МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій

Кафедра програмних систем та технологій

**Лабораторна робота № 2.2**з дисципліни

**«Операційні системи»**

Тема:

«Використання компілятора С у середовищі UNIX»

Виконали студенти групи ІПЗ-33:

Поданенко Денис та

Сковородін Михайло

Київ

2018

**ЗМІСТ**

[**ЗАВДАННЯ 1 2**](#_Toc530322158)

[**ЗАВДАННЯ 2 4**](#_Toc530322159)

[**ЗАВДАННЯ 3 5**](#_Toc530322160)

[**ЗАВДАННЯ 4 6**](#_Toc530322161)

[**ЗАВДАННЯ 5 7**](#_Toc530322162)

[**ЗАВДАННЯ 6 8**](#_Toc530322163)

[**ЗАВДАННЯ 7 9**](#_Toc530322164)

[**ЗАВДАННЯ 8 10**](#_Toc530322165)

[**ЗАВДАННЯ 9 11**](#_Toc530322166)

[**ЗАВДАННЯ 10 12**](#_Toc530322167)

[**ЗАВДАННЯ 11 13**](#_Toc530322168)

[**ЗАВДАННЯ 12 14**](#_Toc530322169)

[**ЗАВДАННЯ 13 15**](#_Toc530322170)

[**ЗАВДАННЯ 14 16**](#_Toc530322171)

[**ЗАВДАННЯ 15 17**](#_Toc530322172)

[**ВИСНОВКИ 18**](#_Toc530322173)

[**ДОДАТОК А 19**](#_Toc530322174)

[**1. Лістинг програми Завдання 1. 19**](#_Toc530322175)

[**2. Лістинг програми Завдання 2. 21**](#_Toc530322176)

[**3. Лістинг програми Завдання 3. 22**](#_Toc530322177)

[**4. Лістинг програми Завдання 4. 24**](#_Toc530322178)

[**5. Лістинг програми Завдання 5. 25**](#_Toc530322179)

[**6. Лістинг програми Завдання 6. 26**](#_Toc530322180)

[**7. Лістинг програми Завдання 7. 28**](#_Toc530322181)

[**8. Лістинг програми Завдання 8. 31**](#_Toc530322182)

[**9. Лістинг програми Завдання 9. 33**](#_Toc530322183)

[**10. Лістинг програми Завдання 10. 35**](#_Toc530322184)

[**11. Лістинг програми Завдання 11. 37**](#_Toc530322185)

[**12. Лістинг програми Завдання 12. 39**](#_Toc530322186)

[**13. Лістинг програми Завдання 13. 42**](#_Toc530322187)

[**14. Лістинг програми Завдання 14. 43**](#_Toc530322188)

[**15. Лістинг програми Завдання 15. 44**](#_Toc530322189)

# **ЗАВДАННЯ 1**

Написати програму знаходження максимального, мінімального середньоарифметичного значень ряду з 8 чисел.

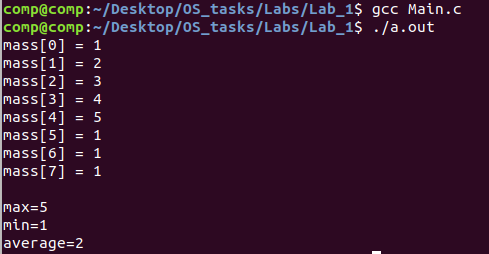


Рисунок 1 – приклад роботи програми

# **ЗАВДАННЯ 2**

Протабулювати функцію y= 1/(1-cos(x)) в заданих межах із заданим кроком.

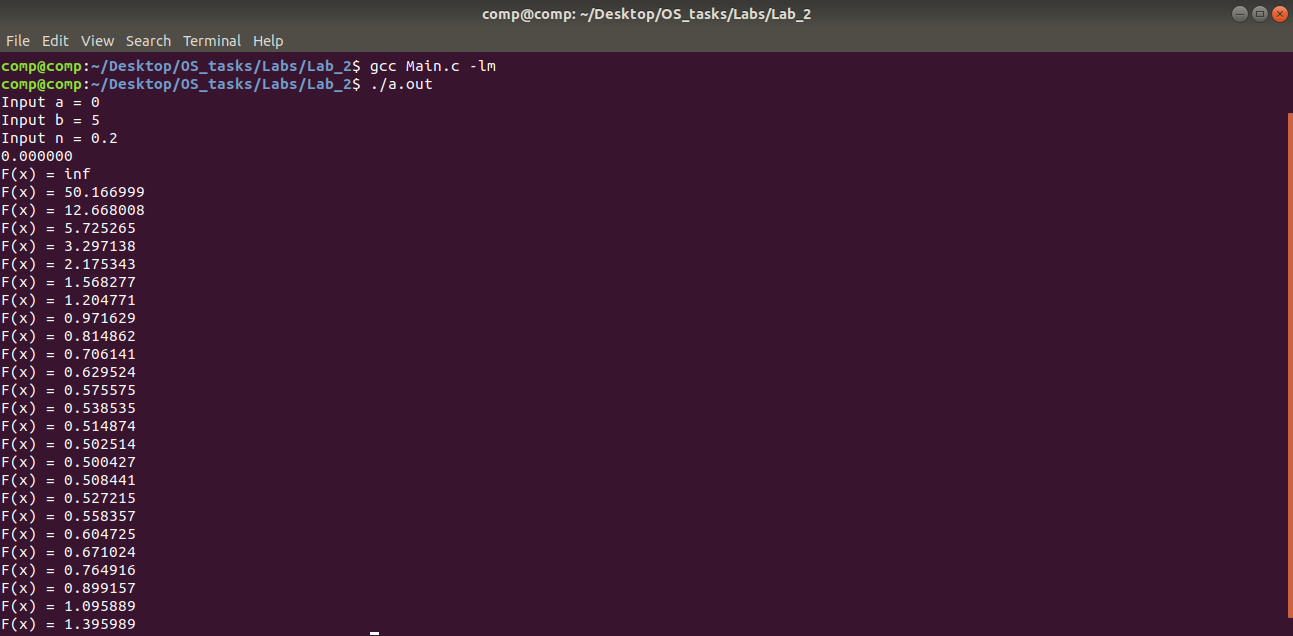


Рисунок 2 – приклад роботи програми

# **ЗАВДАННЯ 3**

Написати програму переводу строки у шістнадцятирічний дамп.

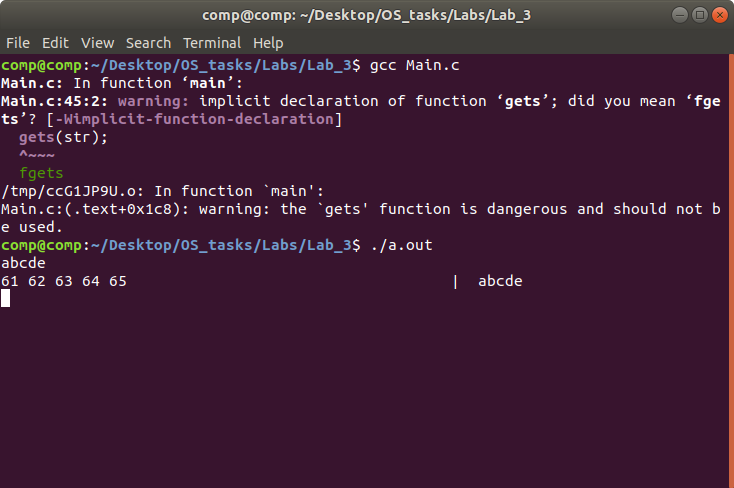


Рисунок 3 – приклад роботи програми

# **ЗАВДАННЯ 4**

Написати програму заміни у рядку одного символу на інший.

