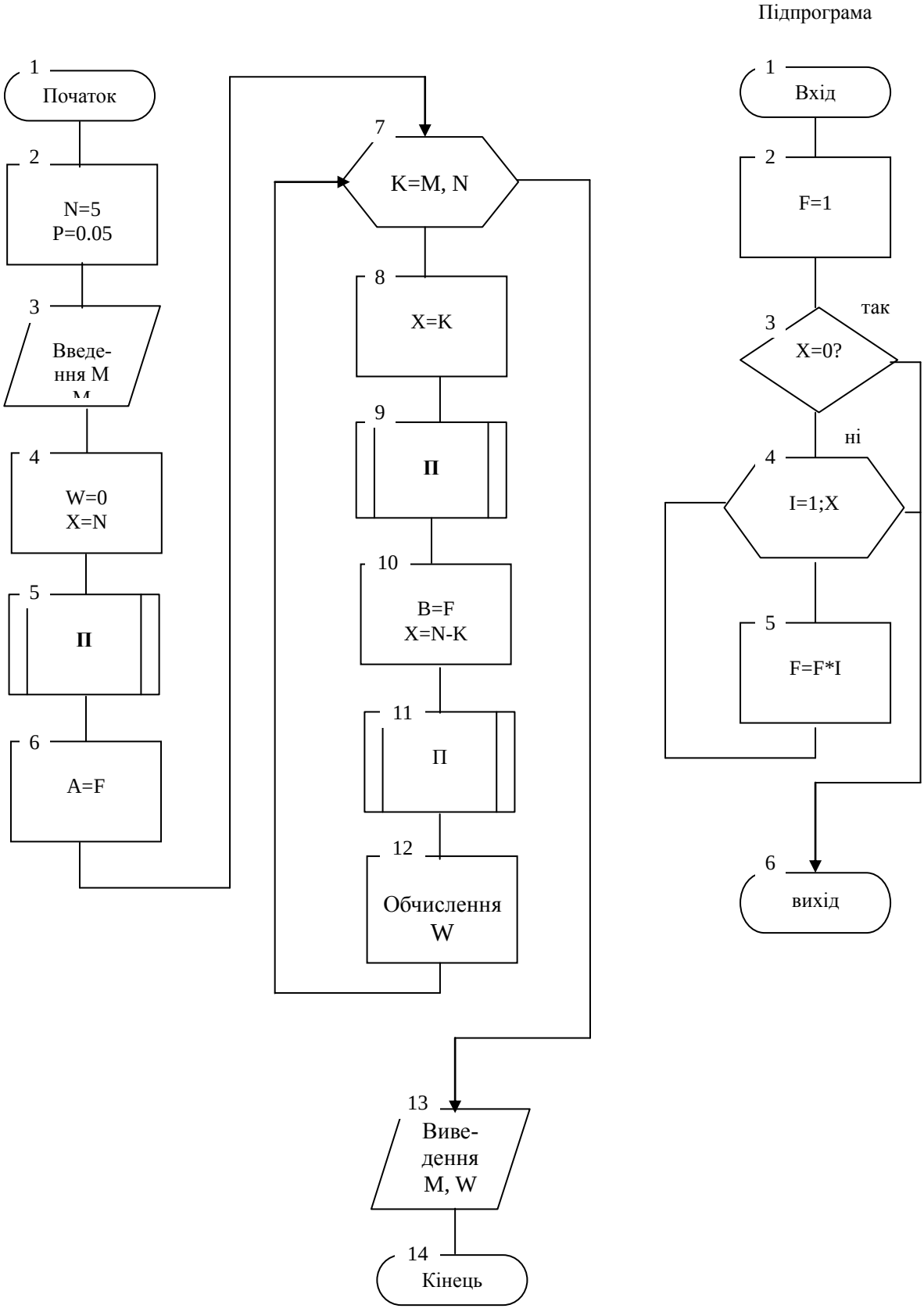


Рисунок 1 – Алгоритм розрахунку імовірності

Підпрограма містить оператор присвоєння F=1 та цикл, в якому здійснюється обчислення факторіалу X!. Слід мати на увазі, що значення X задається в алгоритмі основної програми до виконання підпрограми.

Перед першим зверненням змінній X присвоюється значення фактичного параметра N (блок 4 алгоритму головної програми), після цього здійснюється перехід до підпрограми (блок 5). Після обчислення N! управління з підпрограми передається блоку 6, де значення результату F присвоюється фактичній змінній A. Аналогічно обчислюються B=K! та F= (N - K)!.

Приклад 2: У програмі необхідно багаторазово виводити різноманітні заголовки, оздоблюючи вгорі та знизу рядком зірочок, наприклад:

```
**********************************
ГРАФІК ПЕРШОЇ ФУНКЦІЇ  Y=F (X)
**********************************
```

Доцільно друк заголовку і друк рядка зірочок зробити із застосуванням підпрограм.

Схема алгоритму наведена на рис. 2.

В основній програмі текстовій змінній A$ (блок 2) присвоюється текст заголовку, що повинен бути надрукований, і управління передається підпрограмі П1 друку заголовку (блок 3). Першим оператором цієї підпрограми є оператор виклику другої підпрограми - друку рядка зірочок (блок 3 підпрограми П2). Друга підпрограма друкує рядок зірочок і повертає управління до першої підпрограми П1, на блок 3, що здійснює друк тексту заголовку. Після цього управління знову передається другій підпрограмі П2. Друкується рядок зірочок, що оздоблює текст заголовку знизу, та передається управління першої підпрограмі і вже з неї здійснюється повернення до основної програми (блок 4). Присвоюючи змінній A$ різноманітний текст, можна звертатися до вкладених підпрограм декілька разів.

