Київський національний університет імені Тараса Шевченка

Механіко-математичний факультет

Реферат на тему:

**«Командний рядок. Парсинг командного рядка»**

Виконав:

студент 2 курсу, 1-ша групи комп’ютерної математики

Скриник Владислав

Київ 2022

**ЗМІСТ**

Вступ

завдання 1).

завдання 2).

завдання 3).

завдання 4).

завдання 5).

завдання 6).

завдання 7).

Порівняння роботи на Windows і Linux.

Висновки

**Вступ**

Потрібно скласти програму яка взяємодіє з користувачем, парсить користувацький ввід та виконує відповідні завдання. Потрібно було реалізувати ці завдання на мові програмування C використовуючи стандартні функції цієї мови програмування. Також потрібно було реалізувати ці ж завдання на мові програмування C++ користуючись функціями стандартної бібліотки STL.

Деякі функції(детальніше всі функції будуть розібрані відповідно в кожному пункті) наприклад fopen() компілятор Visual Studio не дозволяє використовувати, замість цієї функції зазвичай створюється ідентична по роботі функція, тільки в кінці назви добавляється “\_s”. Також можна буде порівняємо як працює код на Windows та Linux(на жаль я не маю змоги писати саме на Linux тому використовую дистрибутив Linux – Ubuntu).

Найбільша складність в будь якому з цих завдань – це реалізувати ефективний алгоритм який парсить стрічку яку ввів користувач. В пунктах нижче буде наведено різні рішення цієї проблеми.

Для програми на С++ було використано бібліотеки vector та algorithm.

**завдання 1).**

В завдання 1) треба було написати програму, що приймає з командного рядку 1 цілий аргумент та вивести його квадрат. Якщо аргументів 2 або більше, або жодного – вивести повідомлення про помилку.

*а) Розглянемо реалізацію програми на С:*

На початку програми створюємо стрічку яку має ввести користувач таким чином:

* Створюємо змінну **char str[50] = "";** яку має ввести користувач через метод **fgets(str, 50, stdin)** далі замінюємо останній символ на символ “\0”
* **str[strlen(str) - 1] = '\0';** це зроблено тому що останній символ який ввів користувач коли вводив стрічку буде символ “\n”, оскільки саме так працює метод **fgets()**, тому нам треба просто його замінити.

Далі будемо використовувати відомий алгоритм щоб розбити нашу стрічку на “слова”:

**char\* token = strtok(str, " ");**

**int userInput;**

**int count = 0;**

**while (token != NULL) {**

**count++;**

**//printf("%s\n", token);**

**char\* ptr;**

**userInput = strtol(token, &ptr, 10);**

**token = strtok(NULL, " ");**

**}**

В цьому циклі token виступає в ролі кожного “слова” яке ввів користувач через пробіл. Змінна count рахує всі слова, таким чином якщо після закінчення циклу змінна count буде більше або менше 1, це означатиме що користувач некоректно ввів дані. Також будемо перетворювати стрічку яку ввів користувач в ціле число і будемо записувати її в змінну userInput за допомогою функції strtol().

strol(const char \*str, char \*\*endptr, int base) – функція перетворює рядок str на ціле число за базою base. Функція працює, ігноруючи будь-які пробіли на початку рядка, перетворюючи наступні символи на ціле число та зупиняючись, коли зустрічається перший нецілий символ.

Також був використана функція strtok()

strtok(char \*str, const char \*delim) – функція яка приймає 2 параметри str та delim. Функція розбиває рядок str на серію так званих токенів за допомогою роздільника delim.

char \*fgets(char \*str, int n, FILE \*stream) – функція зчитує символи від поточної позиції потоку до першого символу нового рядка (\n) включно, до кінця потоку або доки кількість прочитаних символів не дорівнює n-1, залежно від того, що відбудеться раніше. Функція fgets() зберігає результат у рядку та додає нульовий символ (\0) у кінець рядка. Рядок містить символ нового рядка, якщо він прочитаний. Якщо n дорівнює 1, рядок порожній.

