МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Кафедра Програмної інженерії

Звіт
з лабораторної роботи № 1
з дисципліни: «Операційні системи»
з теми: «Розробка універсальних додатків для різних типів кодувань символьної інформації»

Виконав:
ст. гр. ПЗПІ-23-2
Ситник Є. С.

Перевірила:
доц. каф. ПІ
Мельнікова Р. В.

Харків – 2025

1 РОЗРОБКА УНІВЕРСАЛЬНИХ ДОДАТКІВ ДЛЯ РІЗНИХ ТИПІВ КОДУВАНЬ СИМВОЛЬНОЇ ІНФОРМАЦІЇ

1.1 Мета роботи

Мета даної лабораторної роботи - навчитися опрацьовувати тексти для типів кодування ASCII та UNICODE, таким чином, щоб програма не залежала від обраного типу.

1.2 Передмова

Я користуюсь операційною системою Arch Linux, це накладає певні обмеження, а саме:

- я не можу використовувати інтегроване середовище розробки Visual Studio, замість цього я використовую редактор NeoVim;
- я не можу використовувати компілятор MSVC, замість цього я використовую GCC та MinGW (що забезпечує кросс-компіляцію);
- код я буду запускати за допомогою шару трансляції Wine, який забезпечує сумісність з Windows.

1.3 Хід роботи

1.3.1 Створіть консольний застосунок для C++ програм

Створення консольного застосунку на C++ відбувається в декілька етапів:

- створення файлу «Makefile» для спрощення збірки програми;
- створення файлу «main.cpp» в якому буде знаходитись код програми.

Makefile

```
1 main: main.cpp
2     x86_64-w64-mingw32-g++ -Wall -Wextra -o main.exe main.cpp
```

Рисунок 1.1 – Вміст файлу «Makefile»

Рисунок 1.2 – Вміст файлу «main.cpp»

1.3.2 Складіть оператори для програмної перевірки типу кодування, заданого за замовчуванням

Щоб дізнатись тип кодування, що задано за замовчуванням можна переглянути розмір типу «TCHAR» із заголовного файлу «tchar.h». Розмір цього типу буде дорівнювати 1 байту для ASCII та 2 байтам для UNICODE.

Рисунок 1.3 – Вміст файлу «main.cpp»

```
dumbnerd@Sentinel ~/D/N/O/1/code (master)> make && wine cmd /c main.exe
x86_64-w64-mingw32-g++ -Wall -Wextra -o main.exe main.cpp
TCHAR size: 1
dumbnerd@Sentinel ~/D/N/O/1/code (master)>
```

Рисунок 1.4 – Результат виконання програми

Можемо побачити, що розмір TCHAR дорівнює 1 байту, це означає, що за замовчуванням використовується кодування ASCII.

1.3.3 Визначте тип кодування по заданих макросах в командному рядку

Оскільки під час написання команди компіляції в «Makefile» я не додав визначення макросів, які б вплинули на кодування, компілятор використовує ASCII як кодування за замовчуванням.

1.3.4 Перемикніть режим завдання символу на протилежний

Щоб задати тип кодування необхідно визначити відповідні макроси. Для UNICODE це макроси «UNICODE» (для WinAPI) та «_UNICODE» (для стандартної бібліотеки), а для ASCII - «MBCS».

Змінимо існуючу ціль в «Makefile» щоб вона відповідала кодуванню ASCII, та додамо нову, що буде відповідати кодуванню UNICODE визначивши відповідні макроси.

Makefile

```
1 unicode: main.cpp
2     x86_64-w64-mingw32-g++ -Wall -Wextra -DUNICODE -D_UNICODE -o main.exe main.cpp
3
4 ascii: main.cpp
5     x86_64-w64-mingw32-g++ -Wall -Wextra -DMBCS -o main.exe main.cpp
```

Рисунок 1.5 – Вміст файлу «Makefile»

1.3.5 Після перемикання режиму знову перевірте тип символу за замовчуванням та командний рядок.

