Практична робота 02. Розробка алгоритму виконання процесу

Мета:
1) навчитися робити декомпозицію задачі;
2) навчитися надавати в графічній формі схему алгоритму виконання задачі.

Постановка задачі.

Потрібно створити додаток/застосунок, який забезпечує введення, накопичення, коригування та оброблення інформації по викладачам і студентам гіпотетичного навчального закладу, а також пошуку інформації по викладачах та студентах. Введення може виконуватися в інтерактивному режимі або з файлу. Оброблення передбачає формування списків студентів по групах, викладачів по групах та предметах, звітності з успішності.

Завдання практичної роботи

1. Повторити матеріал лекцій №2 та 3, та ознайомитися з теоретичним матеріалом, наданим до цієї лабораторної роботи
2. Визначити перелік питань, які з вашої точки зору потребують уточнення для успішного розроблення алгоритму.
3. Провести функціональну декомпозицію задачі, накреслити схему.
4. Розробити блок-схему алгоритму виконання процесу як послідовності дій для отримання результату. Таким чином буде показана узагальнена схема технологічного процесу, який має реалізовувати додаток, тобто схему послідовності дій, які повинні виконуватися в ході вирішення задачі.
5. Розробити та накреслити блок-схему розрахунку коренів квадратного рівняння, де враховується багаторазове введення коефіцієнтів, аналіз дискримінанта, наявність двох, одного кореня та відсутність рішення, виведення результату.
6. Схеми можна виконувати в будь-якому доступному засобі, або накреслити на папері і зробити фотографію. Включити скріншот або фотографію (jpg-файл) в файл Word з практичною роботою/

   Звіт з виконання практичної роботи повинен містити:
   Титульний лист
   Завдання (виділено блакитним)
   Звіт (безпосередньо виконання завдання)

Теоретичні відомості.

Декомпози́ція — науковий метод, що використовує структуру завдання і дозволяє замінити вирішення одного великого завдання рішенням серії менших завдань, нехай і взаємопов'язаних, але більш простих. Декомпозиція, як процес розділення, дозволяє розглядати будь-яку досліджувану систему як складну, що складається з окремих взаємопов'язаних підсистем, які, в свою чергу, також можуть бути розділеними на частини. Як системи можуть виступати не тільки матеріальні об'єкти, а й процеси, явища і поняття.

Під декомпозицією розуміється розбиття **задачі** на відносно незалежні частини (**підзадачі**). **Декомпозиція задачі** може бути проведена кількома

способами: за завданнями, за даними, з інформаційних потоків. **Декомпозиція** за завданнями (функціональна **декомпозиція**) припускає присвоєння різним потокам різних функцій.

Вихідна система розташовується на нульовому рівні. Після її розділення виходять підсистеми першого рівня. Розділення цих підсистем або деяких з них призводить до появи підсистем другого рівня і т. д. Спрощене графічне представлення декомпозиційованої системи називається її ієрархічною структурою.

Для більшості завдань алгоритми їх вирішення є досить великими і громіздкими. При програмуванні потрібно намагатися отримати програму легку для читання, високоефективну і легко модифікуються. Для цього проводять декомпозицію складного алгоритму поставленого завдання, тобто його розбивка на окремі більш прості підзадачі, потім декомпозиція підзадач і т.д. Для цього використовують прийоми процедурного програмування.

Один з основних прийомів - розбивка алгоритму на окремі *функції* і / або *модулі*, Використовуючи функціональну і / або модульну декомпозиції відповідно.

Функціональна декомпозиція - метод розбивки великої програми на окремі *функції*, Тобто загальний алгоритм - на окремі кроки, які потім і оформляють у вигляді окремих функцій.

Алгоритм декомпозиції можна представити таким чином:

- Програму робити як послідовність більш дрібних дій;
- Кожну деталізацію детально описати;
- Кожну деталізацію представити у вигляді абстрактного оператора, який повинен однозначно визначати потрібну дію, і в кінцевому підсумку ці абстрактні дії заміняться на групи операторів вибраної мови програмування.

При цьому треба пам'ятати, що кожна деталізація - це один з варіантів вирішення, і тому необхідно перевіряти, що:

- Рішення приватних завдань призводить рішенням спільної справи;
- Обрана послідовність дій розумна;
- Побудована декомпозиція дозволяє отримувати команди, легко реалізовані на обраною мовою програмування.

