Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

Механіко-технологічний факультет

ЗВІТ

ПРО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ № 9

з навчальної дисципліни

“Базові методології та технології програмування”

РЕАЛІЗАЦІЯ ПРОГРАМНИХ МОДУЛІВ РОЗГАЛУЖЕННЯ ТА ІТЕРАЦІЙНИХ ОБЧИСЛЮВАЛЬНИХ ПРОЦЕСІВ

ЗАВДАННЯ ВИДАВ

доцент кафедри кібербезпеки та програмного забезпечення

Доренський О. П.

[https://github.com/odorenskyi/](https://github.com/odorenskyi/Dmytro-Parkhomenko-KB18)

ВИКОНАВ

студент академічної групи КБ-24

Радченко М. Д.

https://github.com/Levanori

ПЕРЕВІРИВ

ст. викладач кафедри кібербезпеки   
та програмного забезпечення

Коваленко А. С.

Кропивницький – 2025

**Мета роботи :**

полягає у набутті ґрунтовних вмінь і практичнихнавичок реалізації технології модульного програмування,застосування операторів С/С++ арифметичних, логічних, побітовихоперацій, умови, циклів та вибору під час розроблення статичнихбібліотек, заголовкових файлів та програмних засобів у кросплатформовому середовищі Code::Blocks.

**Завдання :**

1. Реалізувати функції розв’язування задач 9.1–9.3 як складовихстатичної бібліотеки libModulesПрізвище.а (проект ModulesПрізвище,створений під час виконання лабораторної роботи №8).

2. Реалізувати програмне забезпечення розв’язування задачі 9.4 на основі функцій статичної бібліотеки libModulesПрізвище.а.

**Варіант :** №21

**ЗАДАЧА 9.1**

Вхід: статистичні дані чисельності населення області: загальна чисельність населення області станом на 1 січня ц.р. та кількість громадян, які станом на 1 число поточного місяця ц.р. знялись з реєстрації (вибули), зареєструвались на постійне місце проживання (прибули), народились, вмерли.

Вихід: чисельність населення області станом на 1 число поточного місяця та інформація про зменшення/збільшення кількості мешканців області у період з 1 січня ц.р. до 1 числа поточного місяця ц.р.

**Аналіз умови до задачі 9.1**

Користувач (в ролі системи обробки статистики) має на вході статистичні дані про населення області: Загальна чисельність населення станом на 1 січня поточного року (ц.р.). Дані про кількість громадян, які: Знялися з реєстрації (вибули), Зареєструвались на постійне місце проживання (прибули), Народилися, Померли — станом на 1 число поточного місяця ц.р. Необхідно: Обчислити: Чисельність населення області станом на 1 число поточного місяця. Інформацію про зміну чисельності населення (зменшення або збільшення) за період з 1 січня ц.р. до 1 числа поточного місяця. Умови коректності: Всі вхідні дані (чисельність, кількість прибулих, вибулих, народжених, померлих) мають бути додатними або нульовими числами. Формули для обчислень: Нова чисельність населення = Початкова чисельність + прибулі + народжені - вибулі - померлі. Зміна чисельності = Нова чисельність - Початкова чисельність.

**Алгоритм до задачі 9.1**

START

1. PRINT "Введіть чисельність населення на 1 січня ц.р.";

2. INPUT startPopulation;

3. PRINT "Введіть кількість прибулих громадян";

4. INPUT arrived;

5. PRINT "Введіть кількість вибулих громадян";

6. INPUT departed;

7. PRINT "Введіть кількість народжених громадян";

8. INPUT born;

9. PRINT "Введіть кількість померлих громадян";

10. INPUT died;

11. CALCULATE currentPopulation = startPopulation + arrived + born - departed - died;

12. CALCULATE populationChange = currentPopulation - startPopulation;

13. PRINT "Чисельність населення на 1 число поточного місяця: ", currentPopulation;

14. PRINT "Зміна чисельності населення з 1 січня ц.р.: ", populationChange;

END

**ЗАДАЧА 9.2**

**Вхід:** зафіксовані в Кропивницькому показники швидкості переміщення повітряних мас (вітру, м/сек) щогодини.  
**Вихід:** найменша швидкість вітру (в балах Бофорта), зафіксована протягом доби.

