МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«Київський політехнічний інститут імені Ігоря Сікорського»

ПРОГРАМУВАННЯ НА МОВАХ С ТА С++

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт з дисципліни «Основи програмування» для студентів спеціальності 121 Інженерія програмного забезпечення спеціалізації Програмне забезпечення комп'ютерних та інформаційно-пошукових систем

Ухвалено
Вченою радою ФПМ НТУУ «КПІ»
(___.___.201_ р., протокол № __)

Київ
НТУУ «КПІ»
2017

Програмування на мовах С та С++: методичні вказівки до виконання лабораторних робіт з дисципліни «Основи програмування» для студентів спеціальності 121 Інженерія програмного забезпечення спеціалізації Програмне забезпечення комп'ютерних та інформаційно-пошукових систем [Електронне видання] / Р. А. Гадиняк. – К. : НТУУ «КПІ», 201_. – __ с.

Навчально-методичне видання

ПРОГРАМУВАННЯ НА МОВАХ С ТА С++ МЕТОДИЧНІ ВКАЗІВКИ ДО ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ З ДИСЦИПЛІНИ «ОСНОВИ ПРОГРАМУВАННЯ» ДЛЯ СТУДЕНТІВ СПЕЦІАЛЬНОСТІ 121«ІНЖЕНЕРІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ» СПЕЦІАЛІЗАЦІЇ «ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ КОМП'ЮТЕРНИХ ТА ІНФОРМАЦІЙНО-ПОШУКОВИХ СИСТЕМ»

Методичні вказівки розроблено для ознайомлення студентів з теоретичними відомостями та практичними прийомами програмування на мовах С та С++, а також вимогами до виконаних лабораторних робіт, зокрема правилами їх оформлення. Навчальне видання призначене для студентів, які навчаються на спеціальності 121 Інженерія програмного забезпечення спеціалізації Програмне забезпечення комп'ютерних та інформаційно-пошукових систем факультету прикладної математики НТУУ «КПІ».

Автор: асистент *Гадиняк Руслан Анатолійович*
асистент *Комісар Дмитро Олександрович*

Відповідальний за випуск *Гадиняк Руслан Анатолійович*

Рецензент *Дичка Іван Андрійович*, доктор техн. наук, проф.

ЗМІСТ

ВСТУП

Особливістю мов програмування С та С++ є те, що їх відносять до так званих мов програмування середнього рівня, тобто вони поєднують переваги високорівневих мов в організації алгоритмів та структур даних, водночас не втративши можливості напряму керувати пам’яттю, працювати з вказівниками, з структурами даних, отримуючи доступ до даних безпосередньо через адреси комірок. Ці особливості роблять їх ідеальними для знайомства з програмуванням як таким, водночас залишаючи необмежені простори для подальшого розвитку. Водночас дуже велика кількість сучасних мов програмування мають Сі-подібний синтаксис.

Основною рекомендацією є використання з перших кроків операційної системи Linux, і, відповідно компілятора GCC. Це дасть змогу орієнтуватись відразу на корпоративний сектор комп’ютерних систем, де саме панують операційні системи Unix-сімейства.

У даних методичних вказівках розглядаються основні принципи та практичні прийоми розроблення програм на мовах С та С++. Методичні вказівки вміщають досить велику кількість завдань, що пропонується виконати в рамках дисципліни «Основи програмування», яка входить до складу циклу загальних дисциплін навчального плану підготовки бакалаврів спеціальності 121 Інженерія програмного забезпечення спеціалізації Програмне забезпечення комп’ютерних та інформаційно-пошукових систем.

В кожному розділі надаються теоретичні відомості з певної теми, завдання на лабораторну роботу з цієї теми, вказівки щодо виконання завдання, а також наводяться вимоги до оформлення звіту з виконаної лабораторної роботи, контрольні питання для самоперевірки та список рекомендованої літератури.

Лабораторні роботи з дисципліни «Основи програмування» розраховані на 72 академічні годин аудиторних занять.

Загальні вимоги до оформлення звіту з лабораторних робіт

Лабораторна робота має бути подана в електронному вигляді.

Електронна версія зберігається в банку даних кафедри програмного забезпечення комп'ютерних систем ФПМ НТУУ "КПІ". Файл з копією лабораторної роботи здається на кафедру безпосередньо під час захисту. Формат файлу *.docx або *.pdf.

Основний текст звіту має бути набраний з дотриманням таких вимог:

- шрифт Times New Roman 14 пт;
- відступ першого рядка 12.5 мм;
- міжрядковий інтервал 1.5;
- вирівнювання по ширині;
- поля: верхнє та нижнє – 20 мм; ліве – 30 мм; праве – 15 мм;
- від краю до верхнього/нижнього колонтитула 12.5 мм.