Перевіримо яке кодування використовує програма після проведених дій.

```
dumbnerd@Sentinel ~/D/N/O/1/code (master)> make ascii && wine cmd /c main.exe
x86_64-w64-mingw32-g++ -Wall -Wextra -DMBCS -o main.exe main.cpp
TCHAR size: 1
dumbnerd@Sentinel ~/D/N/O/1/code (master)> make unicode && wine cmd /c main.exe
x86_64-w64-mingw32-g++ -Wall -Wextra -DUNICODE -D_UNICODE -o main.exe main.cpp
TCHAR size: 2
dumbnerd@Sentinel ~/D/N/O/1/code (master)>
```

Рисунок 1.6 – Результат виконання програми

Можемо побачити, що після компіляції цілі «unicode» розмір типу «TCHAR» змінився на 2 байти, що підтверджує використання кодування UNICODE.

1.3.6 Задайте ПІБ членів своєї сім’ї в ASCII та виведіть задані значення

Щоб кириличні символи коректно друкувались в консолі необхідно виконати деякі дії, які будуть відрізнятись в залежності від кодування. Для кодування UNICODE необхідно встановити системну локаль «Ukrainian» (або іншу, що має кириличні символи), а для ASCII встановити кодову сторінку «CP1251».

Для встановлення системної локалі використаємо функцію «setlocale» з заголовного файлу «locale.h» який надає стандартна бібліотека, а для встановлення кодової сторінки використаємо функції «SetConsoleOutputCP» та «SetConsoleCP» з заголовного файлу «windows.h» який надає бібліотека WinAPI.

Також варто зазначити, що кириличні символи для ASCII відрізняються від аналогічних для UNICODE, тому для сумісності необхідно змінити кодування файлу із UTF-8 (яке є стандартним для Unix-подібних операційних систем таких як MacOS та Linux) на CP1251.

main.cpp

```cpp
#include <locale.h>
#include <stdio.h>
#include <tchar.h>
#include <windows.h>

int main() {
  // Зміна локалізації на українську для сумісності з UNICODE
  setlocale(LC_ALL, "Ukrainian");
  // Зміна кодової сторінки консолі на 1251 для сумісності з ASCII
  SetConsoleOutputCP(1251);
  SetConsoleCP(1251);

  int tchar_size = sizeof(TCHAR);
  printf("TCHAR size: %d\n", tchar_size);

  char ua_strings[3][60] = {"Ситник Єгор Сергійович",
                            "Ситник Ольга Володимирівна",
                            "Ситник Сергій Олексійович"};

  printf("\nua_strings:\n");
  for (int i = 0; i < 3; i++) {
    printf("%d: %s\n", i + 1, ua_strings[i]);
  }

  return 0;
}
```

Рисунок 1.7 – Вміст файлу «main.cpp»

```
dumbnerd@Sentinel ~/D/N/O/1/code (master)> make unicode && wine cmd /c main.exe
x86_64-w64-mingw32-g++ -Wall -Wextra -DUNICODE -D_UNICODE -o main.exe main.cpp
TCHAR size: 2

ua_strings:
1: Ситник Єгор Сергійович
2: Ситник Ольга Володимирівна
3: Ситник Сергій Олексійович
dumbnerd@Sentinel ~/D/N/O/1/code (master)>
```

Рисунок 1.8 – Результат виконання програми

1.3.7 Переведіть задані рядки в UNICODE за допомогою функції «MultiByteToWideChar»

Перевести рядок з ASCII в UNICODE можна за допомогою функції «MultiByteToWideChar» з бібліотеки WinAPI.

main.cpp

```
wchar_t ua_wstrings[3][60] = {};
for (int i = 0; i < 3; i++) {
  MultiByteToWideChar(CP_ACP, 0, ua_strings[i], -1, ua_wstrings[i], 60);
}
```

Рисунок 1.9 – Код доданий до файлу «main.cpp»

1.3.8 Виведіть отриманий масив кожним з трьох способів (функція _tprintf, потік wcout, функція MessageBox)

main.cpp

```
printf("\nua_wstrings:\n");
wchar_t ua_wstrings[3][60] = {};
for (int i = 0; i < 3; i++) {
  MultiByteToWideChar(CP_ACP, 0, ua_strings[i], -1, ua_wstrings[i], 60);

  printf("- - - - - - - - - - - - - - - - - - - - -\n");
  printf("%d:\n", i + 1);

  _tprintf(_T("_tprintf:\t%ls\n"), ua_wstrings[i]);
  std::wcout << L"std::wcout:\t" << ua_wstrings[i] << std::endl;
  MessageBoxW(NULL, ua_wstrings[i], L"MessageBox", MB_OK);
}
```