*б) тепер поглянемо як ця ж сама програма буде реалізована на С++:*

На початку програми створюємо стрічку яку має ввести користувач, та будемо перевіряти коректність користувацького вводу користуючись тим фактом що в цій стрічці не повинно бути жодних пробіл, лише цифри:

**std::string userInputSTR = "";**

**std::cout << "Input => ";**

**std::getline(std::cin, userInputSTR);**

**if (userInputSTR.find(' ') != std::string::npos || userInputSTR == "") {**

**std::cout << "Incorrect number of arguments!\n";**

**return -1;**

**}**

Використаємо метод **getline()** який візьме все що введе користувач і запише в змінну **userInputSTR**. Якщо в стрічці яку ввів користувач немає пробілів тоді метод **userInputSTR.find(' ')** поверне значення **std::string::npos,** відповідно в іншому разі виводимо повідомлення про помилку, або якщо користувач взагалі нічого не ввів. Відповідно якщо все гаразд буде ковертувати те що ввів користувач в ціле число таким чином:

**int userInputINT = atoi(userInputSTR.c\_str());**

const char\* c\_str() const – метод перетворює рядок на масив символів з нульовим символом '\0' у кінці.

int atoi(const char strn) --- функція приймає рядковий параметр, який містить цілі значення, і перетворює переданий рядок у ціле значення.

**завдання 2)**

В завданні 2) треба було написати програму, що приймає з командного рядку 3 дійсних аргументи та виводить їх середнє гармонічне. Якщо серед аргументів більше трьох аргументів, або менше – виводиться повідомлення про помилку. Якщо серед них є нуль — інше повідомлення про помилку.

*а) Розглянемо реалізацію програми на С:*

Аналогічно до роботи попередньо програми, ця програма приймає користувацький ввід, і записує його в змінну **char str[50] = "";**

Але в цій програмі не достатнього тієї перевірки на коректність яка була, тому перед циклом який розбивай головну стрічку на слова, створюється які і в попередній раз змінну **int count,** і в циклі збільшужться її значення, відповідно якщо після закінчення роботи циклу змінна count буде більша або менша за 3, це буде свідчити про помилку в стрічці яку ввів користувач:

**char\* token = strtok(str, " ");**

**float userInput1;**

**float userInput2;**

**float userInput3;**

**int count = 0;**

**while (token != NULL) {**

**count++;**

**//printf("%s\n", token);**

**if (count == 1) {**

**char\* ptr;**

**userInput1 = strtof(token, &ptr, 10);**

**}**

**else if (count == 2) {**

**char\* ptr;**

**userInput2 = strtof(token, &ptr, 10);**

**}**

**else if (count == 3) {**

**char\* ptr;**

**userInput3 = strtof(token, &ptr, 10);**

**}**

**token = strtok(NULL, " ");**

**}**

особливості компілятора Visual Studio трохи змінюють кількість параметрів функції strtof, але принцип роботи функції залишається незмінним.

strtof(const char\* str, char \*\*endptr, int base) --- функція яка перетворює str на дійсне число за базою base. параметр endptr це посилання на вже виділений об’єкт типу char\*, значення якого встановлюється функцією до наступного символу в str після числового значення.

*б) тепер поглянемо як ця ж сама програма буде реалізована на С++:*

Робота програми аналогічна до попередньої хіба що дещо змінена умова перевірки на коректність стрічки яку ввів користувач:

**if (std::count(userInputSTR.begin(), userInputSTR.end(), ' ') != 2 || userInputSTR == "") {**

**std::cout << "Incorrect number of arguments!\n";**

**return -1;**

**}**

**завдання 3)**

В завданні 3) користувачу треба ввести командного рядка ім’я текстового файлу та програма мала підрахувати кількість рядків в цьому файлі. Такоє програма має виводити повідомлення про помилку якщо щось негаразд.