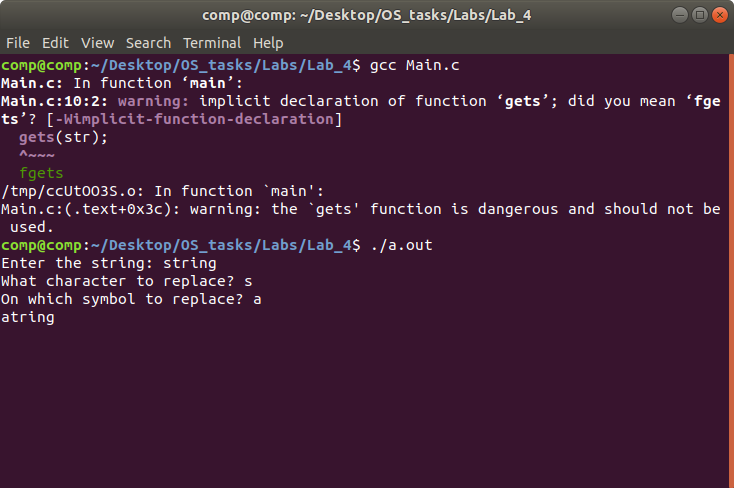


Рисунок 4 – приклад роботи програми

# **ЗАВДАННЯ 5**

Написати програму перетворення строки в її дзеркальне відображення.

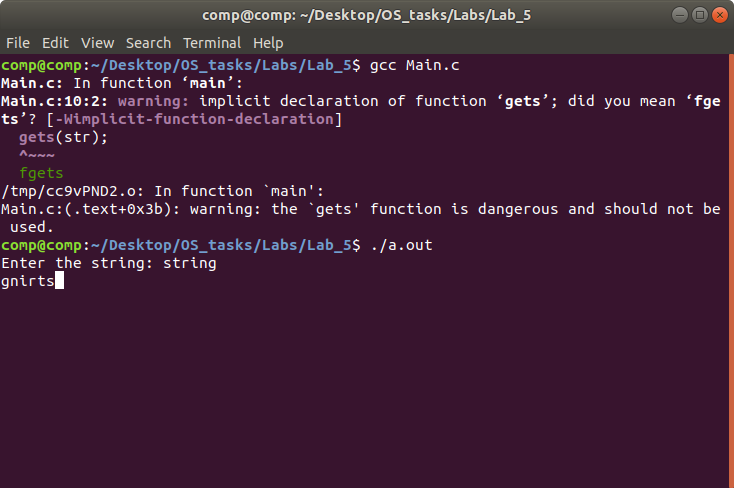


Рисунок 5 – приклад роботи програми

# **ЗАВДАННЯ 6**

Написати програму, яка копіює задані текстові файли, виключаючи символи з кодами 100-110.

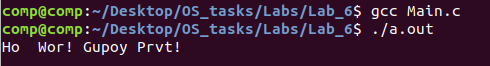


Рисунок 6 – приклад роботи програми

6c2.png

Рисунок 7 – вихідний текст

# **ЗАВДАННЯ 7**

Написати програму, яка шифрує зміст заданого файлу методом Цезаря (ключ задавати через командний рядок).

7c.png

Рисунок 8 – приклад роботи програми

# **ЗАВДАННЯ 8**

Написати програму, яка шукає у заданому файлі слово.

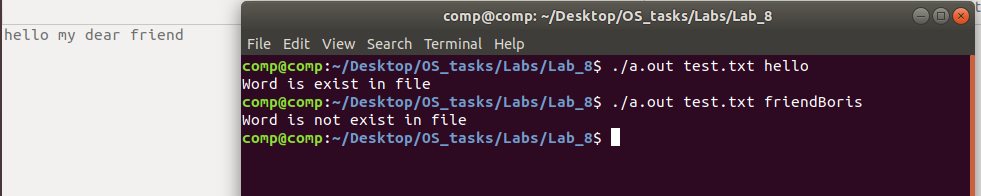


Рисунок 9 – приклад роботи програми

# **ЗАВДАННЯ 9**

Написати програму, яка виділяє з текстового файлу цифрові символи та зберігає їх у іншому файлі (ctype.h, stdlib.h).

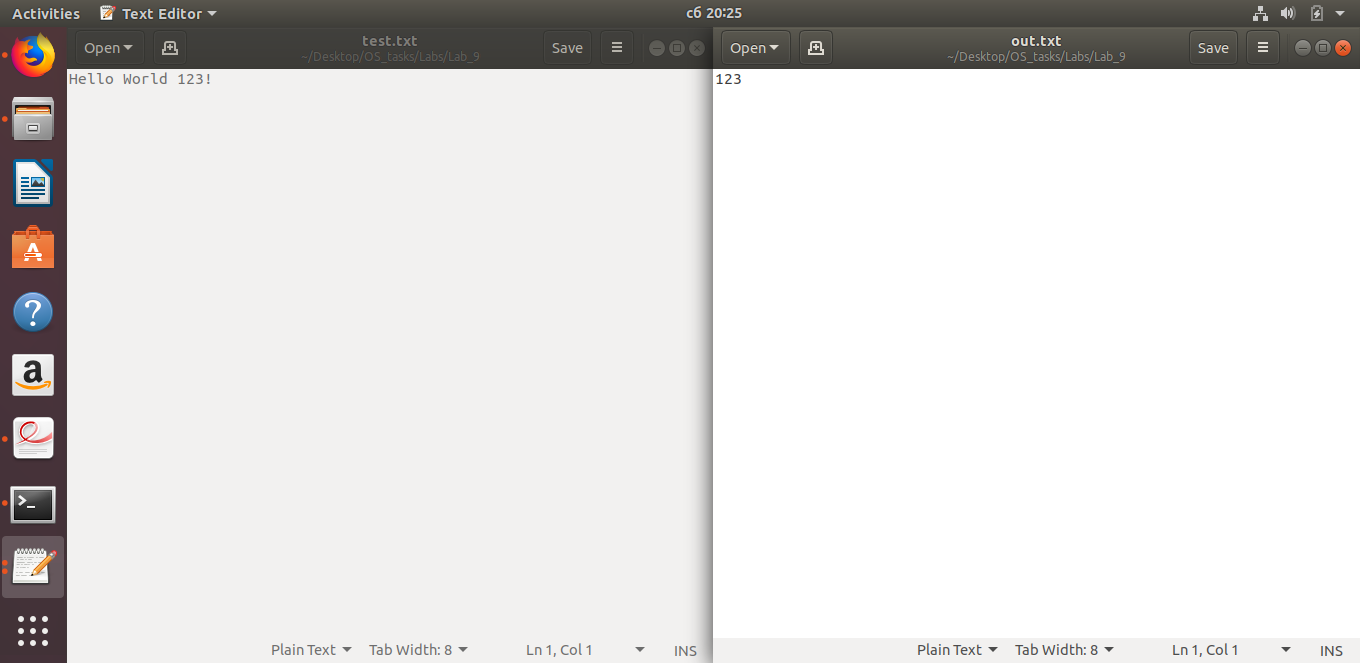


Рисунок 10 – приклад роботи програми

# **ЗАВДАННЯ 10**

Написати програму, яка прибирає зайві символи (!! → !) та пробіли в тексті.