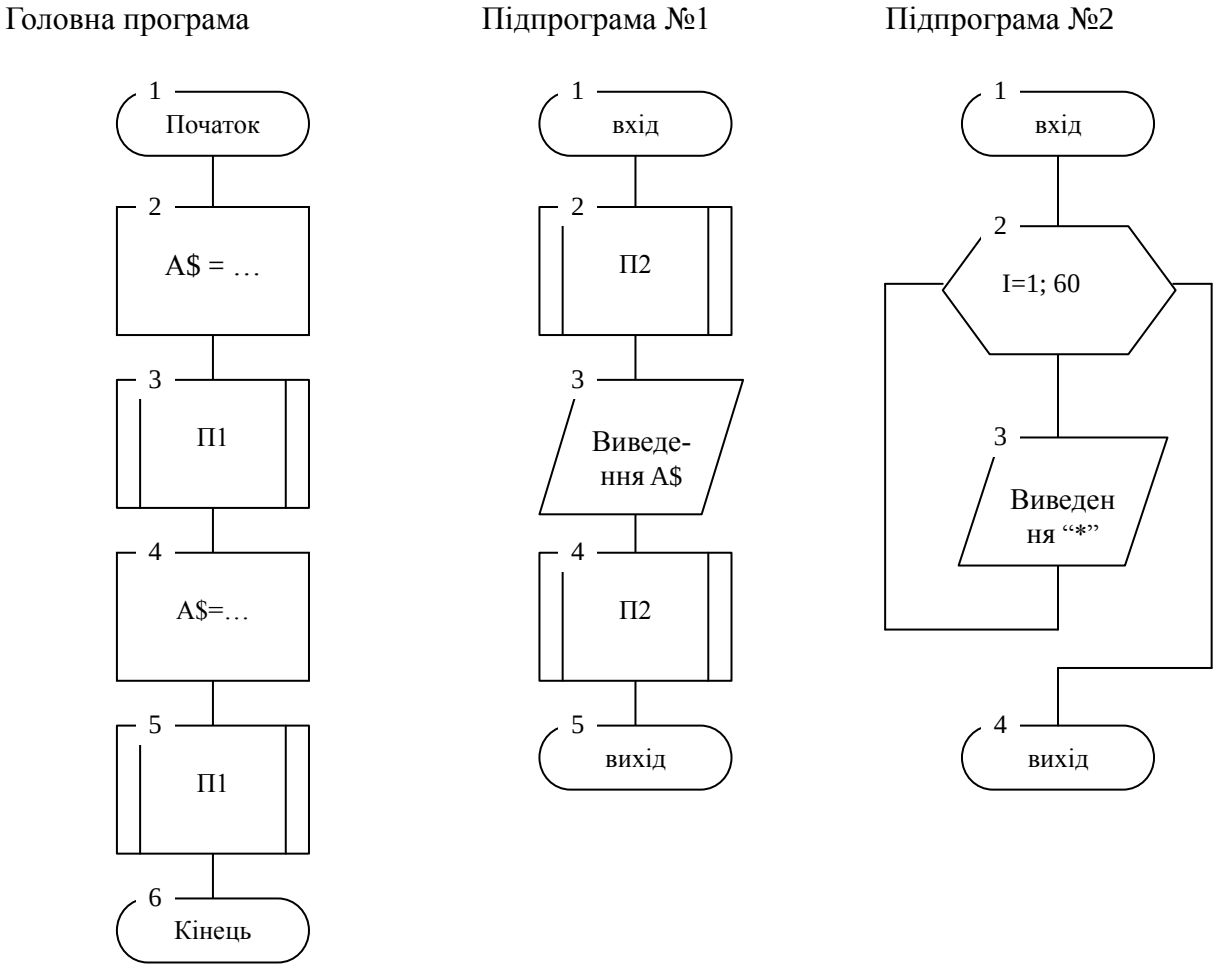


Рисунок 2 – Алгоритм з використання вкладених підпрограм

ВАРІАНТИ ІНДИВІДУАЛЬНИХ ЗАВДАНЬ

1. Обчислити H=(A1+A2)/K1*K2, де A1 і K1 - сума і кількість додатних елементів масиву X(P); A2 і K2 - сума і кількість додатних елементів масиву У(Т). Обидві суми обчислювати в одній підпрограмі.
2. Обчислити

$$R= \frac{\log_2 x - \log_b y}{2\log_{(b+2)}(x+y)}$$

3. Обчислити і запам'ятати суми додатних елементів кожного рядка матриці A (4,3), B (5,4).
4. Обчислити H=(X1+У1)/(X2+У2),
   де X1 і X2 - корені рівняння $2x^2+x - 4=0$
   У1 і У2 - корені рівняння $3y^2+2y - 1=0$.
5. Знайти найбільші елементи і їхні порядкові номери масивів X(P), У(Т).
6. Переписати додатні елементи масиву X(P) і масиву У(Т) в масив А підряд. Запис в масив А здійснюється в підпрограмі.
7. Знайти найменші елементи і номери рядків і стовпчиків, в яких вони

розміщені, для матриць A(5,8) і B(3,5).

8. Вивести на друк елементи цілочисельних матриць A(5,8) і B(5,3), які є кратними п'яти.

9. Обчислити

$$H=(\sum_{i=1}^{10}\sin x_i+\sum_{i=1}^{10}\cos x_i)/\sum_{i=1}^{10}\cos x_i\ ,$$

де змінну x задано масивом. Суми обчислювати в одній підпрограмі.

10. Обчислити $H=(X_{мах} - У_{мін})/2$,

де $X_{мах}$ - максимальний елемент масиву X(15);

$У_{мін}$ - мінімальний елемент масиву У(10).

11. Обчислити і запам'ятати кількість від'ємних елементів кожного стовпчика для матриць A(5,5), B(3,6).

12. Обчислити суми елементів верхньої трикутної матриці для матриць A(5,5), B(3,3).

13.Обчислити суми елементів головних діагоналей матриць A(P, P), B(T, T).

14. Обчислити суми і кількість елементів в інтервалі від A до B для матриць X(5,6) і У(3,5).

15. Перебудувати масиви X(5) і У(6), розташувавши в них підряд тільки додатні елементи; замість інших елементів записати нулі.