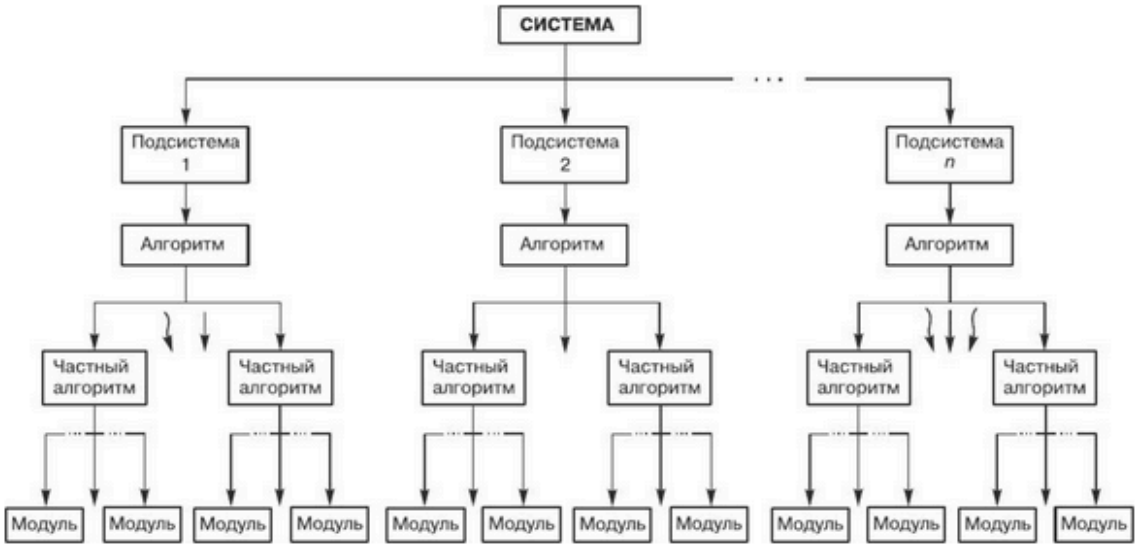


Рисунок 1 - Схема проведення декомпозиції

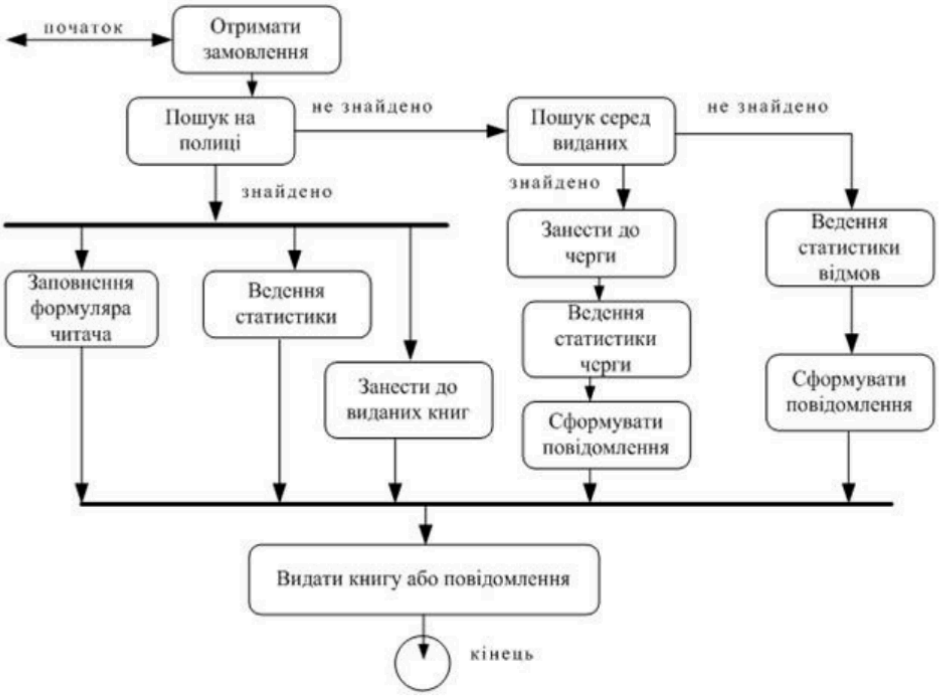


Рисунок 2 - Приклад декомпозиції процесу пошуку книги в бібліотеці по замовленню.