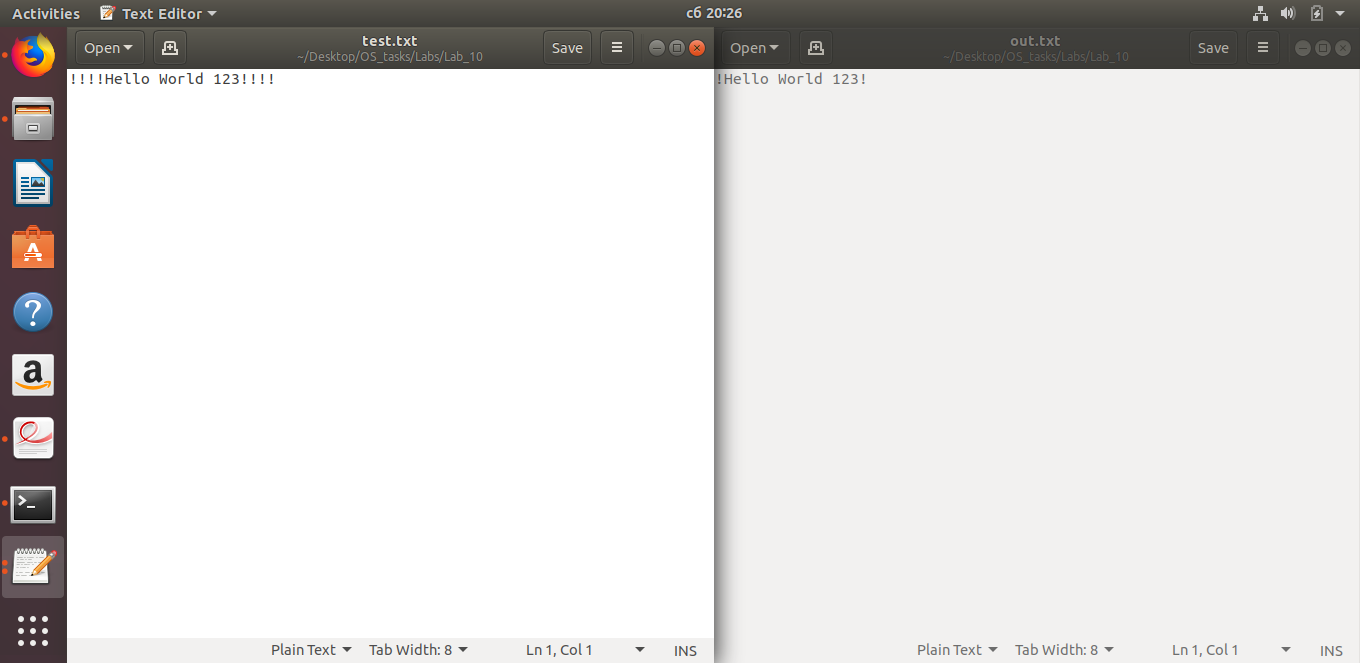


Рисунок 11 – приклад роботи програми

# **ЗАВДАННЯ 11**

Написати програму, яка записує зміст текстового файлу в зворотному порядку.

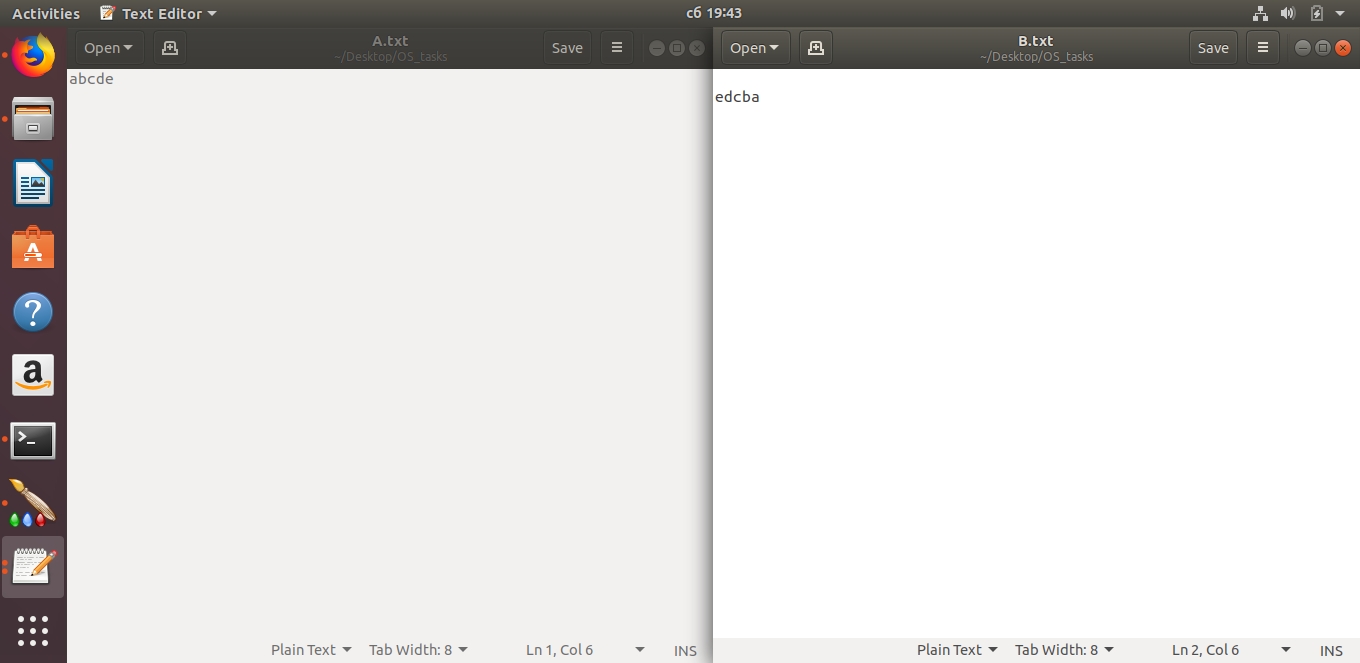


Рисунок 12 – приклад роботи програми

# **ЗАВДАННЯ 12**

Написати програму, яка записує введене слово в задану позицію текстового файлу.

12c1.png

Рисунок 12 – приклад роботи програми

12c2.png

Рисунок 13 – результат роботи програми

# **ЗАВДАННЯ 13**

Написати програму, яка підраховує функцію (наприклад, sin(x)+x-1) та записує результат обчислення в окремий файл (крок та відрізок вводити з клавіатури).