16. Визначити число додатних елементів до першого від'ємного в масивах X(5), У(10), H(8).

17. Обчислити і запам'ятати суми додатних елементів кожного стовпчика матриць A(5,7), B(4,3).

18. Перебудувати масиви X(5) і У(7), розташувавши в них підряд тільки від'ємні елементи, замість інших елементів записати нулі.

19. Знайти найменші елементи та їхні порядкові номери масивів X(N), У(M).

20. Обчислити H= (A1+A2)/(K1*K2),

де A1 і K1 - сума і кількість додатних елементів масиву X(P);

A2 і K2 - сума і кількість від'ємних елементів масиву У(T).

Обидві суми обчислювати в одній підпрограмі.

21. Вивести на друк елементи цілочисельних матриць A(7,4) і B(5,3), які є кратними двом.

22. Обчислити і запам'ятати суми від'ємних елементів кожного рядка матриці A(4,5), B(5,3).

23. Обчислити суми елементів нижче головної діагоналі для матриць A(5,5), B(3,3).

24. Обчислити середнє арифметичне від'ємних елементів для масивів A(8), B(7).

25. Обчислити суми елементів нижніх трикутних матриць для матриць A(5,5), B(7,7).

26. Обчислити суми елементів головної діагоналі для матриць A(5,5), B(3,3).

27. Знайти рядки з найбільшою сумою елементів в матрицях A(5,5) і B(3,3). Вивести на друк знайдені рядки і суми.

28. Знайти рядки з найменшою сумою елементів в матрицях A(5,7) і B(3,2). Вивести на друк знайдені рядки і суми.

29. Обчислити середнє геометричне від'ємних елементів для масивів A(10), B(12). Обчислення здійснювати в підпрограмі.

30. Переписати від'ємні елементи масиву X (10) і масиву У (8) в масив A підряд. Запис в масив A здійснювати в підпрограмі.

31. Обчислити

$$D=(\sum_{i=1}^{10}\cos x_i+\sum_{i=1}^{7}\sin x_i)/2,$$

де x_i - елементи масиву. Суми обчислювати в одній підпрограмі.

32. Знайти рядки з найменшою сумою елементів в матрицях A(5,2) і B(3,4). Вивести на друк знайдені рядки і суми.

КОНТРОЛЬНІ ПИТАННЯ

1 При яких умовах доцільно використання підпрограми?
2 Визначити поняття формальних та фактичних параметрів.
3 Які основні блоки використовуються в алгоритмі головної програми?
4 Які основні блоки використовуються в алгоритмі підпрограми?
5 Перерахувати, як узгоджуються формальні та фактичні параметри.
6 Визначити поняття "вкладена підпрограма".

Лабораторна робота № 8. Алгоритми пошуку

Мета: навчитися складати алгоритми для розв'язку задач, набуття практичних навичок застосування алгоритмів пошуку.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Людина опрацьовує таблиці візуально: одного погляду на таблицю достатньо, щоб відповісти на питання, що цікавлять. Комп’ютер так поступати не може. Він опрацьовує дані в масиві методом перегляду усіх елементів з метою відшукання потрібних. Перевага комп’ютера – він переглядає й аналізує дані швидко і може опрацювати великі масиви даних, що зазвичай людині не під силу.

Головні прийоми опрацювання масивів:

- щоб визначити елементи масиву, які задовольняють умову задачі, потрібно переглянути і проаналізувати всі елементи, від першого до останнього, за допомогою команд циклу і розгалуження;
- з елементами масиву можна виконувати такі ж дії, як і з простими даними відповідних типів (числовими чи текстовими);
- опрацьовують масиви поелементно (тобто виконують дії не з масивом, а з кожним його елементом). Доступ до кожного елемента забезпечує змінна з індексом (наприклад, a[i]). Такий вид доступу називається прямим.

Алгоритми пошуку

Пошук – процес знаходження конкретної інформації у масиві даних. Зазвичай дані представляють собою записи, кожна з яких має хоча б один ключ. Ключ пошуку – це поле запису, за значенням якого відбувається пошук. Метою пошуку є знаходження всіх записів (якщо вони є) з даними значенням ключа.

Структуру даних, в якій проводиться пошук, можна розглядати як таблицю символів (таблицю імен або таблицю ідентифікаторів) – структуру, що містить ключі та дані, та допускає дві операції – вставку нового елемента і повернення елемента із заданим ключем. Іноді таблиці символів називають словниками по аналогії з добре відомою системою впорядкування слів в алфавітному порядку: слово – ключ, його тлумачення – дані.

Пошук є однією дій, що найчастіше зустрічаються у програмуванні. Існує безліч різних алгоритмів пошуку, які принципово залежать від способу організації даних. У кожного алгоритму пошуку є свої переваги і недоліки. Тому важливо вибрати той алгоритм, який краще всього підходить для вирішення конкретного завдання.

Поставимо задачу пошуку в лінійних структурах. Нехай задано масив даних, що описується як масив, що складається з деякої кількості елементів. Перевіримо, чи входить заданий ключ в даний масив. Якщо входить, то знайдемо номер цього елемента масиву, тобто, визначимо перше входження заданого ключа (елемента) у вихідному масиві.

Таким чином, визначимо загальний алгоритм пошуку даних:

Крок 1. Обчислення елемента, що часто передбачає отримання значення елемента, ключа елемента і т.д.

Крок 2. Порівняння елемента з еталоном або порівняння двох елементів (в залежності від постановки задачі).

Крок 3. Перебір елементів множини, тобто проходження по елементах масиву.

Основні ідеї різних алгоритмів пошуку зосереджені в методах перебору і стратегії пошуку.

Розглянемо основні алгоритми пошуку в лінійних структурах більш докладно.

Послідовний (лінійний) пошук

Послідовний (лінійний) пошук – це найпростіший вид пошуку заданого елемента на деякій множині, який здійснюється шляхом послідовного порівняння чергового значення з шуканим до тих пір, поки ці значення не співпадуть.