*а) Розглянемо реалізацію програми на С:*

Для того щоб підрахувати кількість рядків в файлі будемо рахувати кількість символів ‘\n’ в цьому файлі:

**FILE\* file;**

**errno\_t err;**

**char ch;**

**err = fopen\_s(&file, str, "r");**

**if (err != 0) {**

**printf("file can't be opened \n");**

**return -1;**

**}**

**int count = 0;**

**while (!feof(file)) {**

**ch = fgetc(file);**

**//printf("%c", ch);**

**if (ch == '\n') count++;**

**}**

**fclose(file);**

*б) тепер поглянемо як ця ж сама програма буде реалізована на С++:*

На С++ така програма не потребує особливої уваги таких спеціальних символів як ‘\n’, адже є відомий алгоритм читання рядків в файлі та підрахунку рядків в ньому:

**std::string dateFile;**

**std::ifstream file("task3\_input\_v1.txt");**

**if (!file.good()) {**

**std::cout << "Incorrect name of file\n";**

**return -1;**

**}**

**int count = 0;**

**while (getline(file, dateFile)) {**

**count++;**

**//std::cout << dateFile << "\n";**

**}**

**file.close();**

**завдання 4)**

В завданні 4) користувач має ввести з командного рядка ім’я декількох текстових файлів (їх повинно бути більше одного) та програма повинна підрахувати середню щільність символів на рядок в цих файлах.

*а) Розглянемо реалізацію програми на С:*

Спочатку дізнаємося яку кількість файлів користувач має намір ввести потім, виділяючи динамічно память для того щоб створити масив стрічок питає в користувача окремо назву кожного файлу. Також в циклі рахуємо кількість символів та кількість спеціальних символів ‘\n’ для того щоб знати яка кількість рядків в файлі, та відповідно ділимо одну кількість на іншу, таким чином дізнаємося щільність символів на рядок в файлі.

*б) тепер поглянемо як ця ж сама програма буде реалізована на С++:*

Робота програми на С++ аналогічна до С, хіба що в циклі ми не рахуємо кожен символ окремо, а рахуємо відразу кількість символів в стрічці.

Був використаний спеціальне слова **std::vector**

vector – це те ж саме, що й динамічні масиви, з можливістю автоматично змінювати розмір, коли елемент вставляється або видаляється, а їх зберігання автоматично обробляється контейнером. Векторні елементи розміщуються в безперервному сховищі, щоб до них можна було отримати доступ і обходити їх за допомогою ітераторів. У векторах дані вставляються в кінці. Вставлення в кінці займає різний час, оскільки іноді масив може знадобитися розширити.

**завдання 5)**

В завданні 5) користувач повинен ввести через командний рядок наступного стрічку вигляду: -filename name -rows rows .

Імя файлу (name) та кількість рядків (rows), параметри -filename та -rows – це обов'язкові літерали в рядку. Якщо формат команди не такий як приведений вище – програма виводить повідомлення про помилку та підказку. Якщо все вірно, програма створює відповідний бінарний файл, що містить вказану кількість цілих чисел від 0 до rows.

*а) Розглянемо реалізацію програми на С:*

В цій програмі треба було порівняти чи коректно користувач ввів відповідно 1 та 3 аргументи рядка.

Алгоритм який перевіряє чи однакові дві стрічки наведений нижче:

**int test = 0;**

**char compare1[100] = "-filename";**

**for (int i = 0; compare1[i] != '\0'; i++) {**

**if (token[i] != compare1[i])**

**test = 1;**

**}**

**if (test != 0) {**

**printf("Incorrect input!\nYour input should be like this: -filename name -rows rows\n");**

**return -1;**

**}**

Також був вперше використаний метод запису елементу в файл – fwrite(&i, sizeof(i), 1, file);

Це було зроблено тому що ми працюємо з бінарним файлом.

fwrite(const void \*ptr, size\_t size, size\_t nmemb, FILE \*stream) --- функція яка записує дані з масиву, на який вказує ptr, у заданий файл stream.

*б) тепер поглянемо як ця ж сама програма буде реалізована на С++:*

На С++ робота виглядала схоже хіба що простіше реалізовано порівняння двох стрічок, їх можна порівнювати як звичайні змінні.