Рисунок 1.10 – Код доданий до файлу «main.cpp»

```
dumbnerd@Sentinel ~/D/N/O/l/code (master)> make unicode && wine cmd /c main.exe
x86_64-w64-mingw32-g++ -Wall -Wextra -DUNICODE -D_UNICODE -o main.exe main.cpp
TCHAR size: 2

ua_strings:
1: Ситник Єгор Сергійович
2: Ситник Ольга Володимирівна
3: Ситник Сергій Олексійович

ua_wstrings:
- - - - - - - - - - - - - - - - - - - - -
1:
_tprintf:       Ситник Єгор Сергійович
std::wcout:     Ситник Єгор Сергійович
- - - - - - - - - - - - - - - - - --
2:
_tprintf:       Ситник Ольга Володимирівна
std::wcout:     Ситник Ольга Володимирівна
- - - - - - - - - - - - - - - - - --
3:
_tprintf:       Ситник Сергій Олексійович
std::wcout:     Ситник Сергій Олексійович
```

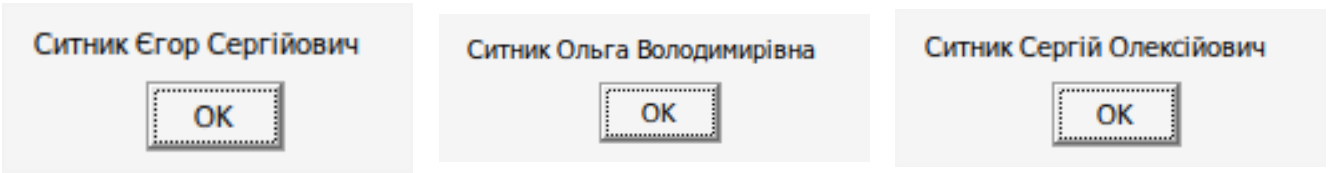

Рисунок 1.11 – Результат виконання програми

1.3.9 Виконайте упорядкування масиву рядків, заданих в UNICODE. Для сортування використовувати універсальну стандартну функцію «qsort» та шаблонну функцію «sort»

а) Функція qsort приймає такі аргументи:
   1) Вказівник на масив: Це адреса початку масиву, який потрібно відсортувати.
   2) Розмір масиву: Кількість елементів у масиві, що дозволяє функції знати, скільки елементів потрібно обробити.
   3) Розмір елемента масиву: Це розмір одного елемента масиву, який можна визначити, помноживши розмір типу «wchar_t» на кількість символів у рядку.

4) Вказівник на функцію порівняння: Функція, яка визначає порядок сортування. Вона повинна мати сигнатуру «int (*)(const void*, const void*)». Стандартна бібліотека C має багато функцій для порівняння рядків, найпоширенішою з яких є функція «strcmp» (та її аналог «wcscmp» для «wchat_t»), однак ця функція порівнює рядки без врахування локалізації, що зіпсує результат в певних випадках. Для порівняння рядків з урахуванням локалізації можна використовувати стандартну функцію «wcscoll». Сигнатура цієї функції відрізняється від сигнатури, яку очікує «qsort», в даному випадку буде достатньо привести тип функції «wcscoll» до такого, який очікує «qsort».

б) Функція sort приймає такі аргументи:

1) Вказівник на початок масиву: Адреса першого елемента масиву, який потрібно відсортувати. Функція sort не підтримує сортування багатовимірних масивів, тому масив, який вона обробляє, повинен бути одновимірним масивом вказівників на рядки, тому перед копіюванням рядків необхідно виділити пам’ять для кожного рядка, використовуючи функцію «malloc». Зазвичай, після виділення пам’яті програміст має звільнити її, коли вона більше не потрібна. Однак, оскільки система автоматично звільняє пам’ять після завершення виконання програми, а програма для лабораторної роботи завершується дуже швидко, виклик функції «free» для звільнення виділеної пам’яті не є обов’язковим.

2) Вказівник на кінець масиву: Адреса елемента, що йде після останнього елемента масиву, що дозволяє функції визначити межі масиву.