Текст в таблицях має бути набраний з дотриманням таких вимог:

- шрифт Times New Roman 12 пт;
- міжрядковий інтервал 1.0;

Текст програм має бути набраний з дотриманням таких вимог:

- шривт Couries New 8 пт;
- міжрядковий інтервал 1.0;

Всі рисунки повинні мати підрисунковий напис. Напис вирівнюється по центру і починається зі скорочення "Рис.", потім ставиться пробіл та порядковий номер рисунку. Після номера рисунка ставиться крапка, пробіл та пишеться назва рисунка.

Першою сторінкою звіту з лабораторної роботи є оформлений за зразком титульний аркуш.

Лабораторна робота №1

тема “Математичні оператори умови та цикли”

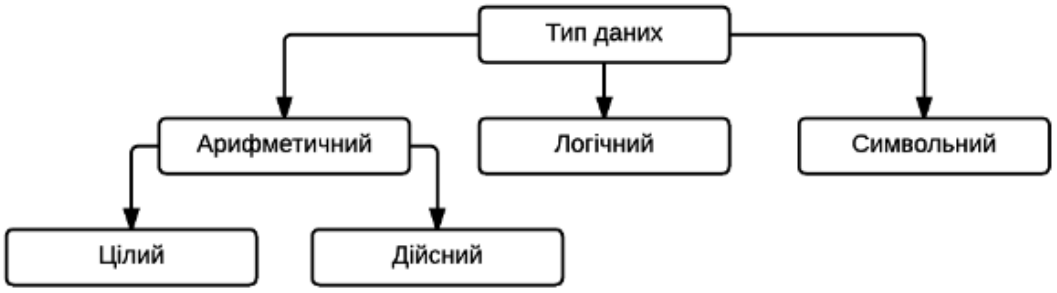
Мета роботи – Навчитися на практиці проводити точні обчислення математичних формул за допомогою операторів та функцій мови програмування С. Застосувати теоретичні знання для створення програмного забезпечення для прийняття рішень на основі вхідних даних за допомогою умовних конструкцій та конструкцій розгалуження. Застосувати на практиці різні види циклічних конструкцій для обчислення математичних формул. Навчитися виконувати компіляцію та базове тестування власного коду за допомогою компілятора мови С та спеціалізованої IDE.

Короткі теоретичні відомості

Тип даних - це характеристика, яку явно або неявно надано об’єкту (змінній, функції, константі, масиву, тощо).

Тип даних визначає:

- множину допустимих значень;
- формат збереження даних;
- розмір виділеної пам’яті;
- набір операцій, які можна виконувати над даними.



Прикладами типів даних є:

- Прості типи даних
    - цілі числа
    - дійсні числа
    - комплексні числа
    - символи
    - логічні (булеві) типи даних
- Складні типи даних
    - рядки (текст) - складається із символів
    - вектори - набори однотипових даних
    - структури - складаються із будь-яких інших типів даних

Мови програмування мають свій набір визначених типів, а також можуть надавати програмістам можливість створювати нові типи даних.

Умовні конструкції (рішення) - спеціальні точки алгоритмів, у яких можливе розгалуження потоку виконання операцій. Умовна конструкція містить певну умовну операцію, в залежності від результату виконання якої потік може розгалузитись на 2 або більше потоків.

Цикл - будь-яка багатократно виконувана послідовність команд, організована будь-яким чином. Термін вважається введеним Адою Лавлейс.

Послідовність інструкцій, призначена для багаторазового виконання, називається **тілом циклу**.

Одноразове виконання тіла циклу називається **ітерацією**.

Вираз, що визначає чи буде вкотре виконуватися ітерація, чи цикл завершиться, називається ***умовою виходу*** або ***умовою завершення циклу*** (або ***умовою продовження*** в залежності від того, як інтерпретується його істинність — як ознака необхідності завершення чи продовження циклу.

Завдання на виконання лабораторної роботи

Дано три завдання. Кожне завдання виконувати у окремій програмі, виводити всі результати у консоль та успішно завершувати виконання програми. Файли з кодом програм можуть бути розміщені у різних піддиректоріях (наприклад, task1/main.c, task2/main.c, task3/main.c). Код повинен успішно компілюватися у програму для запуску у ОС Linux за допомогою команди:

```
gcc main.c -std=c11 -Werror -pedantic-errors -lm
```

(можна додавати -lprogbase якщо ви використовуєте бібліотеку libprogbase). Перед запитом даних у користувача обов'язково повідомляти його про те, що і для чого потрібно ввести.

Завдання №1. Математичні операції

Виконати розрахунки за заданими формулами.

$$a = a_0 + a_1 + a_2,$$

де

$$a_0 = \frac{x^{y+1}}{(x-y)^{\frac{1}{z}}},$$

$$a_1 = \frac{y}{2 \times |x+y|},$$

$$a_2 = (x+0)^{\frac{1}{|sin(y)|}}$$

Вивести значення x,y,z,a0,a1,a2 та a у консоль, наприклад, так:

```
x = 0.2
y = -12.5
z = 1.000
a0 = 15.565
a1 = -45.000
a2 = 998.121
a = 1001.605
```

Якщо деякі зі значень неможливо обчислити (наприклад, відбувається ділення на 0), замість результату виводити повідомлення Can't be computed, наприклад (числа і результат видумані).