Таблиця 1 - **Розміри шоломів**

|  |  |  |
| --- | --- | --- |
| БАЛ БОФОРТА | ШВИДКІСТЬ ВІТРУ м/сек | ДІЯ ВІТРУ |
| 0 | < 0.3 | Відсутність вітру. Дим піднімається прямовисно. Листя дерев нерухомі. |
| 1 | 0.3 – 1.5 | Дим «пливе». Флюгер не обертається. |
| 2 | 1.6 – 9.4 | Рух повітря відчувається обличчям. Шелестить листя. Флюгер обертається. |
| 3 | 9.4 – 5.4 | Тріщать листя, хитаються дрібні гілки. Майорять прапори. |
| 4 | 5.5 – 7.9 | Хитаються тонкі гілки дерев. Вітер піднімає пил та шматки паперу. |
| 5 | 8.0 – 10.7 | Хитаються великі гілки. На воді з’являються хвилі. |
| 6 | 10.8 – 13.8 | Хитаються великі гілки. |
| 7 | 13.9 – 17.1 | Хитаються невеликі стовбури дерев. На морі здіймаються хвилі, піняться. |
| 8 | 17.2 – 20.7 | Ламаються гілки дерев. І важко йти проти вітру. |
| 9 | 20.8 – 24.4 | Невеликі руйнування. Зриває покрівлі, руйнує димарі. |
| 10 | **24.5 – 28.4** | Значні руйнування. Дерева вириваються з корінням. |
| 11 | **28.5 – 32.6** | Великі руйнування. |
| 12 | > 32.7 | Призводить до спустошень. |

**Аналіз задачі 9.2**

Користувач вводить значення швидкості переміщення повітряних мас (вітру) в м/сек, зафіксовані щогодини в Кропивницькому протягом доби.  
Необхідно визначити мінімальну зафіксовану швидкість за добу та перевести її в шкалу балів Бофорта, використовуючи відповідність за заданою таблицею.

**Алгоритм до задачі 9.2**

START

1. PRINT "Enter wind speeds for 24 hours (in m/s)";

2. INPUT speeds[24];

3. minSpeed = speeds[0];

4. FOR i = 1 TO 23 DO

IF speeds[i] < minSpeed THEN

minSpeed = speeds[i];

ENDIF

ENDFOR

5. IF minSpeed < 0.3 THEN

PRINT "Beaufort scale: 0 (Calm)"

6. ELSE IF 0.3 ≤ minSpeed ≤ 1.5 THEN

PRINT "Beaufort scale: 1"

7. ELSE IF 1.6 ≤ minSpeed ≤ 3.3 THEN

PRINT "Beaufort scale: 2"

8. ELSE IF 3.4 ≤ minSpeed ≤ 5.4 THEN

PRINT "Beaufort scale: 3"

9. ELSE IF 5.5 ≤ minSpeed ≤ 7.9 THEN

PRINT "Beaufort scale: 4"

10. ELSE IF 8.0 ≤ minSpeed ≤ 10.7 THEN

PRINT "Beaufort scale: 5"

11. ELSE IF 10.8 ≤ minSpeed ≤ 13.8 THEN

PRINT "Beaufort scale: 6"

12. ELSE IF 13.9 ≤ minSpeed ≤ 17.1 THEN

PRINT "Beaufort scale: 7"

13. ELSE IF 17.2 ≤ minSpeed ≤ 20.7 THEN

PRINT "Beaufort scale: 8"

14. ELSE IF 20.8 ≤ minSpeed ≤ 24.4 THEN

PRINT "Beaufort scale: 9"

15. ELSE IF 24.5 ≤ minSpeed ≤ 28.4 THEN

PRINT "Beaufort scale: 10"

16. ELSE IF 28.5 ≤ minSpeed ≤ 32.6 THEN

PRINT "Beaufort scale: 11"

17. ELSE

PRINT "Beaufort scale: 12"

18. ENDIF

END

**ЗАДАЧА 9.3**

**Вхід**: натуральне число N від 0 до 500700.

**Вихід**: якщо біт D7 числа N рівний 0, кількість двійкових нулів у ньому, інакше - кількість двійкових нулів\*.

\*під час підрахунку кількості бінарних 0 або 1 рекомендовано використати тернарний оператор «? :».

**Аналіз задачі 9.3**

Користувач вводить натуральне число NNN в діапазоні від 0 до 500700.  
Спочатку потрібно перетворити це число у двійкову систему числення.  
Далі потрібно проаналізувати значення біта з номером 7 (рахуючи з нуля справа наліво).

**Алгоритм до задачі 9.3**

START

SET number

SET countZero

SET binaryNumber[32]

PRINT "Enter number"

INPUT number

PRECONDITION: i := 0

FOR поки number > 0

binaryNumber[i] := number % 2

IF number % 2 == 0 THEN

countZero := countZero + 1

ENDIF

number := number / 2

i := i + 1

ENDWHILE

IF binaryNumber[7] == 0 THEN

PRINT "Number of binary zeros: " countZero

ELSE

PRINT "Number of binary zeros: " countZero

ENDIF

END

**ЗАДАЧА 9.4**

За введеним користувачем символом "d" викликається s\_calculation(), "g" — функція задачі 9.1, "h" — функція задачі 9.2, "j" — функція задачі 9.3; якщо користувач вводить інші символи, вони ігноруються, при цьому видається звуковий сигнал про помилкове введення. Після цього, якщо користувач за запитом додатка вводить символ "k", "K" або "к", відбувається вихід з програми, інакше — виконання програми повторюється.