Ідея цього методу полягає в наступному. Масив елементів проглядається послідовно в деякому порядку, який гарантує, що будуть переглянуті всі елементи множини (наприклад, зліва направо). Якщо в ході перегляду масиву буде знайдений шуканий елемент, перегляд припиняється з позитивним результатом, якщо ж буде переглянуто всю множину, а елемент не буде знайдений, алгоритм повинен видати негативний результат.

Алгоритм послідовного пошуку

Крок 1. Вважаємо, що значення змінної циклу i = 0.

Крок 2. Якщо значення елемента масиву x [i] дорівнює значенню ключа key, то повертаємо значення, рівне номеру шуканого елемента, і алгоритм завершує роботу. В іншому випадку значення змінної циклу збільшується на одиницю i = i +1.

Крок 3. Якщо i <k, де k – число елементів масиву x, то виконується Крок 2, в іншому випадку – робота алгоритму завершена і повертається значення рівне -1.

При наявності в масиві декількох елементів із значенням key даний алгоритм знаходить тільки перший з них (з найменшим індексом).

Недоліком розглянутого алгоритму пошуку є те, що в гіршому випадку здійснюється перегляд усього масиву. Тому даний алгоритм використовується, якщо масив містить невелику кількість елементів.

Переваги послідовного пошуку полягають в тому, що він простий в реалізації, не вимагає сортування значень множини, додаткової пам'яті та додаткового аналізу функцій. Отже, може працювати в потоковому режимі при безпосередньому отриманні даних з будь–якого джерела.

Бінарний (двійковий) пошук

Бінарний (двійковий, дихотомічний) пошук – це пошук заданого елемента на впорядкованій множині, який здійснюється шляхом неодноразового розподілу цієї множини на дві частини таким чином, що шуканий елемент потрапляє в одну з цих частин. Пошук закінчується при збігу шуканого елемента з елементом, що є границею між частинами масиву або за відсутності шуканого елемента.

Бінарний пошук застосовується до відсортованої множини і полягає в послідовному розбитті множини навпіл і пошуку елемента тільки в одній половині на кожній ітерації.

Таким чином, ідея цього методу полягає в наступному. Пошук потрібного значення серед елементів упорядкованого масиву (за зростанням або за спаданням) починається з визначення значення центрального елементу цього масиву. Значення даного елемента порівнюється з шуканим значенням і залежно від результатів порівняння робляться певні дії. Якщо шукане і центральне значення виявляються рівні, то пошук завершується успішно. Якщо шукане значення менше центрального або більше, то формується масив,

що складається з елементів, які знаходяться ліворуч або праворуч від центрального відповідно. Потім пошук повторюється у новому масиві (рис.1).

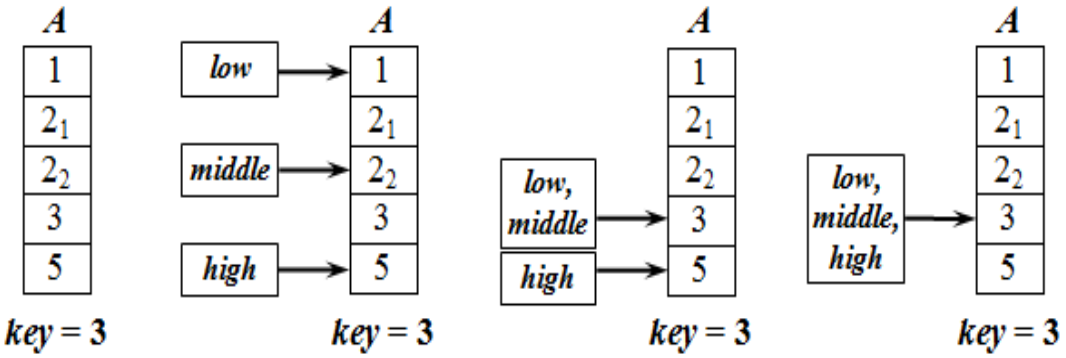


Рисунок 1 – Демонстрація алгоритму бінарного пошуку

Алгоритм бінарного пошуку

Крок 1. Визначити номер середнього елемента масиву middle = (high + low) / 2.

Крок 2. Якщо значення середнього елемента масиву дорівнює шуканому, то повертаємо значення, рівне номеру шуканого елемента, і алгоритм завершує роботу.

Крок 3. Якщо шукане значення більше значення середнього елемента, то візьмемо в якості масиву всі елементи праворуч від середнього, в іншому випадку візьмемо в якості масиву всі елементи ліворуч від середнього (в залежності від характеру впорядкованості). Перейдемо до кроку 1.

У масиві може зустрічатися декілька елементів зі значеннями, рівними ключу. Даний алгоритм знаходить перший елемент, що співпав з ключем, який в порядку проходження в масиві може бути ні першим, ні останнім серед рівних ключу. Наприклад, в масиві чисел 1, 5, 5, 5, 5, 5, 5, 7, 8 з ключем key = 5 співпаде елемент з порядковим номером 4, який не відноситься ні до першого, ні до останнього.

Існують дві модифікації даного алгоритму для пошуку першого і останнього входження. Все залежить від того, як вибирається середній елемент: округленням в меншу або більшу сторону. У першому випадку середній елемент належить до лівої частини масиву, а в другому – до правої.

У процесі роботи алгоритму бінарного пошуку розмір фрагмента, де цей пошук повинен проводитися, щораз зменшується приблизно в два рази. Перевагою даного алгоритму є відносно швидке виконання пошуку, в порівнянні з алгоритмом послідовного пошуку. Недолік полягає в тому, що бінарний пошук може застосовуватися тільки на впорядкованій множині.

ЗАВДАННЯ ТА ПОРЯДОК ВИКОНАННЯ

7. Вивчити теоретичні відомості і методичні вказівки до лабораторної роботи.
8. Вибрати завдання на лабораторну роботу згідно варіанту.
9. Розробити алгоритм розв'язку у текстовому вигляді.
10. Скласти схему алгоритму розв'язку задачі.
11. Розробити систему контрольних прикладів і перевірити роботу алгоритму.
12. Зробити висновки.