Завдання №2. Умовні конструкції

Користувач вводить 3 цілочисельні значення a, b і c.
На основі вхідних даних і заданих умов за допомогою умовних конструкцій (if...else) визначити істинність результату і вивести його у консоль. Значення істинності зберігати у змінну у кожній з віток умовних конструкцій, результат виводити після всіх умов 1 раз:

- Якщо всі введені числа a, b та c від’ємні, то:
  Знайти значення modmin: модуль найменшого числа
  Знайти значення sum2: суму двох інших чисел
  Вивести строку All negative та значення modmin і sum2 у консоль.
  Результат: True якщо модуль sum2 більший за modmin і значення sum2 більше -256; (True або False)
- Якщо серед введених чисел є і від’ємні і додатні (або 0), то:
  - Якщо тільки одне з них від’ємне, то:
    Вивести строку One negative і значення від'ємного числа у консоль
    Результат: True якщо це число парне, False - якщо непарне.
  - Якщо два від’ємних числа, то
    Вивести строку Two negative і значення обох від'ємних чисел у консоль
    Результат: чи сума цих двох від’ємних чисел помножена на 3 більша за -256; (True або False)

- Якщо всі числа додатні (або 0) то:

  Знайти значення max: максимальне з цих чисел

  Знайти значення min: мінімальне з цих чисел

  Вивести строку All positive та значення max і min у консоль.

  Результат: чи max - min більше 32; (True або False)

Результат виводити разом з вхідними значеннями a, b і c, наприклад (результат видуманий):

```
a = -15
b = 35
c = 96
Result = True
```

Завдання №3. Циклічні конструкції

Користувач вводить 2 додатні цілочисельні значення n і m. За допомогою ітераційних циклічних конструкцій (for) обчислити значення x та вивести його консоль, наприклад:

$$x = \sum_{i=1}^{n} \sum_{j=1}^{m} \left(\frac{1}{i+j} + i \times (j - 0) \right)$$

Вивести у консоль значення введених n і m, після цього на кожній ітерації внутрішнього і зовнішнього циклу виводити поточні значення лічильників і та j та проміжні суми, а у кінці вивести значення x.

Наприклад (числа видумані):

```
n = 2
m = 3
-----------------------------------
i = 1, j = 1: sum1 = 0,     sum2 = 45.6
```

```
i = 1, j = 2: sum1 = 0,     sum2 = 51.6
i = 1, j = 3: sum1 = 0,     sum2 = 59.9
i = 2, j = 1: sum1 = 59.9, sum2 = 23.12
i = 2, j = 2: sum1 = 59.9, sum2 = 99.54
i = 2, j = 3: sum1 = 59.9, sum2 = 111.23
-----------------------------------
x = 161.65
```

Контрольні питання

1. Що таке змінна? Як оголошуються змінні у мові С?
2. Що таке тип даних? Які базові типи даних існують у мові С?
3. Що таке цикли? Які види циклічних конструкцій є у мові С?

Лабораторна робота №2

тема “Масиви даних та строки”

Мета роботи – навчитися працювати зі масивами різних типів даних мови програмування С. Застосувати на практиці різні види циклічних конструкцій при роботі з одномірними та багатомірними масивами даних. Вдосконалити вміння роботи з рядками у мові С. Навчитися оформлювати консольну програму для зручності роботи користувача.

Короткі теоретичні відомості

Масив - це тип даних, що складається із розміщеного поряд у пам’яті непустого набору об’єктів певного типу. Кількість цих елементів не змінюється протягом часу існування масиву.

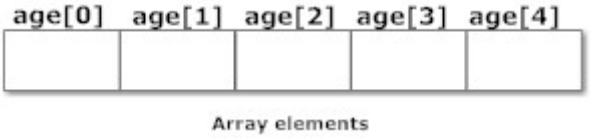


Рис. Графічне представлення одномірних масивів

Масиви із одним розміром є **одновимірними** масивами, із двома - **двовимірними**, трьома - **тривимірними** і так далі. Масиви з більш ніж 1 виміром називаються **багатовимірними масивами**.

Символи у мові С представляються у вигляді 8-бітного коду із таблиці ASCII, а для їх зберігання використовуються змінні типу **char**.

Символи можуть належати деяким **класам символів**.

Рядки в С - це послідовність символів, що закінчується на нульовий термінальний символ ‘**\0’**. (null-terminated string).

Строкова послідовність символів - це нуль або більше символів кожен з яких є або багатобітовий символ (виключаючи **("), \, і символ переходу на новий рядок**) або символьна послідовність чи hex-, octal- escape.

Завдання на виконання лабораторної роботи

Дано три завдання. Всі завдання реалізувати разом у одній консольній програмі.