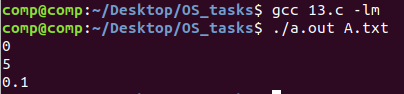


Рисунок 14 – приклад роботи програми

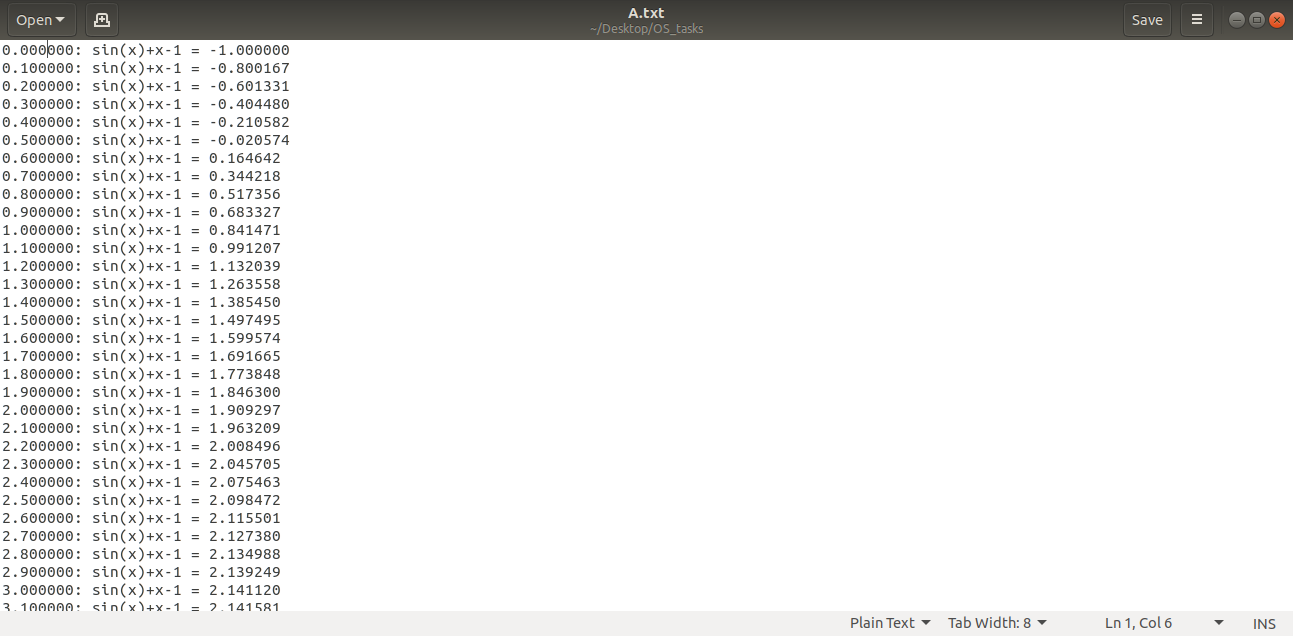


Рисунок 15 – результат роботи програми

# **ЗАВДАННЯ 14**

Написати програму, яка переводить символи алфавіту текстового файлу в їх еквівалентний числовий формат (код символу) та записує в окремий файл.

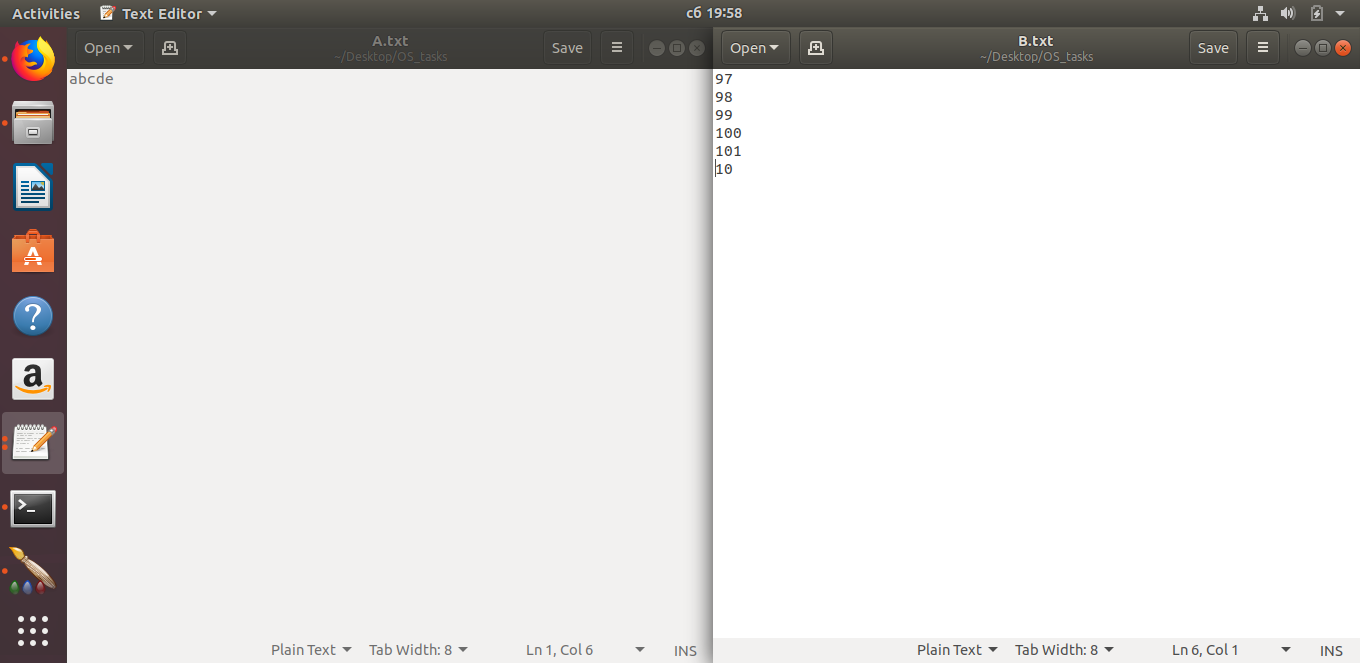


Рисунок 16 – приклад роботи програми

# **ЗАВДАННЯ 15**

Написати програму, яка перевіряє ідентичність двох текстових файлів, результат виводить у вигляді: дата перевірки (ctime.h), назва файлів, результат перевірки та відсоток ідентичності файлів.