3) Вказівник на функцію порівняння: Функція, яка визначає порядок сортування. Вона повинна мати сигнатуру «bool (*)(wchar_t*, wchar_t*)». Для використання функції «wcscoll» з стандартної бібліотеки, її вихід потрібно перетворити, наприклад обгорнувши її в лямбда-функцію.

main.cpp

```cpp
// Копіювання масиву рядків для перевірки сортування
wchar_t ua_wstrings_to_qsort[3][60] = {};
wchar_t *ua_wstrings_to_sort[3] = {};
for (int i = 0; i < 3; i++) {
  // Копіювання масиву для qsort
  wcscpy(ua_wstrings_to_qsort[i], ua_wstrings[i]);

  // Копіювання масиву для std::sort
  ua_wstrings_to_sort[i] = (wchar_t *)malloc(sizeof(wchar_t) * 60);
  wcscpy(ua_wstrings_to_sort[i], ua_wstrings[i]);
}
```

Рисунок 1.12 – Код для копіювання рядків

main.cpp

```cpp
printf("\nqsort:\n");
qsort(ua_wstrings_to_qsort, 3, sizeof(wchar_t) * 60,
      (int (*)(const void *, const void *))wcscmp);
for (int i = 0; i < 3; i++) {
  _tprintf(_T("%ls\n"), ua_wstrings_to_qsort[i]);
}
```

Рисунок 1.13 – Код для перевірки функції «qsort»

main.cpp

```cpp
printf("\nstd::sort:\n");
std::sort(ua_wstrings_to_sort, ua_wstrings_to_sort + 3,
          [](const wchar_t *a, const wchar_t *b) -> bool {
            return wcscmp(a, b) < 0;
          });
for (int i = 0; i < 3; i++) {
  _tprintf(_T("%ls\n"), ua_wstrings_to_sort[i]);
}
```

Рисунок 1.14 – Код для перевірки функції «sort»

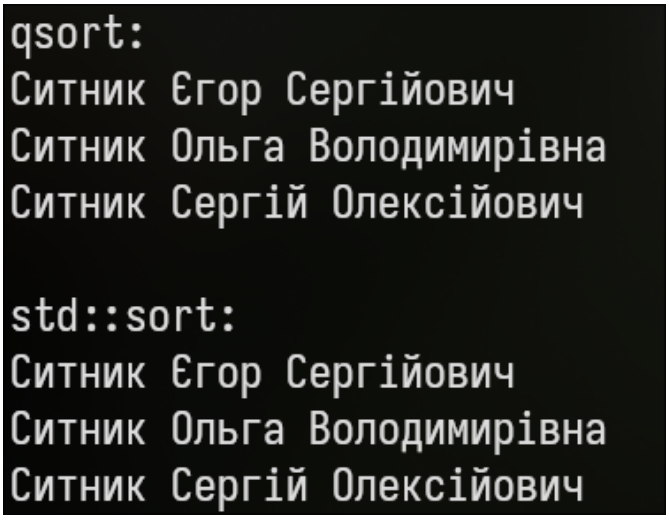

Рисунок 1.15 – Результат виконання програми

1.3.10 Виконайте зворотне перетворення масиву з Unicode в ASCII

Перевести рядок з UNICODE в ASCII можна за допомогою функції «WideCharToMultiByte» з бібліотеки WinAPI.

main.cpp

```
char ua_strings_from_wstrings[3][60] = {};
for (int i = 0; i < 3; i++) {
  WideCharToMultiByte(CP_ACP, 0, ua_wstrings[i], -1,
                      ua_strings_from_wstrings[i], 60, NULL, NULL);
}
```

Рисунок 1.16 – Код доданий до файлу «main.cpp»

1.3.11 Виведіть отриманий результат

main.cpp

```
printf("\nua_strings_from_wstrings:\n");
char ua_strings_from_wstrings[3][60] = {};
for (int i = 0; i < 3; i++) {
  WideCharToMultiByte(CP_ACP, 0, ua_wstrings[i], -1,
                      ua_strings_from_wstrings[i], 60, NULL, NULL);

  printf("%d: %s\n", i + 1, ua_strings_from_wstrings[i]);
}
```

Рисунок 1.17 – Код доданий до файлу «main.cpp»

```
ua_strings_from_wstrings:
1: Ситник Сергій Олексійович
2: Ситник Єгор Сергійович
3: Ситник Ольга Володимирівна
```

Рисунок 1.18 – Результат виконання програми

1.3.12 Задано текстовий файл. Не залежно від способу кодування символів в цьому файлі, переставити символи в кожному рядку в зворотному порядку.