Після запуску програми користувач має можливість обрати завдання зі списку завдань і перейти у відповідний стан.

Завжди виводити інформацію про стан, у якому знаходиться користувач та надати спеціальну команду (або клавішу) для виклику допомоги (*help*) по поточному стану. Дати можливість користувачу за допомогою спеціальної команди (або нажаття спеціальної клавіші) перейти у початковий стан, у якому можна обрати інше завдання.

Розділити консоль у стані на дві "зони": **зону відображення** та **командну зону**.

У зоні відображення виводити об'єкт з яким працюватиме користувач у відповідному стані (наприклад, масив або рядок). При переході до кожного із станів одразу виводити об'єкт у зоні відображення.
У командній зоні дати користувачу можливість вводити команди та виводити результати їх виконання (наприклад, максимальне значення у масиві як результат відповідної команди).

Весь ввід користувача перевіряти на валідність (чи допустиме від'ємне значення або чи введене значення є індексом поточного масиву).

Підсвічувати фон шуканих елементів, або елементів, що змінюються за допомогою кольорів.

Завдання №1. Одномірний масив

У цьому стані у користувача є одномірний масив цілих чисел (розміром N елементів) ініціалізований нулями.

Доступні операції над масивом:

1. Заповнити масив випадковими числами від L до H.
2. Обнулити всі елементи масиву.
3. Знайти мінімальний елемент масиву та його індекс.
4. Знайти суму елементів масиву.
5. Вивести добуток від'ємних елементів масиву.
6. Поміняти місцями значення максимального і мінімального елементів масиву.
7. Зменшити всі елементи масиву на введене число.

Завдання №2. Двомірний масив

У цьому стані у користувача є двомірний масив цілих чисел (квардратна матриця NxN) ініціалізований нулями.

Доступні операції над матрицею:

1. Заповнити масив випадковими числами від L до H.
2. Обнулити всі елементи масиву.
3. Знайти максимальний елемент та його індекси (i та j).
4. Знайти суму елементів побічної діагоналі масиву.
5. Знайти суму елементів стовпця за заданим індексом.
6. Поміняти місцями максимальний і мінімальний елементи масиву.
7. Змінити значення елементу за вказаними індексами на задане.

Завдання №3. Обробка рядків мови С

У цьому стані у користувача є буфер (розміром N символів) для зберігання строки.
У початковому стані буфер повинен містити строку із видимих випадкових символів.
Біля виведеної строки завжди виводити її поточну довжину.

Доступні операції над рядком:

1. Заповнити строку введеним значенням із консолі (пробіли також мають працювати).
2. Очистити строку.
3. Вивести підстроку із заданої позиції і заданої довжини.
4. Вивести список підстрок, розділених заданим символом.
5. Вивести найкоротше слово (слова - непуста послідовність буквенних символів).
6. Знайти та вивести всі дробові числа, що містяться у строці.
7. Знайти та вивести добуток всіх цілих чисел, що містяться у строці

Контрольні питання

1. Що таке масиви даних? Як оголошуються масиви у мові С?
2. Як розміщуються у пам'яті елементи двомірного масиву?
3. Які класи символів існують у таблиці ASCII?
4. Що таке рядки у мові С?

Лабораторна робота №3

тема “Структури даних, функції, вказівники і файлові потоки”

Мета роботи – Навчитися працювати зі структурами, файлами, динамічно виділяти пам’ять, а також вдосконалити та закріпити вміння роботи з вказівниками.

Короткі теоретичні відомості

Структури даних призначені для об’єднання різних типів даних у новий тип даних.

Таким чином стає можливим створення змінних структурного типу, що міститимуть у собі декілька різнотипних значень.

Функція - це конструкція, що групує код для виконання певного завдання.

Також, функція - це підпрограма, що реалізує певний алгоритм обробки вхідних даних для отримання вихідного результату.

Функції обов’язково мають **назву функції** (ідентифікатор функції), і ця назва повинна відображати дію чи роботу, яку виконує функція.

Назва функції - це ідентифікатор, який повинен бути унікальним в межах програми.

У прикладі це привітання програми. Нехай назвою функції буде **programSayHello**.

Окрім назви і переліку команд, кожна функція мови С має мати **тип повертаємого значення** і **список параметрів**.

Тип повертаємого значення, назва функції і список параметрів функції разом описують **сигнатуру функції**.

Список параметрів функції розміщується після назви функції у парі круглих дужок.

Найпростіша з функцій нічого не повертає і не має списку параметрів. Тому замість них потрібно використовувати ключове слово **void**.

Завдання на виконання лабораторної роботи

Створити консольну програму для роботи зі списком сутностей типу **Музична композиція**.

Після входу у програму користувач має можливість зробити вибір:

- Створити новий пустий список сутностей.
- Зчитати список сутностей із текстового файлу.