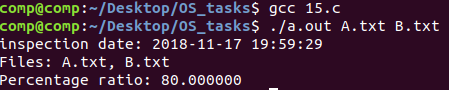


Рисунок 17 – приклад роботи програми

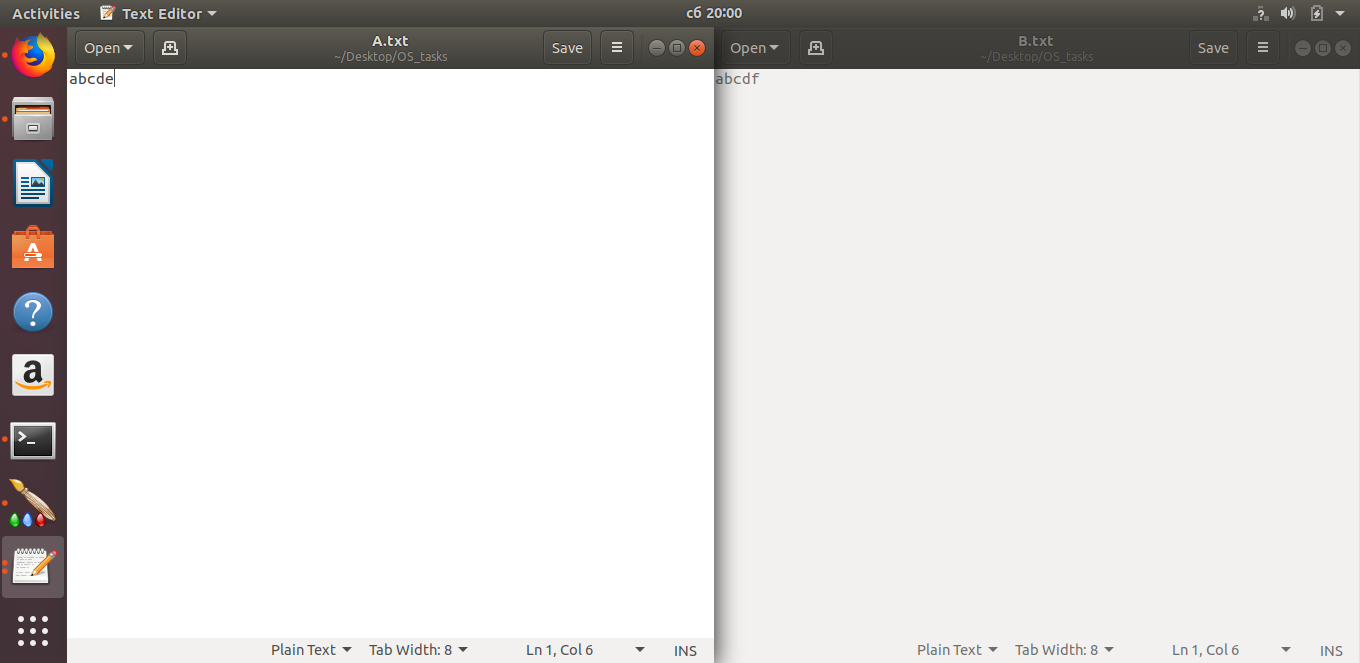


Рисунок 18 – вміст текстових файлів

# **ВИСНОВКИ**

Під час виконання Лабораторної роботи №2.2 були виконані завдання, що направлені на отриманні навичок з використання компілятора С у середовищі UNIX. Це дало змогу зрозуміти принципи роботи з текстовими редакторами та компілятором середовища UNIX, як через термінал, так і через файловий менеджер, та освіжити знання мови програмування C, яка є основною для роботи програміста у UNIX подібних системах.

# **ДОДАТОК А**

## Лістинг програми Завдання 1.

|  |
| --- |
| #include <stdio.h> |
|  | //#include <iostream> |
|  | //#include <conio.h> |
|  |  |
|  | //Функция которая вычисляет максимальный элемент |
|  | void max(int mass[], int size) { |
|  | int max = mass[0]; |
|  | for (int i = 1; i<size; i++) |
|  | if (mass[i]>max) { |
|  | max = mass[i]; |
|  | } |
|  |  |
|  | printf("\nmax=%d", max); |
|  | } |
|  |  |
|  | //Функция которая вычисляет минимальный элемент |
|  | void min(int mass[], int size) { |
|  | int min = mass[0]; |
|  | for (int i = 1; i<size; i++) |
|  | if (mass[i]<min) { |
|  | min = mass[i]; |
|  | } |
|  | printf("\nmin=%d", min); |
|  | } |
|  |  |
|  | //Функция которая вычисляет среднее значение |
|  | void average(int mass[], int size) { |
|  | int av; |
|  | int sum = 0; |
|  | for (int i = 0; i<size; i++) |
|  | { |
|  | sum += mass[i]; |
|  | } |
|  | av = sum / size; |
|  | printf("\naverage=%d", av); |
|  | } |
|  |  |
|  | int main() { |
|  | int n = 8; |
|  | int mass[n]; |
|  |  |
|  | //Заполняем массив |
|  | for (int i = 0; i<n; i++) |
|  | { |
|  | printf("mass[%d] = ", i); |
|  | scanf\_s("%d", &mass[i]); |
|  | } |
|  |  |
|  | int size = sizeof(mass) / sizeof(int); |
|  |  |
|  |  |
|  |  |
|  | max(mass,size); |
|  | min(mass,size); |
|  | average(mass, size); |
|  |  |
|  |  |
|  | getchar(); |
|  | } |

## Лістинг програми Завдання 2.

|  |
| --- |
| #include <stdio.h> |
|  | #include <math.h> |
|  |  |
|  | //Функция которая возвращает значение заданной функции |
|  | double function(double x) |
|  | { |
|  | return 1/(1-cos(x)); |
|  | } |
|  |  |
|  | void main() { |
|  |  |
|  | float a, b, n; |
|  |  |
|  | //Вводим границу a |
|  | printf("Input a = "); |
|  | scanf("%f",&a); |
|  |  |
|  | //Вводим границу b |
|  | printf("Input b = "); |
|  | scanf("%f", &b); |
|  |  |
|  | //Вводим шаг n |
|  | printf("Input n = "); |
|  | scanf("%f", &n); |
|  |  |
|  | printf("%f", a); |
|  |  |
|  |  |
|  | //Вычисляем функцию с заданными параметрами |
|  | for (float i = a; i <= b; i += n) { |
|  | printf( "\nF(x) = %f ", function(i) ); |
|  |  |
|  | } |
|  |  |
|  |  |
|  | getchar(); |
|  | getchar(); |
|  | } |

## Лістинг програми Завдання 3.

|  |
| --- |
| #include <stdio.h> |
|  | #include <string.h> |
|  | #include <stdlib.h> |
|  |  |
|  | //Функция которая возвращает 16-й дамп |
|  | void DumpHex(const void\* data, size\_t size) { |
|  | char ascii[17]; |
|  | size\_t i, j; |
|  | ascii[16] = '\0'; |
|  | for (i = 0; i < size; ++i) { |
|  | printf("%02X ", ((unsigned char\*)data)[i]); |
|  | if (((unsigned char\*)data)[i] >= ' ' && ((unsigned char\*)data)[i] <= '~') { |
|  | ascii[i % 16] = ((unsigned char\*)data)[i]; |
|  | } |
|  | else { |
|  | ascii[i % 16] = '.'; |
|  | } |
|  | if ((i + 1) % 8 == 0 || i + 1 == size) { |
|  | printf(" "); |
|  | if ((i + 1) % 16 == 0) { |
|  | printf("| %s \n", ascii); |
|  | } |
|  | else if (i + 1 == size) { |
|  | ascii[(i + 1) % 16] = '\0'; |
|  | if ((i + 1) % 16 <= 8) { |
|  | printf(" "); |
|  | } |
|  | for (j = (i + 1) % 16; j < 16; ++j) { |
|  | printf(" "); |
|  | } |
|  | printf("| %s \n", ascii); |
|  | } |
|  | } |
|  | } |
|  | } |
|  |  |
|  | void main() { |
|  |  |
|  | /\*char myString[100]; |
|  | printf("Input string: "); |
|  | fgets(myString, 100, stdin); |
|  | printf("Your stirng %s", myString); |
|  | DumpHex(myString, 100);\*/ |
|  |  |
|  | char str[1024]; |
|  | gets(str); |
|  | DumpHex(str, strlen(str)); |
|  |  |
|  |  |
|  |  |
|  | getchar(); |
|  | getchar(); |
|  | } |

## Лістинг програми Завдання 4.

|  |
| --- |
| #include <stdio.h> |
|  | #include <string.h> |
|  | #include <stdlib.h> |
|  |  |
|  |  |
|  | void main() { |
|  |  |
|  | printf("Enter the string: "); |
|  | char str[1024]; |
|  | gets(str); |
|  |  |
|  | printf("What character to replace? "); |
|  | char replace = getchar(); |
|  | //printf\_s("%c", replace); |
|  |  |
|  | printf("On which symbol to replace? "); |
|  | char prom = getchar(); |
|  | char which = getchar(); |
|  | //printf\_s("%c", which); |
|  |  |
|  |  |
|  | //Перебираем в цикле символы и заменяем если нужно |
|  | for (int i = 0; i < strlen(str); i++) { |
|  | if (str[i] == replace) { |
|  | str[i] = which; |
|  | } |
|  | } |
|  |  |
|  | printf("%s", str); |
|  |  |
|  |  |
|  | getchar(); |
|  | getchar(); |
|  | } |