ВАРІАНТИ ІНДИВІДУАЛЬНИХ ЗАВДАНЬ

Розробити алгоритм пошуку в одновимірних масивах. Забезпечити пошук потрібних елементів використовуючи один із алгоритмів пошуку: лінійний пошук, лінійний пошук з

бар’єром, бінарний пошук, пошук Фібоначі, пошук з перестановкою в початок, пошук с транспозицією. Оцінити час виконання операцій.

№	Завдання
1	Скласти алгоритм знаходження номера найменшого елемента масиву A[N], а також номер цього елемента.
2	Відомі номера x1, x2,...,x10 виграшних білетів. Визначити , чи є заданий номер виграшний.
3	Знайти найбільший елемент, що знаходиться на головній діагоналі. Вивести на екран його індекси.
4	Дано лінійну таблицю A[1:n] дійсних чисел. Складіть алгоритм пошуку середнього арифметичного всіх від’ємних елементів даної таблиці.
5	Дано лінійну таблицю A[1:n] дійсних чисел. Складіть алгоритм підрахунку кількості повторень у даній таблиці заданого числа x.
6	Дано лінійну таблицю A[1:n] дійсних чисел. Складіть алгоритм обчислення середнього арифметичного мінімального і максимального елементів даної таблиці.
7	Дано лінійну таблицю A[1:n] дійсних чисел. Складіть алгоритм пошуку середнього арифметичного всіх додатних елементів даної таблиці.
8	Дано лінійну таблицю A[1:n] дійсних чисел. Складіть алгоритм підрахунку середнього арифметичного всіх непарних елементів даної таблиці.
9	Дано лінійну таблицю A[1:n] дійсних чисел. Складіть алгоритм, що подвоює значення всіх додатних елементів даної таблиці, а всі від’ємні заміняє нулем.
10	Дано лінійну таблицю A[1:n] дійсних чисел. Складіть алгоритм, що підраховує кількість додатних і кількість від’ємних елементів таблиці, а потім визначає, яка з величин є більшою.
11	Дано лінійну таблицю A[1..n]. Складіть алгоритм, у якому всі елементи, що мають парні порядкові номери подвоюються, а непарні – збільшуються на 1.
12	Дано лінійну таблицю A[1..n]. Складіть алгоритм знаходження суми всіх позитивних елементів даної таблиці і підрахунку їхньої кількості.

КОНТРОЛЬНІ ПИТАННЯ

1. У чому полягає ідея лінійного пошуку? Сформулюйте умову припинення перегляду елементів.
2. Які переваги в алгоритмі лінійного пошуку надає використання бар'єра? Як реалізується бар'єр?
3. У якому випадку може використовуватися бінарний пошук? У чому його основна ідея?
4. Сформулюйте алгоритм бінарного пошуку.

Лабораторна робота № 9. Особливості використання рядків, множин та структур

Мета: навчитися складати алгоритми для розв'язку задач, набуття практичних навичок використовування рядків, множин та структур.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Рядкові величини

Рядкові величини використовуються в алгоритмах обробки текстів. Типовий приклад такого алгоритму – алгоритм пошуку усіх входжень даного слова в текст і заміни цього слова його синонімом.

Значеннями рядкових величин є слова, тобто ланцюжки символів (букв). Для того, щоб уточнити систему команд виконавця алгоритмів обробки текстів, потрібно почати з алфавіту, в символах якого записуються ці тексти.

За багатовікову історію свого розвитку людство винайшло багато способів запису інформації на матеріальних носіях (на каменях, глиняних табличках, папері, магнітному диску, у чипі пам'яті комп'ютера). У навчальних посібниках з інформатики згадуються: текстова інформація, числова інформація, графічна інформація, звукова (аудио) інформація, відеоінформація. Однак, незалежно від виду інформації, елементарною одиницею її представлення є символ. Справді, тексти складаються з букв і розділових знаків. Числа ми записуємо за допомогою цифр і знаків-роздільників. В інформатиці графічні образи (картинки) формуються з кольорових точок, розташованих у прямокутній таблиці. Тому кожну точку представляють три числа: точка = (номер рядка, стовпця, номер кольору). Таке представлення (кодування) графічної інформації називають цифруванням. Цифрування інформації – універсальний спосіб її представлення в комп'ютері.

Виконавець текстових алгоритмів має у своєму розпорядженні алфавіт символів, з яких повинні складатися оброблювані ним тексти.

Алфавіт – це скінчений впорядкований набір символів. Символи розташовані в алфавіті в строго визначеному порядку – один за іншим.

Таким чином, в алфавіті існує перший (початковий) символ і заключний (кінцевий) символ. Для кожного символу алфавіту, крім заключного, можна вказати наступний за ним символ. Так само, для кожного символу алфавіту, крім початкового, можна вказати попередній йому символ.

Отже, символи алфавіту можна порівнювати за допомогою операцій порівняння:

$$s_1 < s_2,\ s_1 > s_2,\ s_1 <= s_2,\ s_1 >= s_2,\ s_1 = s_2,\ s_1 <> s_2.$$

Наприклад, для алгоритмів обробки слів, що представляють десяткові дроби, алфавіт складається з наступних 13-ти символів:

{ 0 1 2 3 4 5 6 7 8 9 . - + }

Для того, щоб відрізняти символи від інших позначень, у яких використовуються букви, символи беруть у лапки.

У цьому алфавіті початковий символ – '0', а заключний – знак '+'. Символи нашого алфавіту задовольняють ланцюжкові нерівностей:

'0' < '1' < '2' < '3' < '4' < '5' < '6' < '7' < '8' < '9' < '.' < '-' < '+'

Алгоритми обробки текстів українською мовою вимагають значно більше символів. Це – прописні і малі літери абетки, розділові знаки, дужки, інші символи, що зустрічаються в текстах.