Після цього список сутностей можна модифікувати:

- Додати у кінець списку нову сутність із вказаними користувачем даними
- Видалити сутність із вказаної позиції у списку
- Перезаписати всі дані полів сутності у вказаній позиції на нововведені користувачем (повне оновлення сутності)
- Перезаписати тільки обране поле даних сутності із вказаної позиції у списку (часткове оновлення сутності)
- Знайти і вивести К найдовших композицій. Вивести результат у консоль.
- Зберегти поточний список сутностей на файлову систему, запропонувавши користувачу ввести назву дискового файлу, у який збережуться всі дані.

Методичні вказівки до виконання

Кроки, що описані далі, є рекомендаціями щодо поступової розробки описаного завдання.

Список сутностей

Список сутностей рекомендується реалізувати як масив вказівників на структури фіксованої довжини (наприклад, 10, це потрібно захардкодити у деякій константі), що описують тип **Музична композиція**:

1. Якщо вказівник у комірці масиву містить значення NULL - комірка масиву вважається порожньою (пуста сутність).
2. У масиві вказівників всі непусті елементи мають бути на позиціях з початку масиву, всі пусті - у кінці масиву.
3. Розміром списку буде кількість непустих сутностей у масиві вказівників. Розмір самого масиву - це максимальний розмір списку сутностей.
4. При видаленні сутності з певної позиції всі сутності, що розташовані справа від неї зсувати на одну позицію вліво.
5. При додаванні нової сутності використовувати динамічне виділення пам'яті. При видаленні сутності звільняти динамічну пам'ять сутності. На виході із програми звільняти динамічну пам'ять всіх сутностей, що залишились у списку.

Крок №1. Розробка структур і функцій

1. Описати нову структуру даних типу **"Музична композиція"**. Визначити у структурі не менше 3-х полів різного типу, серед яких обов'язковими є:
    - строка
    - ціле число
    - дробове число
    - **структура іншого типу**
2. Виконати декомпозицію завдання на окремі функції і описати їх прототипи:

- Описати функції, одна з яких виконує запис даних із одної структури (через вказівник) у строку, та функцію, що на основі вхідної строки у певному текстовому форматі заповнює поля одної структури (через вказівник на неї).
- Описати функцію, що із тексту заповнює список сутностей та зворотню функцію, що зі списку сутностей заповнює строку у певному текстовому форматі.
- Описати функцію, що по індексу видаляє сутність зі списку, функцію, що виконує перезапис даних структури по індексу даними із іншої структури.
- Описати функцію, що отримує вказівник на структуру, строку, що позначає назву поля у структурі та строку, що позначає нове значення вказаного поля у структурі і виконує перезапис поля заданої структури новим значенням.
- Описати функцію, що приймає на вхід даний масив вказівників на структури і виконує дію:
  Знайти і вивести K найдовших композицій.
  Результат виконання дії записати у вихідний масив вказівників на структури.

Крок №2. Тестування чистих функцій

1. Протестувати всі чисті функції за допомогою assert() у окремій функції(-ях). Кожна функція має тестуватись мінімум на 5 тестових випадках.

Крок №3. Робота із файловими потоками

1. Реалізувати функцію для зчитування даних із текстового файлу і перетворення їх у список.
2. Реалізувати функцію для запису списку у текстовий файл.

Крок №4. Розробка CUI-інтерфейсу

1. Створити **CUI** (Console User Interface, консольний інтерфейс користувача) для використання описаного функціоналу (*це просто взаємодія з користувачем через консоль*).
2. Весь ввід користувача перевіряти на валідність щоби не допускати збоїв у роботі програми.
3. Виводити користувачу повідомлення про неможливість виконання операції:
    - при спробі додати нову сутність у заповнений список
    - при спробі видалення чи зміни сутності на пустій позиції
    - при спробі змінити поле сутності за назвою, якої не існує
4. Обов'язково перевіряти існування дискового файлу при спробі його відкрити і повідомляти користувачу про неможливість здійснення такої дії.

Замітки по кроках:

1. Описані функції (крок 1) - це не повний перелік функцій, що необхідні для виконання завдання. Вам потрібно створити додаткові функції. Старатись по максимуму виділяти нові чисті функції і тестувати їх (крок 2).
2. Описувати всі прототипи функцій на початку файлу main.c (після включень заголовочних файлів). Відділити коментарями прототипи функцій, що належать одній із трьох груп:

1. функції перетворення даних, роботи зі списком сутностей та взаємодії з файлами (крок 3).
2. тестові функції (крок 2).
3. функції для забезпечення CUI інтерфейсу (крок 4).

3. Запускати виконання тестових функцій (крок 2) тільки при заданні певного аргумента командного рядка (наприклад, ./a.out -t) у іншому випадку запускати CUI інтерфейс (крок 4).