## Лістинг програми Завдання 5.

|  |
| --- |
| #include <stdio.h> |
|  | #include <string.h> |
|  | #include <stdlib.h> |
|  |  |
|  |  |
|  | void main() { |
|  |  |
|  | printf("Enter the string: "); |
|  | char str[1024]; |
|  | gets(str); |
|  |  |
|  | //Цикл в котором выводим строку в зеркальном отображении |
|  | for (int i = strlen(str)-1; i >= 0; i--) { |
|  | printf("%c", str[i]); |
|  | } |
|  |  |
|  |  |
|  | getchar(); |
|  | getchar(); |
|  | } |

## Лістинг програми Завдання 6.

|  |
| --- |
| #include <stdio.h> |
|  | #include <string.h> |
|  | #include <stdlib.h> |
|  | //#include <fstream> |
|  | //#include <cstdio> |
|  | #define \_CRT\_SECURE\_NO\_WARNINGS |
|  | #pragma warning(disable:4996) |
|  |  |
|  |  |
|  |  |
|  | void main() { |
|  |  |
|  | FILE \*file; |
|  | FILE \*file2; |
|  | char arr[2048]; |
|  |  |
|  | //Открываем поток на считывание файла |
|  | file = fopen("fscanf.txt", "r"); |
|  |  |
|  | //Открываем поток на запись файла |
|  | file2 = fopen("out.txt", "w"); |
|  | int i = 0; |
|  | //Считываем пока не конец фала |
|  | while ((arr[i] = fgetc(file)) != EOF) { |
|  | if (arr[i] == '\n') { |
|  | arr[i] = '\0'; |
|  | printf("%s\n", arr); |
|  | //Если код символа находится в заданом промежутке, тогда пропускаем его |
|  | if (arr[i] >= 100 && arr[i] <= 110) |
|  | continue; |
|  | putc(arr[i], file2); |
|  | i = 0; |
|  | } |
|  | //Если нет - сохраняем в массив |
|  | else { |
|  | if (arr[i] >= 100 && arr[i] <= 110) |
|  | continue; |
|  | putc(arr[i], file2); |
|  | i++; |
|  | } |
|  | } |
|  | arr[i] = '\0'; |
|  | printf("%s\n", arr); |
|  |  |
|  | char prom; |
|  | //Записываем массив в новый файл |
|  | while (!feof(file)) { |
|  | prom = fgetc(file); |
|  | putc(prom, file2); |
|  | } |
|  |  |
|  |  |
|  | getchar(); |
|  | getchar(); |
|  | } |

## Лістинг програми Завдання 7.

|  |
| --- |
| #include <stdio.h> // printf |
|  | #include <stdlib.h> // malloc, free |
|  | #include <string.h> // strlen |
|  | #include <unistd.h> |
|  | #include <ctype.h> |
|  |  |
|  | char\* ReadFile(char \*filename) |
|  | { |
|  | char \*buffer = NULL; |
|  | int string\_size, read\_size; |
|  | FILE \*handler = fopen(filename, "r"); |
|  |  |
|  | if (handler) |
|  | { |
|  | // Seek the last byte of the file |
|  | fseek(handler, 0, SEEK\_END); |
|  | // Offset from the first to the last byte, or in other words, filesize |
|  | string\_size = ftell(handler); |
|  | // go back to the start of the file |
|  | rewind(handler); |
|  |  |
|  | // Allocate a string that can hold it all |
|  | buffer = (char\*) malloc(sizeof(char) \* (string\_size + 1) ); |
|  |  |
|  | // Read it all in one operation |
|  | read\_size = fread(buffer, sizeof(char), string\_size, handler); |
|  |  |
|  | // fread doesn't set it so put a \0 in the last position |
|  | // and buffer is now officially a string |
|  | buffer[string\_size] = '\0'; |
|  |  |
|  | if (string\_size != read\_size) |
|  | { |
|  | // Something went wrong, throw away the memory and set |
|  | // the buffer to NULL |
|  | free(buffer); |
|  | buffer = NULL; |
|  | } |
|  |  |
|  | // Always remember to close the file. |
|  | fclose(handler); |
|  | } |
|  |  |
|  | return buffer; |
|  | } |
|  |  |
|  | int main(int argn, char \*\*args) |
|  | { |
|  | char \*string = ReadFile(args[1]); |
|  |  |
|  | char \*encryptString = string; |
|  | char \*p; |
|  | int h = strtol(args[2], &p, 10); |
|  | for(int i=0;i<strlen(string);i++) |
|  | { |
|  | int bigNum = 0; |
|  | int numChar = encryptString[i]; |
|  | if(numChar>64 && numChar<91) |
|  | { |
|  | bigNum = 1; |
|  | numChar = tolower(numChar); |
|  | } |
|  | //printf("big num - %d",bigNum); |
|  | if(numChar<97 || numChar>122) |
|  | { |
|  | continue; |
|  | } |
|  |  |
|  | if(numChar+h>122) |
|  | { |
|  | int k = numChar + h - 122; |
|  | numChar=k+97; |
|  | } |
|  | else |
|  | { |
|  | numChar+=h; |
|  | } |
|  |  |
|  | char ch = numChar; |
|  | if(bigNum == 1) |
|  | { |
|  | ch = toupper(ch); |
|  | } |
|  | //printf("%d\n",h); |
|  | encryptString[i] = ch; |
|  | } |
|  |  |
|  |  |
|  | printf("%s",encryptString); |
|  | return 0; |
|  | } |

## Лістинг програми Завдання 8.

|  |
| --- |
| #include<stdio.h> |
|  | #include <stdlib.h> // malloc, free |
|  | #include <string.h> // strlen |
|  | #include <unistd.h> |
|  | #include <ctype.h> |
|  |  |
|  | char\* ReadFile(char \*filename) |
|  | { |
|  | char \*buffer = NULL; |
|  | int string\_size, read\_size; |
|  | FILE \*handler = fopen(filename, "r"); |
|  |  |
|  | if (handler) |
|  | { |
|  | // Seek the last byte of the file |
|  | fseek(handler, 0, SEEK\_END); |
|  | // Offset from the first to the last byte, or in other words, filesize |
|  | string\_size = ftell(handler); |
|  | // go back to the start of the file |
|  | rewind(handler); |
|  |  |
|  | // Allocate a string that can hold it all |
|  | buffer = (char\*) malloc(sizeof(char) \* (string\_size + 1) ); |
|  |  |
|  | // Read it all in one operation |
|  | read\_size = fread(buffer, sizeof(char), string\_size, handler); |
|  |  |
|  | // fread doesn't set it so put a \0 in the last position |
|  | // and buffer is now officially a string |
|  | buffer[string\_size] = '\0'; |
|  |  |
|  | if (string\_size != read\_size) |
|  | { |
|  | // Something went wrong, throw away the memory and set |
|  | // the buffer to NULL |
|  | free(buffer); |
|  | buffer = NULL; |
|  | } |
|  |  |
|  | // Always remember to close the file. |
|  | fclose(handler); |
|  | } |
|  |  |
|  | return buffer; |
|  | } |
|  |  |
|  | int main(int argn, char \*\*args) |
|  | { |
|  | char \*string = ReadFile(args[1]); |
|  | char \*word = args[2]; |
|  | char \*pch; |
|  | int isWordExist = 0; |
|  | pch = strtok (string," ,.-"); |
|  | while (pch != NULL) |
|  | { |
|  | //printf ("%s -- %s\n",pch,word); |
|  | if(strcmp(word,pch) == 0) |
|  | { |
|  | isWordExist = 1; |
|  | break; |
|  | } |
|  | pch = strtok (NULL, " ,.-"); |
|  | } |
|  |  |
|  | if(isWordExist == 0) |
|  | { |
|  | printf("Word is not exist in file\n"); |
|  | } |
|  | else |
|  | { |
|  | printf("Word is exist in file\n"); |
|  | } |
|  | } |