Також бву введений новий метод напису данних в файл – file.write(reinterpret\_cast<const char\*>(&i), sizeof(i));

Це було зроблено як і в попередньому випадку тому що ми працюємо з бінарним файлом.

**завдання 6)**

В завданні 6) користувач повинен ввести через командний рядок наступного стрічку вигляду: -filename name -rows rows -cols cols .

Імя файлу (name) та кількість рядків (rows) та стовпчиків(cols),

параметри -filename та -rows, -cols – це обов'язкові літерали в рядку. Якщо формат команди не такий як приведений вище – програма виводить повідомлення про помилку та підказку. Якщо все вірно, створюється відповідний текстовий файл, що містить вказану кількість рядків заповнену cols нулями через табуляцію.

**завдання 7)**

В завданні 7) користувач повинен ввести через командний рядок наступного стрічку вигляду: -filename1 name1 -filename2 name2 -rows rows. Імя файлу (name) та кількість рядків (rows) параметри -filename1

та -filename2, це обов'язкові літерали в рядку. А параметр -rows rows може бути необов'язковий. Якщо формат команди не такий як приведений вище – програма виводить повідомлення про помилку та підказку. Якщо все вірно, порівнюється чи співпадає в даних двох файлах перші rows рядків з точністю до пробілів, якщо rows не вказаний – файли порівнюються повністю всі рядки.

Алгоритм для порівняння рядків що в програмі на С та С++ аналогічний до описаних вище.

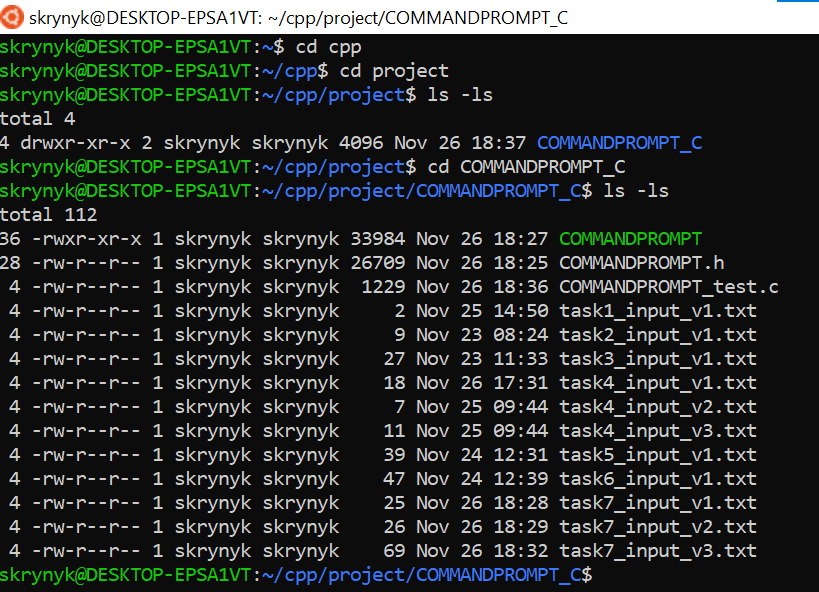
**8.Порівняння роботи на Windows і Linux.**