Контрольні питання

1. Що таке структура даних? Як оголошуються структури даних у мові С? Які операції можна виконувати над змінними структур даних.
2. Як розміщуються поля структур даних у пам“яті?
3. Що таке функції? Що таке прототипи функцій? З чого складається сигнатура функції?
4. Що таке вказівник? Які операції можна виконувати над вказівниками?

Лабораторна робота №4

тема “Модульне програмування та динамічні структури даних”

Мета роботи – Навчитися створювати і використовувати динамічні структури даних. Навчитися розбивати програмний код на модулі і перевіряти їх роботу за допомогою модульних тестів.

Короткі теоретичні відомості

Модуль - це функціонально закінчений фрагмент програми, що оформлений у вигляді окремого файла з кодом, або іменованої частини коду і призначений для використання у інших програмах.

Зазвичай модулі проектуються таким чином, щоби надати програмістам зручну для багаторазового використання функціональність (**інтерфейс**) у вигляді набору об’єднаних спільною тематикою функцій, типів і тих даних, якими вони управляють.

Зручність використання модульної архітектури полягає у можливості оновлення чи заміни модуля програми без необхідності внесення змін у інші частини програми.

Модульна парадигма програмування: *визначте, які модулі потрібні, поділіть програму так, щоби дані були сховані у цих модулях*. Дана парадигма також відома як “**принцип приховування даних**”.

Модуль - програмна сутність, що складається із *інтерфейсу* і *реалізації*.

Інтерфейс (Interface) визначає *ЩО* модуль виконує. Інтерфейс оголошує ідентифікатори, типи і функції, що стають доступні зовнішньому коду, який використовує інтерфейс (клієнтський код).

Інтерфейс – сукупність можливостей, способів і методів одночасної дії (і обміну інформації) двох, що мають спільний розподіл (границю), тобто не зв'язаних лінійно, інформаційних систем, приладів чи програм, що

визначаються їх характеристиками, а також характеристиками з'єднання, сигналів обміну і т.п.

Інтерфейс розділяє *реалізацію* і *клієнта (користувача реалізації)*.

Обмін інформацією через інтерфейс може бути одностороннім і двостороннім.

Реалізація (Implementation) визначає *ЯК* модуль виконує оголошений функціонал, описаний у інтерфейсі.

У кожного модуля зазвичай один інтерфейс, але може бути багато реалізацій, що реалізують його можливості. Різні реалізації можуть використовувати різні алгоритми і типи даних для досягнення цілей інтерфейсу.

Модульність є механізмом для підвищення гнучкості і зрозумілості системи та скорочення часу на розробку.

Кожна задача формує окремий і виразний програмний модуль:

- Під час реалізації модулів їх точки входу і виходу чітко описані.
- В певний момент часу, такі модулі можуть бути протестовані незалежно один від одного.
- Виникнення помилок легше локалізувати на рівні модулів.
- Створення модуля не вимагає від розробника детальних і точних знань про внутрішню структуру (код) суміжних модулів. Це також дозволяє легко замінювати реалізації модулів у програмі на інші.
- Розбиття коду на модулі дозволяє працювати над програмою декільком незалежним розробникам чи групам розробників. Зміни коду одного модуля не впливають на роботу іншим модулів.

Завдання на виконання лабораторної роботи

(Модифіковане завдання із лабораторної роботи №3)

Створити консольну програму для роботи зі списком сутностей типу **Музична композиція**.

(Головне меню) Після входу у програму користувач має можливість зробити вибір:

- Створити новий пустий список сутностей (перейти до меню списку).
- Зчитати список сутностей із текстового файлу, назву якого вводить користувач.

(Меню списку) Після цього список сутностей можна модифікувати:

- (Стан створення) Обрати позицію вставки і перейти до створення нової сутності:
    - Заповнити дані всіх полів сутності
    - Зберегти зміни (зі вставкою у список) чи відмінити (і повенутися у попереднє меню)
- (Стан редагування) Перейти до редагування окремої сутності (нової або обраної зі списку):
    - Змінити дані обраного поля
    - Видалити обрану сутність зі списку
    - Зберегти або відмінити зміни (і повенутися у попереднє меню)
- Знайти і вивести K найдовших композицій. Результуючі елементи списку виділити (наприклад, фоном певного кольору).
- Зберегти поточний список сутностей на файлову систему, запропонувавши користувачу ввести назву дискового файлу, у який збережуться всі дані
- Повернутися у головне меню

Вимоги до реалізації

- Мова програмування: **C**.

- Всі списки, у яких зберігаються сутності повинні бути динамічними (реалізація зв'язаними нодами).
- Формат даних, у якому будуть зберігатися сутності на файловій системі повинен бути **CSV**.
- Весь код повинен бути розділений на модулі
- Всі чисті функції модулів мають бути протестовані за допомогою модульних тестів

Контрольні питання

1. Що таке абстракція? У чому принцип інкапсуляції?
2. Що таке модуль і як реалізувати модулі у мові програмування C?
3. Що таке неповний тип даних? Як можна використовувати неповні типи даних?
4. Що таке інтерфейс? Опишіть інтерфейс та принцип роботи динамічної структури даних типу Список. Які можливі реалізації цього типу даних?
5. Опишіть інтерфейс та принцип роботи динамічної структури даних типу Стек. Які можливі реалізації цього типу даних?