## Лістинг програми Завдання 9.

|  |
| --- |
| #include <stdio.h> |
|  | #include <string.h> |
|  | #include <stdlib.h> |
|  | #include <ctype.h> |
|  |  |
|  | #define DELIM " " |
|  | #define WORD "hello" |
|  | #define \_CRT\_SECURE\_NO\_WARNINGS |
|  | #pragma warning(disable:4996) |
|  |  |
|  | int main(void) |
|  | { |
|  | //Объявляем массив |
|  | char buf[BUFSIZ]; |
|  | /\*printf\_s("Enter the search word: "); |
|  | char \*WORD = NULL; |
|  | size\_t len; |
|  | if (getline(&buf, &len, stdin) == -1 && ferror(stdin)) |
|  | err(1, "getline");\*/ |
|  |  |
|  | //Открываем потоки чтения и записи |
|  | FILE \*f = fopen("test.txt", "r"); |
|  | FILE \*f1 = fopen("out.txt", "w"); |
|  |  |
|  | if (!f) { |
|  | fprintf(stderr, "Error open file\n"); |
|  | return 1; |
|  | } |
|  |  |
|  | if (!fgets(buf, BUFSIZ, f)) { |
|  | fprintf(stderr, "Error read file\n"); |
|  | return 1; |
|  | } |
|  | fclose(f); |
|  |  |
|  | //Объявляем массив |
|  | char out[BUFSIZ]; |
|  | int i = 0, j = 0; |
|  | //Считываем пока не конец файла |
|  | while (buf[i] != NULL) { |
|  | //Если число, то записываем в новый файл |
|  | if (isdigit(buf[i])) { |
|  | //out[j] = buf[i]; |
|  | //j++; |
|  | fputc(buf[i], f1); |
|  | } |
|  | i++; |
|  | } |
|  | //printf\_s("123"); |
|  |  |
|  | //system("pause"); |
|  | getchar(); |
|  | return 0; |
|  | } |

## Лістинг програми Завдання 10.

|  |
| --- |
| #include <stdio.h> |
|  | #include <string.h> |
|  | #include <stdlib.h> |
|  | #include <ctype.h> |
|  |  |
|  | #define DELIM " " |
|  | #define WORD "hello" |
|  | #define \_CRT\_SECURE\_NO\_WARNINGS |
|  | #pragma warning(disable:4996) |
|  |  |
|  | int main(void) |
|  | { |
|  | //Объявляем массив |
|  | char buf[BUFSIZ]; |
|  | /\*printf\_s("Enter the search word: "); |
|  | char \*WORD = NULL; |
|  | size\_t len; |
|  | if (getline(&buf, &len, stdin) == -1 && ferror(stdin)) |
|  | err(1, "getline");\*/ |
|  |  |
|  | //Открываем потоки чтения и записи |
|  | FILE \*f = fopen("test.txt", "r"); |
|  | FILE \*f1 = fopen("out.txt", "w"); |
|  |  |
|  | if (!f) { |
|  | fprintf(stderr, "Error open file\n"); |
|  | return 1; |
|  | } |
|  |  |
|  | if (!fgets(buf, BUFSIZ, f)) { |
|  | fprintf(stderr, "Error read file\n"); |
|  | return 1; |
|  | } |
|  | fclose(f); |
|  |  |
|  | //Объявляем массив |
|  | char out[BUFSIZ]; |
|  |  |
|  | int i = 0, j = 0; |
|  | while (buf[i] != NULL) { |
|  | //buf[i] = tolower(buf[i]); |
|  | //Убираем повторяющиеся символы |
|  | if ((buf[i] == buf[i+1]) && !isdigit(buf[i]) && !isalpha(buf[i])){ |
|  | i++; |
|  | continue; |
|  |  |
|  | } |
|  | //Записываем текст в новый файл |
|  | fputc(buf[i], f1); |
|  | i++; |
|  | } |
|  | //printf\_s("123"); |
|  |  |
|  | //system("pause"); |
|  | getchar(); |
|  | return 0; |
|  | } |

## Лістинг програми Завдання 11.

|  |
| --- |
| #include <stdio.h> // printf |
|  | #include <stdlib.h> // malloc, free |
|  | #include <string.h> // strlen |
|  |  |
|  | //function for rear all text from file |
|  | char\* ReadFile(char \*filename) |
|  | { |
|  | char \*buffer = NULL; |
|  | int string\_size, read\_size; |
|  | FILE \*handler = fopen(filename, "r"); |
|  |  |
|  | if (handler) |
|  | { |
|  | // Seek the last byte of the file |
|  | fseek(handler, 0, SEEK\_END); |
|  | // Offset from the first to the last byte, or in other words, filesize |
|  | string\_size = ftell(handler); |
|  | // go back to the start of the file |
|  | rewind(handler); |
|  |  |
|  | // Allocate a string that can hold it all |
|  | buffer = (char\*) malloc(sizeof(char) \* (string\_size + 1) ); |
|  |  |
|  | // Read it all in one operation |
|  | read\_size = fread(buffer, sizeof(char), string\_size, handler); |
|  |  |
|  | // fread doesn't set it so put a \0 in the last position |
|  | // and buffer is now officially a string |
|  | buffer[string\_size] = '\0'; |
|  |  |
|  | if (string\_size != read\_size) |
|  | { |
|  | // Something went wrong, throw away the memory and set |
|  | // the buffer to NULL |
|  | free(buffer); |
|  | buffer = NULL; |
|  | } |
|  |  |
|  | // Always remember to close the file. |
|  | fclose(handler); |
|  | } |
|  |  |
|  | return buffer; |
|  | } |
|  |  |
|  | int main(int argn, char \*\*args) |
|  | { |
|  | //text from file argument 1 |
|  | char \*string = ReadFile(args[1]); |
|  |  |
|  | //length off text |
|  | size\_t l = strlen(string); |
|  | //new array length l + 1 for nullterminator |
|  | char\* r = (char\*)malloc((l + 1) \* sizeof(char)); |
|  | r[l] = '\0'; |
|  |  |
|  | //make r = invert text |
|  | for(int i = 0; i < l; i++) |
|  | { |
|  | r[i] = string[l - 1 - i]; |
|  | } |
|  |  |
|  | //open stream for write r in new file |
|  | FILE \*f; |
|  | f = fopen(args[2], "w"); |
|  | fprintf(f,"%s",r); |
|  | fclose(f); |
|  | return 0; |
|  | } |