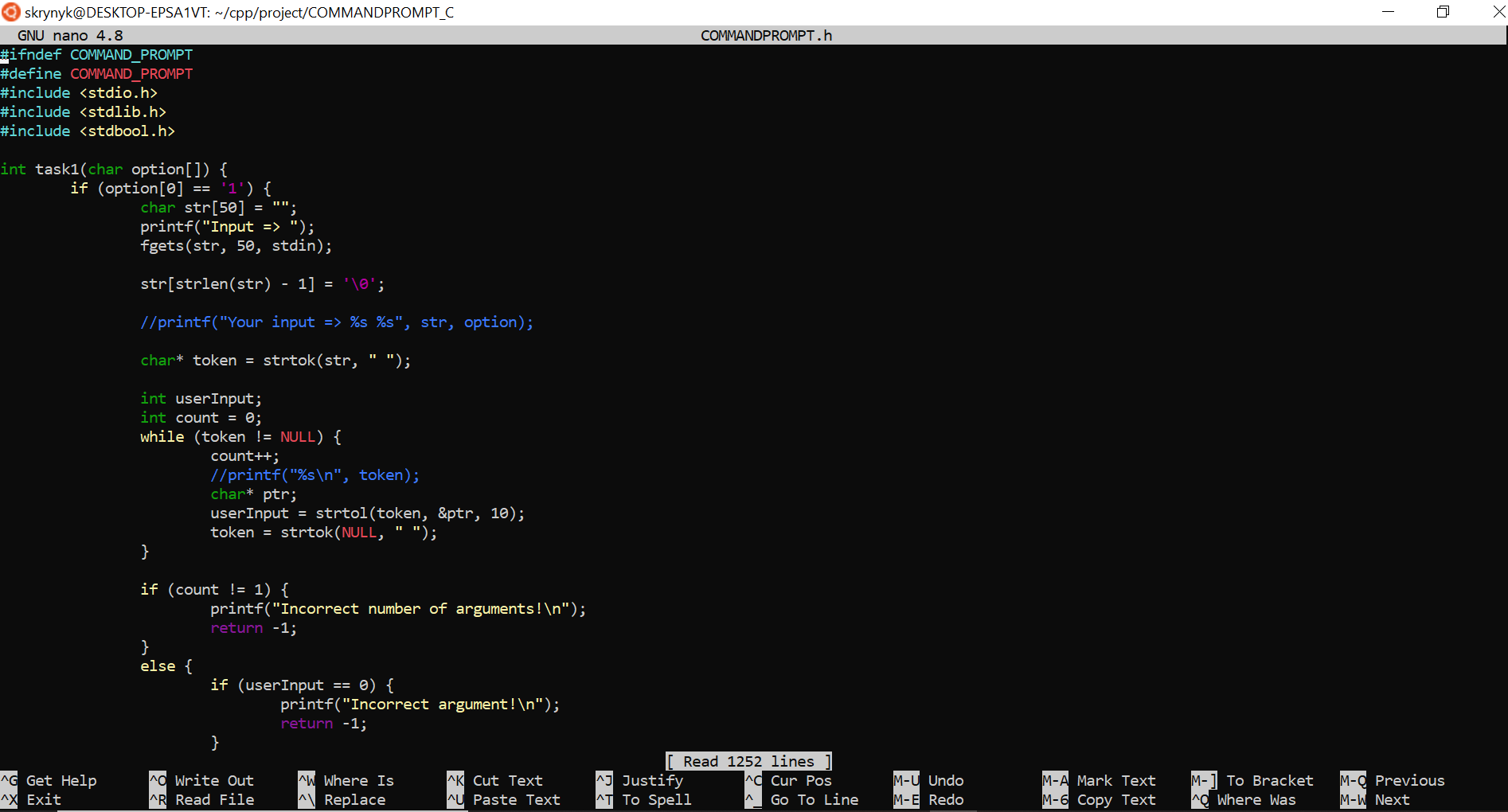
Програма реалізована на С з версією компілятора:

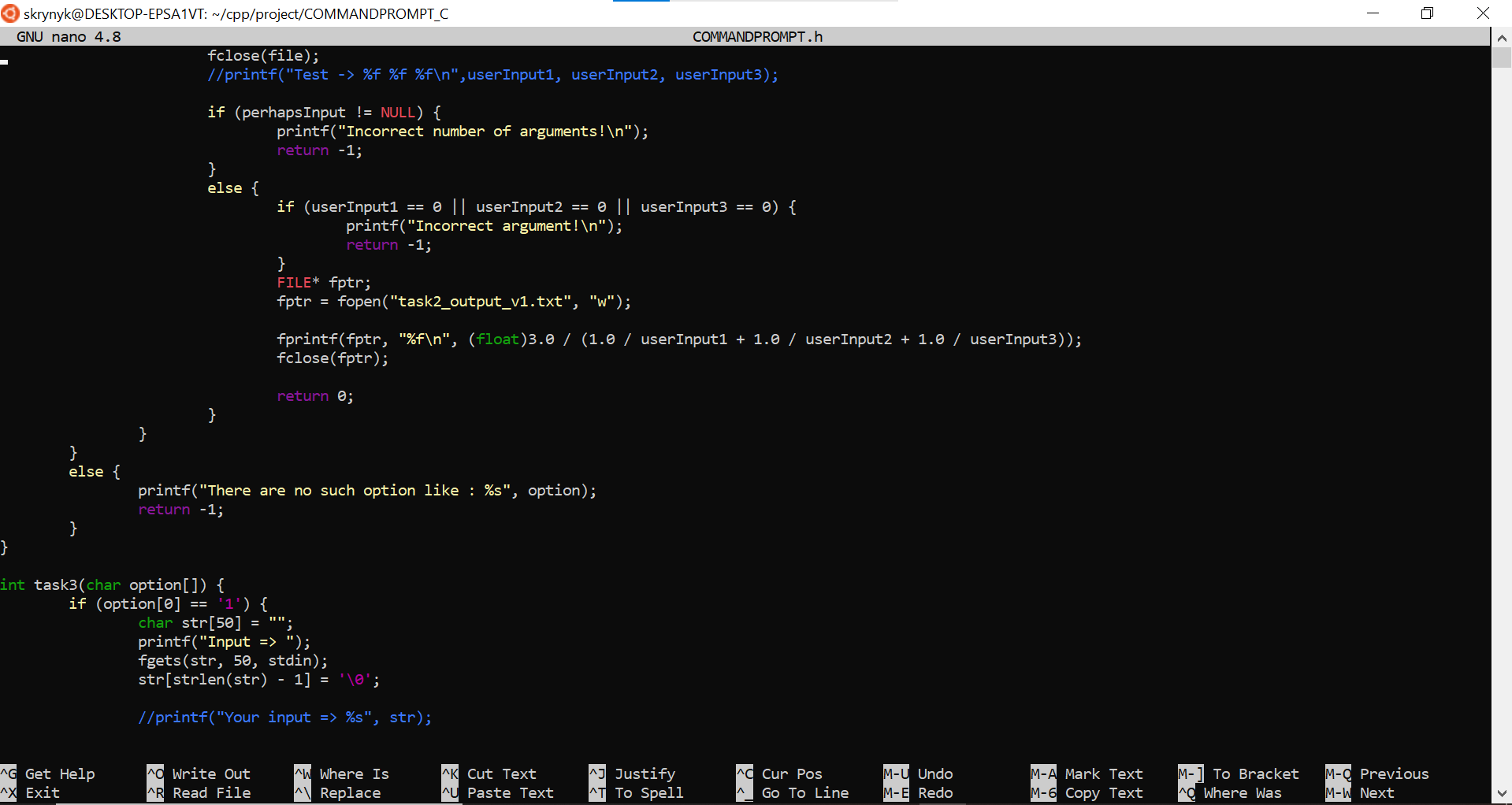
gcc (Ubuntu 9.3.0-17ubuntu1~20.04) 9.3.0