Лабораторна робота №5

тема “Програми із графічним інтерфейсом користувача”

Мета роботи – навчитися створювати віконні програми в ОС Linux за допомогою фреймворку Qt та редактора Qt Creator.

Короткі теоретичні відомості

Вікно (**Window**) - це прямокутна зона на екрані, що отримує ввід користувача і виконує вивід у вигляді тексту або графіки.

Кожне вікно ідентифікується назвою.

Більшість програмних функцій ініціюються через віконні меню.

Якщо інформація завелика для вікна, використовуються скроллбари (scrollbars). Деякі із елементів меню викликають вікна-діалоги (dialogs), у яких користувач може вводити інформацію.

Більшість віконних програм надають користувачу клавіатурний інтерфейс та інтерфейс миші, тому користувач може обирати зручніший для себе спосіб взаємодії із програмою.

Оскільки більшість сучасних ОС підтримує багатозадачність (multitasking),можлива одночасна робота декількох віконних програм.

Користувач може рухати вікна по екрані, змінювати їх розмір.

Програмування GUI диктує використання певних принципів **об’єктно-орієнтованого програмування**. Все побудовано на використанні спеціальних об’єктів - “**вікон**”.

Програмування графічного інтерфейсу і віконних програм є **подійно-орієнтованим**. Взаємодія користувача із вікнами генерує події відповідних типів із даними, які через **систему подій ОС** обробляються у первих функціях-обробниках програми.

Класи для представлення віконних елементів за допомогою наслідування організовують у ієрархії

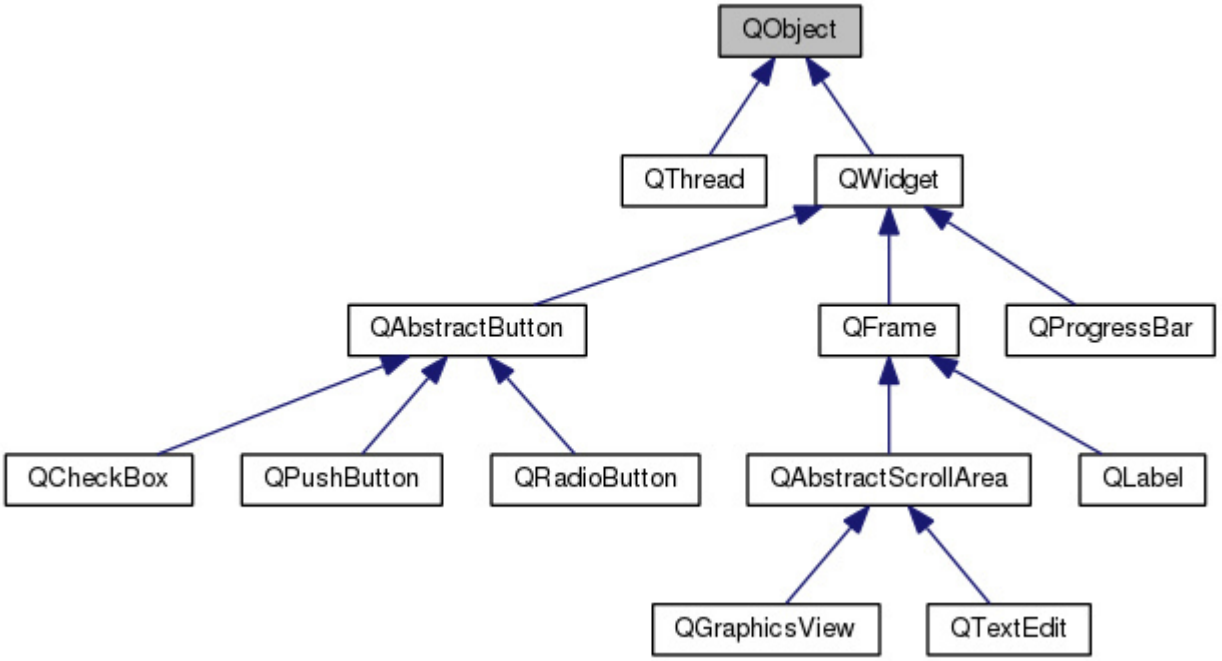


Рис. Ієрархія віконних класів в Qt:

Завдання на виконання лабораторної роботи

(Модифіковане завдання із лабораторної роботи №4)

Створити десктопну програму із графічним інтерфейсом користувача для роботи зі списком сутностей типу **Кінофільм**.