## Лістинг програми Завдання 12.

|  |
| --- |
| #include <stdio.h> // printf |
|  | #include <stdlib.h> // malloc, free |
|  | #include <string.h> // strlen |
|  | #include <unistd.h> |
|  |  |
|  | //function for reading text from file |
|  | char\* ReadFile(char \*filename) |
|  | { |
|  | char \*buffer = NULL; |
|  | int string\_size, read\_size; |
|  | FILE \*handler = fopen(filename, "r"); |
|  |  |
|  | if (handler) |
|  | { |
|  | // Seek the last byte of the file |
|  | fseek(handler, 0, SEEK\_END); |
|  | // Offset from the first to the last byte, or in other words, filesize |
|  | string\_size = ftell(handler); |
|  | // go back to the start of the file |
|  | rewind(handler); |
|  |  |
|  | // Allocate a string that can hold it all |
|  | buffer = (char\*) malloc(sizeof(char) \* (string\_size + 1) ); |
|  |  |
|  | // Read it all in one operation |
|  | read\_size = fread(buffer, sizeof(char), string\_size, handler); |
|  |  |
|  | // fread doesn't set it so put a \0 in the last position |
|  | // and buffer is now officially a string |
|  | buffer[string\_size] = '\0'; |
|  |  |
|  | if (string\_size != read\_size) |
|  | { |
|  | // Something went wrong, throw away the memory and set |
|  | // the buffer to NULL |
|  | free(buffer); |
|  | buffer = NULL; |
|  | } |
|  |  |
|  | // Always remember to close the file. |
|  | fclose(handler); |
|  | } |
|  |  |
|  | return buffer; |
|  | } |
|  |  |
|  | int main(int argn, char \*\*args) |
|  | { |
|  | //text from file argumet 1 |
|  | char \*string = ReadFile(args[1]); |
|  | //word from argument 2 |
|  | char \*word = args[2]; |
|  |  |
|  | //convert to int position for insert |
|  | char \*p; |
|  | int position = strtol(args[3], &p, 10); |
|  |  |
|  | //length of text |
|  | size\_t lstr = strlen(string); |
|  | //length of word |
|  | size\_t lword = strlen(word); |
|  | //new array length text+word+1 |
|  | char\* newString = (char\*)malloc((lstr + lword + 1) \* sizeof(char)); |
|  | newString[lstr+lword] = '\0'; |
|  |  |
|  | //insert word in text |
|  | int iterator=0; |
|  | if(lstr<position) |
|  | { |
|  | for(int i=0; i < lstr; i++) |
|  | { |
|  | newString[iterator] = string[i]; |
|  | iterator+=1; |
|  | } |
|  |  |
|  | for(int i=0; i<lword; i++) |
|  | { |
|  | newString[iterator] = word[i]; |
|  | iterator+=1; |
|  | } |
|  | } |
|  | else |
|  | { |
|  | for(int i=0; i<lstr;i++) |
|  | { |
|  | if(i == position) |
|  | { |
|  | for(int j=0; j<lword; j++) |
|  | { |
|  | newString[iterator] = word[j]; |
|  | iterator+=1; |
|  | } |
|  | } |
|  | newString[iterator] = string[i]; |
|  | iterator+=1; |
|  | } |
|  | } |
|  |  |
|  | //write new text in file argument 1 |
|  | FILE \*f; |
|  | f = fopen(args[1], "w"); |
|  | fprintf(f,"%s",newString); |
|  | fclose(f); |
|  | return 0; |
|  | } |

## Лістинг програми Завдання 13.

|  |
| --- |
| #include <stdio.h> |
|  | #include <math.h> |
|  |  |
|  | int main(int argn, char \*\*args) |
|  | { |
|  | //open stream for write data |
|  | FILE \*f; |
|  | f = fopen(args[1],"w"); |
|  |  |
|  |  |
|  | double start,end,h; |
|  | //get variables values |
|  | scanf("%lf%lf%lf",&start,&end,&h); |
|  |  |
|  | //write answer od function into file |
|  | for(double i = start;i<=end;i+=h) |
|  | { |
|  | double d = sin(i)+i-1; |
|  | fprintf(f,"%lf: sin(x)+x-1 = %lf\n",i,d); |
|  | } |
|  | fclose(f); |
|  | return 0; |
|  | } |

## Лістинг програми Завдання 14.

|  |
| --- |
| #include <stdio.h> // printf |
|  | #include <stdlib.h> // malloc, free |
|  | #include <string.h> // strlen |
|  |  |
|  | int main(int argn, char \*\*args) |
|  | { |
|  | int c; |
|  | //open stream for reading |
|  | FILE \*file; |
|  | file = fopen(args[1], "r"); |
|  |  |
|  | //open stream for writing |
|  | FILE \*f; |
|  | f = fopen(args[2],"w"); |
|  | if (file) |
|  | { |
|  | //write numeric format into new file |
|  | while ((c = getc(file)) != EOF) |
|  | { |
|  | fprintf(f,"%d\n",c); |
|  | } |
|  | fclose(file); |
|  | fclose(f); |
|  | } |
|  | return 0; |
|  | } |

## Лістинг програми Завдання 15.

|  |
| --- |
| #include <stdio.h> // printf |
|  | #include <stdlib.h> // malloc, free |
|  | #include <string.h> // strlen |
|  | #include <unistd.h> |
|  | #include <time.h> |
|  |  |
|  | //function for read text from text |
|  | char\* ReadFile(char \*filename) |
|  | { |
|  | char \*buffer = NULL; |
|  | int string\_size, read\_size; |
|  | FILE \*handler = fopen(filename, "r"); |
|  |  |
|  | if (handler) |
|  | { |
|  | // Seek the last byte of the file |
|  | fseek(handler, 0, SEEK\_END); |
|  | // Offset from the first to the last byte, or in other words, filesize |
|  | string\_size = ftell(handler); |
|  | // go back to the start of the file |
|  | rewind(handler); |
|  |  |
|  | // Allocate a string that can hold it all |
|  | buffer = (char\*) malloc(sizeof(char) \* (string\_size + 1) ); |
|  |  |
|  | // Read it all in one operation |
|  | read\_size = fread(buffer, sizeof(char), string\_size, handler); |
|  |  |
|  | // fread doesn't set it so put a \0 in the last position |
|  | // and buffer is now officially a string |
|  | buffer[string\_size] = '\0'; |
|  |  |
|  | if (string\_size != read\_size) |
|  | { |
|  | // Something went wrong, throw away the memory and set |
|  | // the buffer to NULL |
|  | free(buffer); |
|  | buffer = NULL; |
|  | } |
|  |  |
|  | // Always remember to close the file. |
|  | fclose(handler); |
|  | } |
|  |  |
|  | return buffer; |
|  | } |
|  |  |
|  | //function for comprahance text of 2 files |
|  | double fileComprahance(char \*file1, char \*file2) |
|  | { |
|  | //get length of file 1 |
|  | double l1 = strlen(file1); |
|  | //counter of equal symbols |
|  | double equalSymb = 0; |
|  |  |
|  | for(int i=0;i<l1-1;i++) |
|  | { |
|  | if(i == strlen(file2)) |
|  | { |
|  | break; |
|  | } |
|  |  |
|  | if(file1[i] == file2[i]) |
|  | { |
|  | equalSymb+=1; |
|  | } |
|  | } |
|  |  |
|  | //calculate pecent of equality |
|  | return (equalSymb/(l1-1)) \* 100; |
|  | } |
|  |  |
|  | int main(int argn, char \*\*args) |
|  | { |
|  | //get current time |
|  | time\_t t = time(NULL); |
|  | struct tm tm = \*localtime(&t); |
|  |  |
|  | printf("inspection date: %d-%d-%d %d:%d:%d\n", tm.tm\_year + 1900, tm.tm\_mon + 1, tm.tm\_mday, tm.tm\_hour, tm.tm\_min, tm.tm\_sec); |
|  |  |
|  | //read file 1 and file 2 |
|  | char \*file1 = ReadFile(args[1]); |
|  | char \*file2 = ReadFile(args[2]); |
|  |  |
|  | printf("Files: %s, %s\n",args[1],args[2]); |
|  |  |
|  | //search bigger file and comprahance them |
|  | if(strlen(file1)>strlen(file2)) |
|  | { |
|  | printf("Percentage ratio: %lf\n", fileComprahance(file1, file2)); |
|  | } |
|  | else |
|  | { |
|  | printf("Percentage ratio: %lf\n", fileComprahance(file2, file1)); |
|  | } |
|  | return 0; |
|  | } |