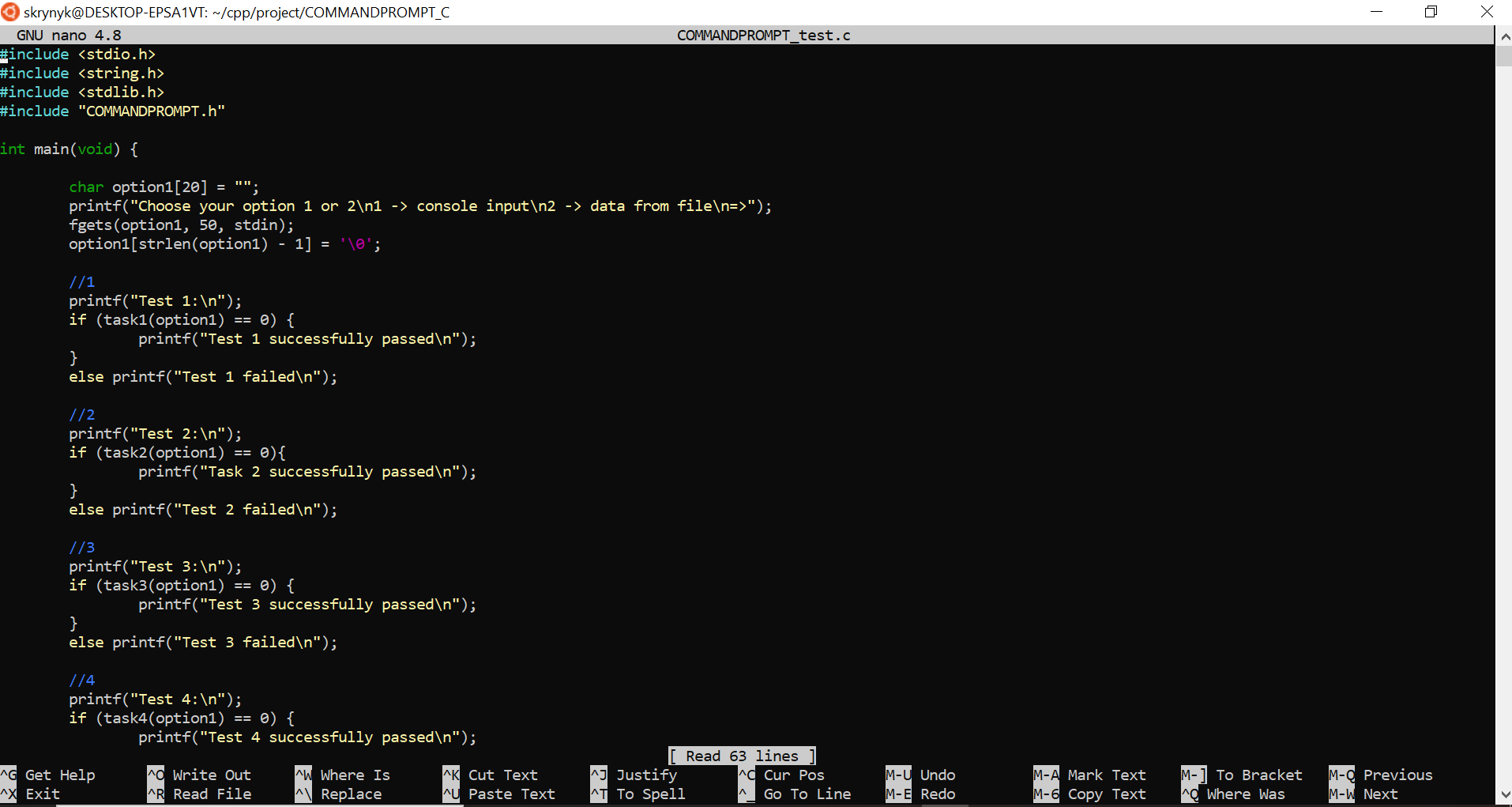
Дозволяє використовувати функції без суфіксу “\_s” які середовище Visual Studio вважало небезпечними. **Самий код та його логіка не відрізняється(хіба що за винятком деяких функцій) від коду написаного на С.**

Якщо робити скріншоти всієї програми вийде занадто багато скрінів, тому публікую лише кілька, проте може показати весь код якщо в Вас виникне таке бажання.









**Висновки**

Деякі стандартні функції Visual Studio вважає небезпечними тому створює їх копію з суфіксом “\_s”, але сама логіка роботи цих функцій не змінюється. Як було наведено вище робота та стиль написання коду не відрізняється від коду написано на Ubuntu.

Для того щоб порівняти чи однакові результати роботи двох кодів(на С і С++) був створений файл COMMANDPROMPT\_cross\_test.cpp в якому викликаються всі функції на С++ і на С та порівнюється вміст відповідних файлів, це було зроблено за допомогою спеціальної функції system().