Головне вікно програми має містити:

- Меню із командами:
  - Створити новий пустий список сутностей
  - Завантажити список сутностей із CSV або JSON/XML (на вибір) файлу
  - Зберегти поточний список сутностей у CSV або JSON/XML (на вибір) файл
- Графічний список сутностей та додатковий список результатів фільтрації списку

- Панель відображення детальної інформації про обраний зі списку елемент
- Кнопки для створення нової сутності та редагування або видалення обраної сутності
- Форма для вводу даних фільтрації списку за варіантом завдання

Також програма має мати діалогове вікно для створення або редагування сутності.

Контрольні питання

1. Що таке клас і об“єкт? Як описуються класи у мові C++?
2. Що таке методи класу? Що таке конструктори та деструктори?
3. Які стандартні колекції доступні у мові C++?
4. Що таке графічне вікно?

Лабораторна робота №6

тема “Клієнт-серверна архітектура програм”

Мета роботи – Навчитися реалізувати взаємодію між двома програмами за допомогою протоколу TCP. Вивчити принципи побудови клієнт-серверних архітектур програмних систем.

Короткі теоретичні відомості

Хост - це комп’ютер, що підключений до мережі.

Мережевий сокет (network socket) - це кінцева точка двосторонньої міжпроцесної взаємодії через комп’ютерну мережу.

На теперішній час найпоширенішим протоколом взаємодії є Internet протокол, тому більшість мережевих сокетів є **інтернет сокетами**.

API сокетів - це програмний інтерфейс, який надається операційною системою і дозволяє програмам контролювати і використовувати мережеві сокети.

Клієнт-серверна модель - це розподілена програмна структура, що розбиває завдання, або навантаження між постачальниками ресурсу чи сервісу (**серверами**) і запитувачами ресурсів чи сервісу (**клієнтами**).

Сокет - це файл, доступ до якого програмі надає ОС.

Після отримання доступу сокет можна налаштувати як **пасивний** (серверний) або **активний** (клієнтський).

Адреса сокета - це комбінація **IP адреси** і **номера порта**. На основі цієї адреси інтернет сокети доставляють вхідні пакети даних до відповідного сокета ОС.

Завдання на виконання лабораторної роботи

(Модифіковане завдання із лабораторної роботи №5)

Розділити код програми із попередньої лабораторної роботи на клієнтську та серверну частини та реалізувати взаємодію між ними.

TCP-сервер

Серверна частина має виконувати всю роботу по взаємодії зі списками сутностей, що зберігаються у дискових файлах. Сервер працювати лише із файлами, що розташовані у директорії /data відносно файлу сервера.
Також сервер має відповідати на TCP запити від клієнтів:

- Отримання списку назв файлів із директорії /data сервера
- Запит на завантаження робочого списку сутностей із файлу із сутностями
- Запит на створення нового робочого списку сутностей
- Запит на створення нової сутності до робочого списку
- Запит на видалення сутності зі списку по її ідентифікатору
- Запит на оновлення значень полів сутності за її ідентифікатором
- Запит на отримання робочого списку всіх сутностей
- Запит на збереження робочого списку сутностей у файл директорії /data

TCP-клієнт

Клієнтська частина має мати графічний інтерфейс користувача для його взаємодії із списком сутностей сервера.
Після запуску клієнта користувачу показується форма підключення до сервера. Дана форма повинна містити поле для вводу повної адреси сервера. При неуспішному підключенні виводити відповідну помилку на форму. При успішному підключенні переходити до головного вікна для управління робочим списком сервера.

Додатково графічний інтерфейс має містити вікно зі списком файлів із директорії /data сервера, до якого підключений клієнт. Це вікно має відображатись для завантаження списку сутностей із файлу сервера або для збереження робочого списку сутностей у файл на сервері.

Методичні вказівки

- Для передачі сутностей між програмами їх потрібно серіалізувати у обраний текстовий формат (XML або JSON).
- Код типів сутностей, модуль серіалізації та десеріалізації потрібно винести у статичну бібліотеку, яка підключається до клієнтської та серверної програм.

Контрольні питання

1. Що таке комп“ютерна мережа? Хост? IP-адреса?
2. Опищіть принцип роботи протоколу TCP

Рекомендована література

1. М. Эллис, Б. Строуструп. Справочное руководство по языку C++ с комментариями: Пер. с англ. - Москва: Мир, 1992. 445с.
2. Стенли Б. Липпман. C++ для начинающих: Пер. с англ. 2тт. - Москва: Унитех; Рязань: Гэлион, 1992, 304-345сс.
3. Бруно Бабэ. Просто и ясно о Borland C++: Пер. с англ. - Москва: БИНОМ, 1994. 400с.
4. В.В. Подбельский. Язык C++: Учебное пособие. - Москва: Финансы и статистика, 1995. 560с.
5. Т. Сван. Освоение Borland C++ 4.5: Пер. с англ. - Киев: Диалектика, 1996. 544с.
6. Г. Шилдт. Самоучитель C++: Пер. с англ. - Санкт-Петербург: BHV-Санкт-Петербург, 1998. 620с.