Для алфавіту символів, які використовуються в інформатиці, існують міжнародні стандарти.

Словом називається скінчена послідовність (ланцюжок) символів з алфавіту виконавця. Слова є значеннями рядкових величин.

Поряд з терміном „слово” в програмуванні використовуються також терміни „рядок”, „літерал”. (Термін „текст”, яким ми користувалися, в програмуванні має дещо інше значення.)

Операції порівняння

Порядок, визначений на символах алфавіту, використовують для визначення порядку на множині рядків. Цей спосіб порівняння й упорядкування рядків (слів) називають алфавітним або лексикографічним порядком. Алфавітний порядок застосовують при складанні словників, довідників і т.і.

Для того, щоб визначити, яке зі слів менше, порівнюють спочатку перші букви цих слів, потім другі, і так далі. Припустимо, що порівнюються слова P = ‘$a_1a_2…a_k$‘ і Q = ‘$b_1b_2…b_m$’...

Якщо ‘a_1‘< ‘b_1‘ то P < Q;

Якщо ‘a_1‘ > ‘b_1‘ то P > Q;

Якщо ‘a_1‘= ‘b_1‘ то порівнюємо символи ‘a_2 ‘ і ‘b_2’ , і так далі.

На кожному кроці порівняння можливі наступні ситуації:

Одне зі слів уже прочитано цілком, а друге - ще ні. У цьому випадку менше те слово, символи якого вже прочитані.

Обидва слова прочитані цілком. У цьому випадку слова рівні.

Символ одного зі слів менше, ніж символ іншого. У цьому випадку менше те слово, символ якого менше.

Символи обох слів рівні. У цьому випадку переходимо до порівняння наступних символів.

Наприклад:

'мама' < 'папа', оскільки 'м' < 'п'.

'алгебра' < 'алгоритм', оскільки перші три букви рівні, а 'е' < 'о'.

‘program’ < ‘programmer’, оскільки при побуквеному порівнянні в першому слові всі букви будуть вичерпані, а в другому – ще ні.

Операції Знайти і Замінити

Оскільки спосіб представлення інформації у виді рядка є універсальним, будь-яку алгоритмічну задачу можна розглядати як задачу обробки рядкових даних. Виявляється, що можна так визначити систему команд виконавця, щоб він, з одного боку, обробляв тільки рядки, а з іншого боку – був здатний виконати будь-який алгоритм. Таку систему команд знайшов у 50-х роках XX століття А.А. Марков. Відкрита ним алгоритмічна система називається нормальними алгоритмами Маркова.

Основна операція нормального алгоритма Маркова – операція підстановки:

У слові W знайти слово P і замінити його словом Q.

Цю операцію прийнято записувати формулою P => Q. Операція підстановки знаходить в рядку перше зліва входження слова P і замінює його на слово Q. Марков вводить також заключну підстановку P => Q. Після застосування заключної підстановки алгоритм закінчує свою роботу.

Нормальний алгоритм Маркова є набором підстановок (звичайних і заключних):

$$p_1 \rightarrow q_1,\ p_2 \rightarrow q_2, \ldots, p_k \rightarrow q_k$$

Нормальні алгоритми Маркова використовуються в теоретичних дослідженнях алгоритмічних проблем.

Форма запису рядків

Щоб відрізнити в тексті алгоритму рядок від інших примітивів (імені величини, числа і т.п.) рядок прийнято брати в лапки. Наприклад: 3.14 – запис числа, а ‘3.14’ – запис рядка.

Операції над рядками

В сучасних алгоритмічних мовах набір команд виконавця (перелік операцій над рядками) у порівнянні з алгоритмічною системою Маркова значно ширший. В нього можуть входити разом з командами порівняння, наприклад такі команди:

З’єднати рядки P та Q.
Видалити з рядка P, починаючи з n-тої букви, рядок з m букв.
Вставити в рядок P, починаючи з n-тої букви, рядок Q.
Скопіювати з рядка P, починаючи з n-тої букви, рядок з m букв.
Визначити довжину рядка P.
Визначити у рядку P найменший номер букви, починаючи з якої в нього входить рядок Q.

В алгоритмічних мовах для позначення таких команд використовуються скорочення. Наприклад:

З’єднати (P, Q).
Видалити (P, n, m).
Вставити (P, n, Q).
Копія (P, n, m).
Довжина (P).
Номер (P, Q).

Для того, щоб програміст правильно використовував ці команди, він має знати їх форму запису та типи їх аргументів та результатів. Наприклад, для команди **Копія (P, n, m)** маємо таку форму запису:

Q = Копія (P, n, m).
Вхід: рядок P, натуральне n, натуральне m.
Вихід: рядок Q.

Ця команда має функціональну форму запису y = f(x). Тому команди такої форми називають функціями. Зауважимо, що результат цієї команди програміст має „повернути” у деяку змінну. В нашому прикладі це змінна Q.

Команда Вставити (P, n, Q) **має процедурну форму запису:**
Вставити (P, n, Q).
Вхід: рядок P, натуральне n, рядок Q.
Вихід: рядок P.

Зауважимо, що в цій команді результат співпадає з одним з аргументів: процедура **Вставити (P, n, Q)** змінює свій перший аргумент – змінну P. Команди такої форми називають процедурами.

Таким чином, набір команд обробки рядкових величин складається з таких процедур та функцій:

Функція З’єднати (P: **рядок**, Q: **рядок**) : **рядок.**
Процедура Видалити (P:**рядок**, n:**натуральне**, m:**натуральне**).
Процедура Вставити (P:**рядок**, n:**натуральне**, Q:**рядок**).
Функція Копія (P:**рядок,** n:**натуральне**, m:**натуральне**):**рядок**
Функція Довжина (P:**рядок**):**натуральне.**
Функція Номер (P:**рядок,** Q:**рядок**):**натуральне.**

Приклад 1. Визначити кількість літер “л”, які містить даний рядок символів.