**Аналіз задачі 9.4**

Користувач вводить символ з клавіатури. Якщо введений символ — "е", викликається функція s\_calculation(). Якщо символ — "r", викликається функція задачі 9.1. Якщо символ — "f", викликається функція задачі 9.2. Якщо символ — "t", викликається функція задачі 9.3. Якщо введений символ не відповідає жодному з вищезазначених, він ігнорується, і виводиться звуковий сигнал про помилкове введення. Після виконання відповідної функції програма запитує у користувача: якщо він вводить "g", "G" або "о", програма завершується. В іншому випадку виконання програми повторюється, і користувач знову вводить символ для подальших дій.

**Алгоритм до задачі 9.4**

**START**

1. WHILE поки action != ‘g’ AND action != ‘G’ AND action != ‘o’

2. PRINT "Enter action";

3. INPUT action;

4. IF action = ‘e’ THEN

s\_calculate();

5. ELSE IF action = ‘r’

task\_9\_1();

6. ELSE IF action = ‘f’

task\_9\_2();

7. ELSE IF action = ‘t’

task\_9\_3();

8. ELSE

PRINT "Звуковий сигнал";

9. ENDWHILE

**END**

**Висновок**

Під час виконання лабораторної роботи № 9 на тему «Реалізація програмних модулів розгалужених та ітераційних обчислювальних процесів» з курсу «Базові Методології та Технології Програмування» я здобув практичний досвід у створенні статичних бібліотек, розробці власних модулів і заголовних файлів, а також у написанні текстових драйверів для своїх модулів. В процесі роботи я набув глибоких навичок модульного програмування, освоїв використання арифметичних, логічних та побітових операцій, умовних операторів, циклів та операторів вибору в мовах C/C++. Все це я застосовував при розробці статичних бібліотек і програмних засобів у кросплатформовому середовищі Code::Blocks.

Під час лабораторної роботи було виконано такі пункти:

1. Завантажено завдання №9 із репозиторію за допомогою команди git pull
2. Після цього було створено папки prj, Software, Test Suite, та Report і заповнено README.md файл.
3. Прочитано завдання та виконано декомпозицію задачі 9.1
4. Виконано аналіз першого завдання, в результаті чого, створено файл–шаблон тестового набору (Unit testing) із назвою UT\_9\_1.
5. Розроблено 5 Test Case
6. У Code::Blocks створено статичну бібліотеку із назвою ModulesRadchenko
7. За допомогою визначених вимог до задачі та розробленого алгоритму було реалізовано функцію task\_9\_1 () і поміщено прототип функції в заголовний файл ModulesRadchenko.h.
8. Прочитано завдання та виконано декомпозицію задачі 9.2
9. Виконано аналіз другого завдання, в результаті чого, створено файл–шаблон тестового набору (Unit testing) із назвою UT\_9\_2.
10. Розроблено 5 Test Case
11. За допомогою визначених вимог до задачі та розробленого алгоритму було реалізовано функцію task\_9\_2 () і поміщено прототип функції в заголовний файл ModulesRadchenko.h.
12. Прочитано завдання та виконано декомпозицію задачі 9.3
13. Виконано аналіз другого завдання, в результаті чого, створено файл–шаблон тестового набору (Unit testing) із назвою UT\_9\_3.
14. Розроблено 5 Test Case
15. За допомогою визначених вимог до задачі та розробленого алгоритму було реалізовано функцію task\_9\_3 () і поміщено прототип функції в заголовний файл ModulesRadchenko.h.
16. Потім було написаний тестовий драйвер для функції task\_9\_1(), task\_9\_2(), task\_9\_3() та протестовано із всіма позначками passed.
17. Після компіляції TestDriver було створено TestDriver.exe . Цей файл був переміщенний в папку Software.
18. Після компіляції Radchenko\_task було створено Radchenko\_task.exe . Цей файл був переміщенний в папку Software.
19. Виконано системне тестування ПЗ застосунку Radchenko\_task.exe, результати записані у TS\_9\_4, всі тести були пройдені із результатом passed.
20. Виконано системне тестування ПЗ застосунку Radchenko\_task.exe, результати записані у TS\_8\_2, всі тести були пройдені із результатом passed.
21. Всі данні було відправлено на репозиторій в папку lab09