ЗАВДАННЯ

[Для попередньої інформації можна переглянути будову словника даних прогнозу погоди, яка викладена в файлі "Задачі для словників".]

Мета завдання.
1. Реалізувати приклад макета окремих частин проєкта, який можна використати як представлення пропозиції (реклама) чи уточнення постановки задачі для розробки. Оцінити окремі кроки розробки проєкта.
2. Використати структуру даних dict (словник) для отримання досвіду програмування операцій з словниками практичного наповнення.

Загальні кроки реалізації задачі.
1.Отримати з стороннього джерела дані про погоду.
2.Скласти перелік потенційних задач опрацювання даних. Реалізувати кожну задачу програмно.
3.Визначити спосіб зберігання результатів. Реалізувати збереження і передавання результатів обчислень.
4.Укласти інструкцію користувача програмного продукту.

Схема реалізації.
1.Для макета реального проєкта замість отримання даних з стороннього джерела можна вводити дані з текстового файла. Для цього треба, насамперед, точно визначити формат такого файла. А перед тим, як визначати формат, треба обрати перелік потрібних даних. Наприклад, перелік можна взяти подібно до зазначеної задачі на початку.

1.1.Записати перелік показників погоди, які ви вважаєте за потрібне: країна, місто, час отримання даних, напрям вітру, температура, атмосферний тиск тощо. [Цей перелік в майбутньому може бути переглянутий за побажаннями користувача, тому передбачити можливість неруйнівної модифікації.]

1.2.Точно визначити формат переліку показників, наприклад:

```
suite ::=   info+   NEWLINE
info  ::=   city  delim date delim  time delim  temperature delim  pressure delim
            wind  NEWLINE
city  ::=   "Назва міста"
date  ::=   "month"  ":"  "date"  |  "today"
temperature  ::=  "integer number"
pressure   ::=   "integer number"  |   "none"
wind   ::=   "N"  |  "NE"  |  "E"  |  "SE"  |  "S"  |  "SW"  |  "W"  |  "NW"  |  "none"
delim  ::=   TAB+  |  "<->"  |  "#"
```

Окремо вирішити питання пробілів в тексті показників.

[Факультативно. Задача для активних розробників: що робити, якщо формат вхідних даних не дотримано?]

1.3. Підготувати один конкретний варіант тестового файла вхідних даних. Можна "вручну" пошукати метеопоказники за сайтами погоди – щоб відібрати тестові дані. [Доведеться трохи попрацювати ☺]. Варто все-таки обирати реалістичні дані, а не довільні, щоб можна було візуально оцінити можливості задач опрацювання даних показників погоди.
Для надійного тестування нашої програми, а також для справляння більшого позитивного враження про нашу програму, кількість даних має бути достатньою. Можна для початку записати в тестовий файл вхідних даних 15-20 показників.

1.4.Визначити структуру словника dict для відображення показників погоди у внутрішню пам'ять. Скласти перелік ключів і значень. Ключі не обов'язково мають дублювати перелік записаних вище показників. Наприклад, за ключем date можна побудувати вкладений словник за ключем time. Таке проектування полегшить пошук потрібних даних в словнику.
Варто мати на увазі, що вдале проектування словника автоматично спрощує будову функцій опрацювання словника, і потребує уважного вивчення.

1.5.Скласти функцію читання даних з файла і збереження даних в словнику. [В майбутньому реальному проєкті таку функцію можна перемкнути від тестового файла до сервера метеоданих, з яким хоче співпрацювати користувач нашого програмного продукту]

2.1.Приклад переліку потенційних задач опрацювання метеоданих:
- просто відобразити матеодані заданого міста; було б непогано мати можливість вибору формату відображення;
- в якому місті найвища/найнижча температура;
- динаміка зміни температури в зазначеному місті протягом дня;
- який переважаючий напрям вітру в списку визначених міст;
- де температура менша від t – тобто, де настало похолодання.

Очевидно, що подібних задач може бути дуже багато.
З огляду на потенційні задачі варто повернутись до кроку 1.4 і за потреби внести корекцію структури словника.

2.2.Програмна реалізація задач.
На початку кожну задачу варто реалізувати окремою функцією (в майбутньому вона може стати методом класу – якщо буде сенс будувати клас). Кожна функція може мати необов'язковий параметр (який?) і спосіб повернення результатів. Результати в загальному випадку можна повертати оператором return (технологія процедурної декомпозиції), або записом в спільну ділянку пам'яті (технологія ООП чи розподілених обчислень). Варто передбачити обидва варіанти.
Переконливо рекомендуємо супроводжувати текст кожної функції розширеними коментарями.

2.3.Виконати внутрішнє незалежне тестування кожної функції. Дуже сподіваємось, що нам дещо відомо про технологічні прийоми тестування функцій.

3.Спосіб зберігання результатів.
Для макета реального проєкта результати можна записати в текстовий файл. [В майбутньому замість текстового файла дані можна надсилати користувачам наприклад, за електронною адресою, чи надавати доступ до результатів.]
Отже, треба скласти функцію вибору обчислених даних і перетворення до потрібного формату, який хотів би бачити користувач. [Формат визначаємо подібно до п.1.2]. Реалізувати функцію. Виконати тестування функції.

4.Інструкція користувача.
Це є звичайний файл текстового формата. Має бути опис призначення програмного продукту, перелік функціональних можливостей, вказівки до виконання функцій програми. Зважити, що інструкцію пишуть для різних категорій користувачів і термінологія інструкції має бути зрозумілою для всіх. Не можна вживати терміни з програмування, а тим більше передбачати виправлення програми користувачем.

Результати проєкту надіслати в чотирьох файлах: інструкція користувача; тестовий файл вхідних даних; файл програмної реалізації (єдиний!); приклад файла результатів.