Спочатку визначимо довжину (кількість символів) для даного рядка. Довжину рядка використаємо як кінцевий параметр циклу. У циклі послідовно, починаючи з першого, братимемо по одному символу і перевірятимемо: якщо вказаний символ рівний літері “л”, то лічильник кількості літер “л” збільшуємо на одиницю. Зауважимо, що на початку роботи програми лічильнику потрібно присвоїти значення 0 (n := 0).

Блок-схема алгоритму:

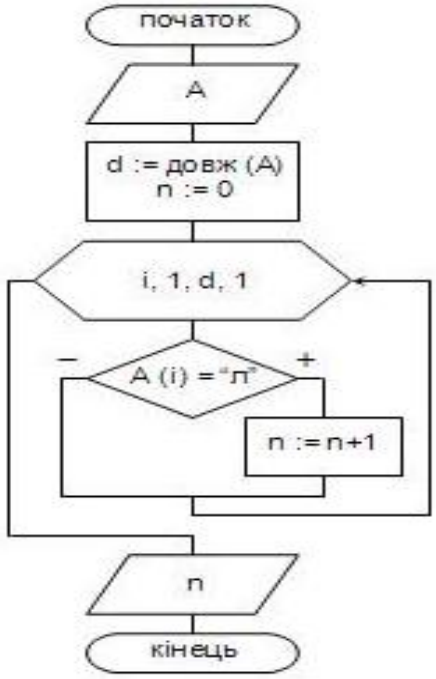


Рисунок 1 – Алгоритм знаходження літер

Приклад 2. У рядку символів видалити всі пропуски.

Вводимо даний рядок символів як значення змінної A. Результуючий рядок сформуємо як значення нової змінної B. Процес утворення результуючого рядка реалізуємо так: визначимо довжину даного рядка; використаємо цикл, за допомогою якого послідовно братимемо по одному символу і, якщо взятий символ не пропуск, то "приклеїмо" його справа до поточного значення створюваного рядка.

Блок-схема алгоритму:

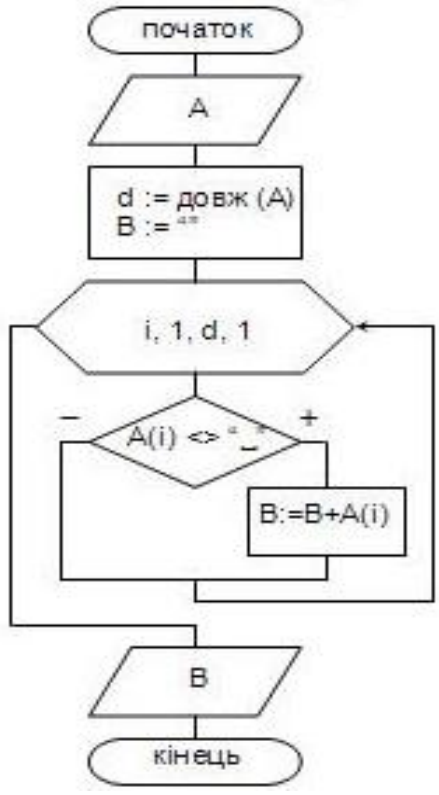


Рисунок 2 – Алгоритм знаходження пропусків

Записи. Файли.

Запис складається зі скінченної кількості елементів, що називаються полями, кожний з яких може бути різного типу. Цим запис відрізняється від масиву, всі компоненти якого повинні бути однотиповими.

Файл - це структура даних, яка складається з довільного числа елементів одного типу. На відміну від масивів число компонентів файлів, що називається довжиною файлу, при його означенні не задається.

Множини

Множини - це сукупність довільної кількості неупорядкованих елементів одного типу. Множина може вміщувати елементи описаного типу, всі підмножини описаної множини та пусту множину.

Множини можна порівнювати з допомогою операцій >, >=, <, <=, =, <>. Логічний вираз A1=A буде істинним, якщо A1 і A мають набір одних і тих же елементів; A1<=A буде істинним, якщо A1 є підмножиною A. Наприклад, нижченаведені вирази будуть мати значення TRUE:

```
     [3,4,5]    =     [3..5];
['0','1','2']   <=    ['0'..'9'];
       [4]      <>    [4,5];
[crw,bil]       >=    [ ].
```

Всі елементи множини повинні бути одного і того ж перелічувального типу. **Сталі множини задаються переліком їх елементів:**

```
[1, 2, 3, 4, 5];
[A, B, C];
[];
[1..N];
```

в квадратних дужках можуть бути як сталі, так і вирази, що складаються з елементів того самого типу, що елементи множини.

Змінні множини описуються таким чином

VAR ім'я : SET OF базовий тип

Digits = set of 0..9; {тип множини цифр }
Letters = set of 'A'..'Z'; {тип множини латиниці}

Розглянемо декілька прикладів задання множин:

Set Of 'A'..'Z'	Множина прописних англійських літер;
Set Of 1..100	Множина цілих чисел;
Set Of Chr	Множина всіх символів.

Відношення між множинами та операції над множинами

Відношення	
Належність	x in A
Рівність	A = B
Включення	A < B
Не строге включення	A x чисел;
Set Of Chr	Множина всіх символів.

Операції	
Об'єднання	A + B
Перетин	A*B
Різниця	A — B

ЗАВДАННЯ

13.Вивчити теоретичні відомості до лабораторної роботи.
14.Вибрати завдання на лабораторну роботу згідно варіанту.
15.Розробити алгоритм розв'язку у текстовому вигляді.
16.Скласти схему алгоритму розв'язку задачі.
17.Розробити систему контрольних прикладів і перевірити роботу алгоритму.
18.Зробити висновки.

ВАРІАНТИ ЗАВДАНЬ

Розробити алгоритм і блок-схему задачі. Оцінити час виконання та складність алгоритму.