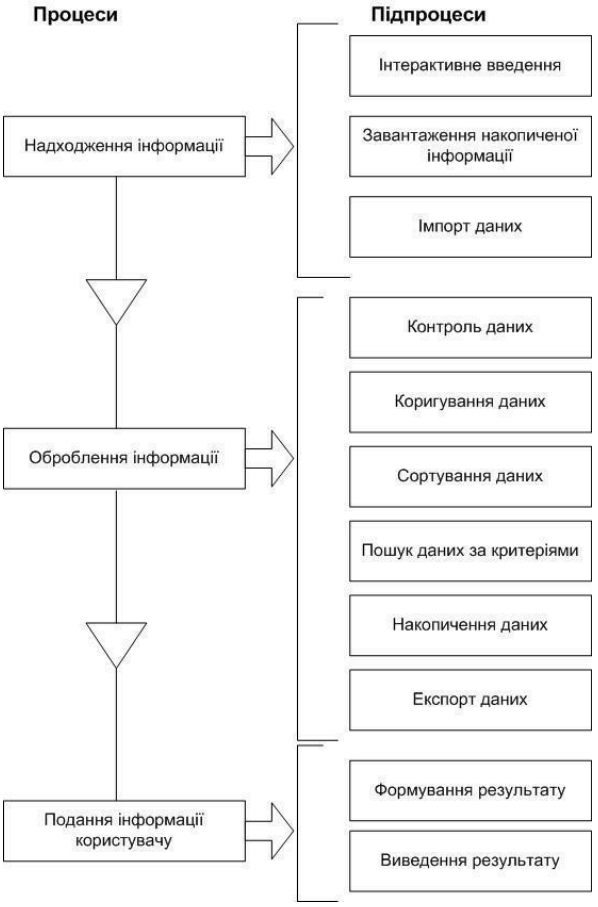
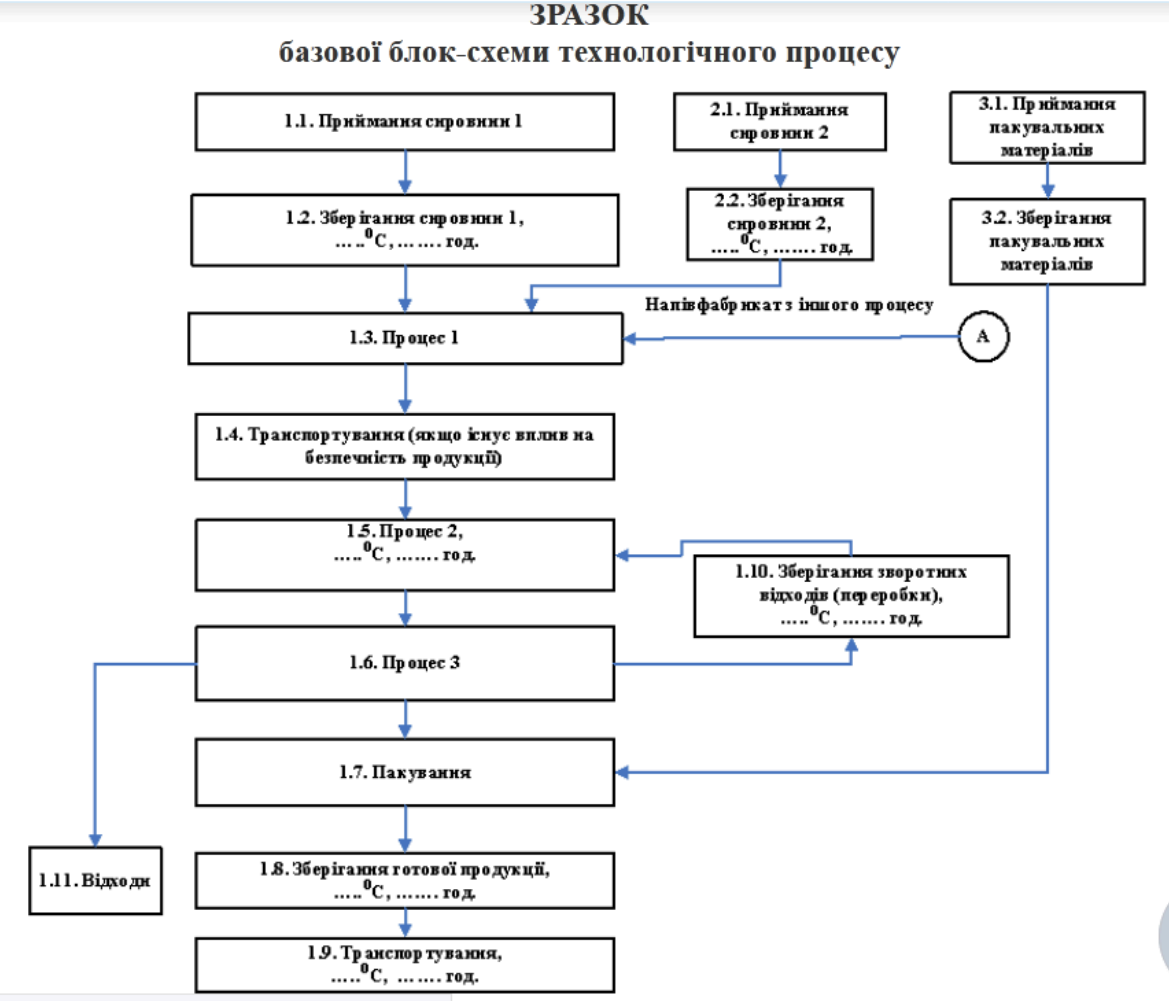


Рисунок 3 – Приклад декомпозиції (узагальнений випадок)



Контрольні запитання.

1. Дайте визначення алгоритму та перелічите його властивості.
2. Як можна визначити виконавця алгоритму та систему його команд?
3. Перелічите правила побудови алгоритмів.
4. Які види алгоритмів ви можете визначити?
5. В який спосіб можна записати алгоритм?
6. Які позначки використовуються при запису блок-схем?
7. Яким чином програма перетворюється в машинні коди?
8. Дайте визначення алгоритмізації.
9. Які процеси та етапи алгоритмізації?
10. В чому полягає декомпозиція?
11. Коли застосовується інтеграція?
12. Наведіть приклади алгоритму та алгоритмізації в повсякденному житті.