Варіант	**Завдання**
1	Дано символьний рядок. Замінити всі символи '!' крапками, крім першого й вивести отриманий рядок.
2	Дано символьний рядок. Замінити всі послідовності символів 'on' на 'online' і вивести новий рядок.
3	Дано символьний рядок. Слово - послідовність символів між пробілами, не утримуючі пробіли усередині себе. Визначити кількість слів у даному рядку.
4	Дано символьний рядок. Замінити символи 'ing' на 'ed' і вивести отриманий рядок.
5	Створити блок-схему, яка для заданого користувачем речення підраховує, скільки в ньому голосних букв і розділових знаків.
6	Заданий рядок символів, підрахувати кількість букв «а» і «б» та визначити яких більше.
7	У заданому рядку символів замінити всі крапки знак оклику, а коми - тире.
8	Створити блок-схему, яка у заданому рядку символів підраховує кількість букв «v» і «b».
9	У заданому рядку символів замінити всі букви «а» на «с».
10	Із заданого рядка символів видалити всі крапки.
11	Визначити кількість розділових знаків у заданому рядку символів.
12	Дано речення, що складається зі слів, розділених будь-якою кількістю пробілів. Створити блок-схему, що редагує речення, залишаючи між словами тільки по одному пробілі.
13	У заданому рядку символів видалити всі крапки і подвоїти всі пробіли.
14	Дано символьний рядок. Замінити всі символи '!' крапками, крім першого й вивести отриманий рядок.
15	Дано символьний рядок. Замінити всі послідовності символів 'on' на 'online' і вивести новий рядок.
16	Дано символьний рядок. Слово - послідовність символів між пробілами, не утримуючі пробіли усередині себе. Визначити кількість слів у даному рядку.
17	Дано символьний рядок. Замінити символи 'ing' на 'ed' і вивести отриманий рядок.
18	Створити блок-схему, яка для заданого користувачем речення підраховує, скільки в ньому голосних букв і розділових знаків.
19	Заданий рядок символів, підрахувати кількість букв «а» і «б» та визначити яких більше.
20	У заданому рядку символів замінити всі крапки знак оклику, а коми - тире.
21	Створити блок-схему, яка у заданому рядку символів підраховує кількість букв «v» і «b».

22	У заданому рядку символів замінити всі букви «a» на «c».
23	Із заданого рядка символів видалити всі крапки.

КОНТРОЛЬНІ ПИТАННЯ

1. Що таке множина?
2. Як зберігається множина в пам'яті ЕОМ? Який максимальний обсяг оперативної пам'яті можна відвести під зберігання однієї множини?
3. Які операції можна виконувати над множинами?
4. Як додати елемент в множину?
5. Як видалити елемент з множини?
6. Як вивести елементи множини? Як підрахувати кількість елементів у множині?

Рекомендована література

1. Ахо А., Хопкрофт Д., Ульман Д. Структуры данных и алгоритмы: Пер. с англ. – М.: Изд. дом “Вільямс”, 2001. – 384 с.
2. Вирт Н. Алгоритмы и структуры данных: Пер. с англ.-2-е изд., испр. - СПб: Невский Диалект, 2001.-352 с.
3. Вирт Н. Алгоритмы и структуры данных. — М.: Мир, 1989. — 360 с.
4. ГОСТ 19.701-90. ЕСПД. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения. – Введ 01.01.92. – М.: Изд-во стандартов, 1991. – 25 с.
5. Гудман С., Хидетниеми С. Введение в разработку и анализ алгоритмов. – М.: Мир, 1981. – 368 с.
6. Кнут Д. Искусство программирования, т. I. Основные алгоритмы, 3-е изд. — М.: "Вильяме", 2000.
7. Колодницький М. М. Технічне та програмне забезпечення комп’ютерних інформаційних технологій: Навч. посібник. – Житомир: ЖІТІ, 1995. – 231 с.
8. Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы: построение и анализ. – М.: МЦНМО, 2000. – 960 с.
9. Овсяк В. Алгоритми: аналіз методів, алгебра впорядкувань, моделі, моделювання. – Львів, 1996. – 132 с.
10. Спиричева Н.Р. Структура данных и основные алгоритмы: Учебное пособие. - Екатеринбург: ГОУ ВПО УГТУ-УПИ, 2004. - 92 с.
11. Шаховська Н.Б., Голощук Р.О. Алгоритми та структури даних. – Львів: Магнолія-2006. – 2009. – 216 с.
12. Алгоритмы и программы решения задач на графах и сетях./ Нечепуренко М.И., Попков В.К. и др. - Новосибирск: Наука. Сиб.отделение, 1990. - 515 с.
13. Брой М. Информатика. Теоретическая информатика, алгоритмы и структуры данных, логическое программирование, объектная ориентация: В 4-х частях. Ч.4/ Пер.с нем. - М.:Диалог-МИФИ, 1998. - 224 с.
14. Евстигнеев В.А. Применение теории графов в программировании/ Под ред. А.П.Ершова. - М.: Наука. Гл. Ред. ФМЛ, 1985. - 352 с.
15. Кондратьева С.Д. Введение в структуры данных: лекции и упражнения по курсу. – М.: Изд-во МГТУ им. Н.Э.Баумана, 2000. – 376 с.
16. Костюк Ю.Л. Основы алгоритмизации: Учебное пособие. - Томск: Изд.ТГУ, 1996. - 124 с.
17. Кристофидес Н. Теория графов. Алгоритмический подход. – М.: Мир, 1978. – 432 с.
18. Орлов В.А. Теория графов и комбинаторика: Учебное пособие. - Томск: Изд.ТПИ, 1988. - 96 с.
19. Райли Д. Абстракция и структуры данных: Вводный курс: Пер.с англ. - М.: Мир, 1993. - 752 с.
20. Свами М., Тхуласираман К. Графы, сети и алгоритмы. – М.: Мир, 1984. – 454 с.
21. Трамбле Ж., Соренсон П. Введение в структуры данных: Пер.с англ. - М.: Машиностроение, 1982. - 784 с.
22. Харари Ф. Теория графов. – М.:Мир, 1973. – 300 с.