main.cpp

```
void reverse_file(const char *input_filename, const char *output_filename) {
  FILE *input_file = fopen(input_filename, "r");
  if (input_file == NULL) {
    fprintf(stderr, "Cant open input file '%s'\n", input_filename);
    return;
  }

  FILE *output_file = fopen(output_filename, "w");
  if (output_file == NULL) {
    fprintf(stderr, "Cant create output file '%s'\n", output_filename);
    fclose(input_file);
    return;
  }

  int codepage = get_codepage(input_filename);

  char buffer[4096];

  while (fgets(buffer, sizeof(buffer), input_file) != NULL) {
    size_t len = strlen(buffer);

    if (len > 0 && buffer[len - 1] == '\n') {
      buffer[len - 1] = '\0';
      len--;
    }

    reverse_string(buffer, len, codepage);

    fprintf(output_file, "%s\n", buffer);
  }

  fclose(input_file);
  fclose(output_file);

  printf("Successfully reversed file '%s' to '%s'\n", input_filename,
         output_filename);
};
```

Рисунок 1.19 – Функція, яка створює копію файлу з розвернутими рядками

main.cpp

```c
void reverse_string(char *str, size_t len, int codepage) {
  if (str == NULL || len == 0)
    return;

  wchar_t *wstr = (wchar_t *)malloc(sizeof(wchar_t) * (len + 1));
  if (wstr == NULL) {
    fprintf(stderr, "Error encountered while allocating memory\n");
    return;
  }

  MultiByteToWideChar(codepage, 0, str, len + 1, wstr, len + 1);
  size_t wlen = wcslen(wstr);

  for (size_t i = 0, j = wlen - 1; i < j; i++, j--) {
    wchar_t temp = wstr[i];
    wstr[i] = wstr[j];
    wstr[j] = temp;
  }

  WideCharToMultiByte(codepage, 0, wstr, len, str, len, NULL, NULL);

  free(wstr);
}
```

Рисунок 1.20 – Функція, яка розвертає рядок

main.cpp

```c
int get_codepage(const char *filename) {
  FILE *file = fopen(filename, "r");
  if (file == NULL) {
    fprintf(stderr, "Cant open file '%s'\n", filename);
    return -1;
  }

  int codepage;

  if (fgetc(file) == 0xEF && fgetc(file) == 0xBB && fgetc(file) == 0xBF) {
    codepage = CP_UTF8;
  } else {
    codepage = CP_ACP;
  }

  fclose(file);

  return codepage;
}
```

Рисунок 1.21 – Функція, яка визначає кодову сторінку файлу

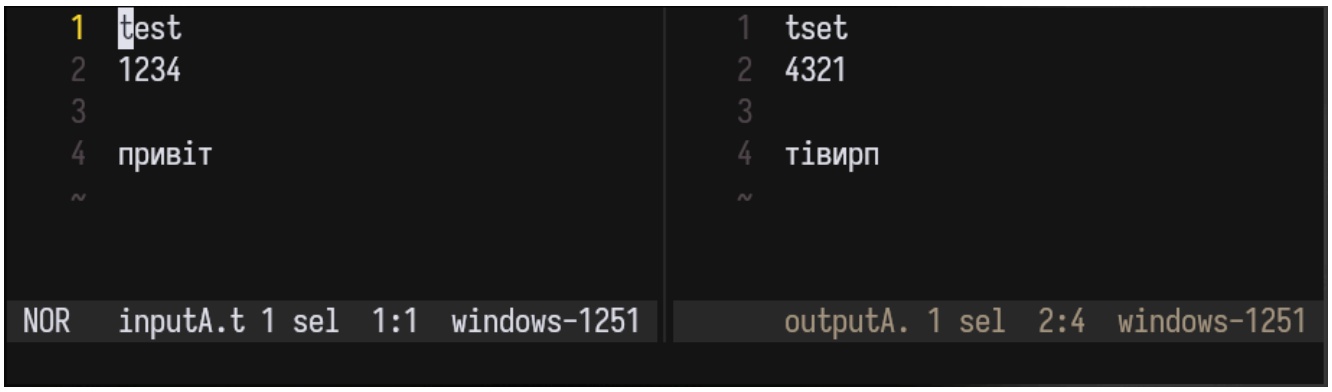

Рисунок 1.22 – Результат перетворення ASCII файлу

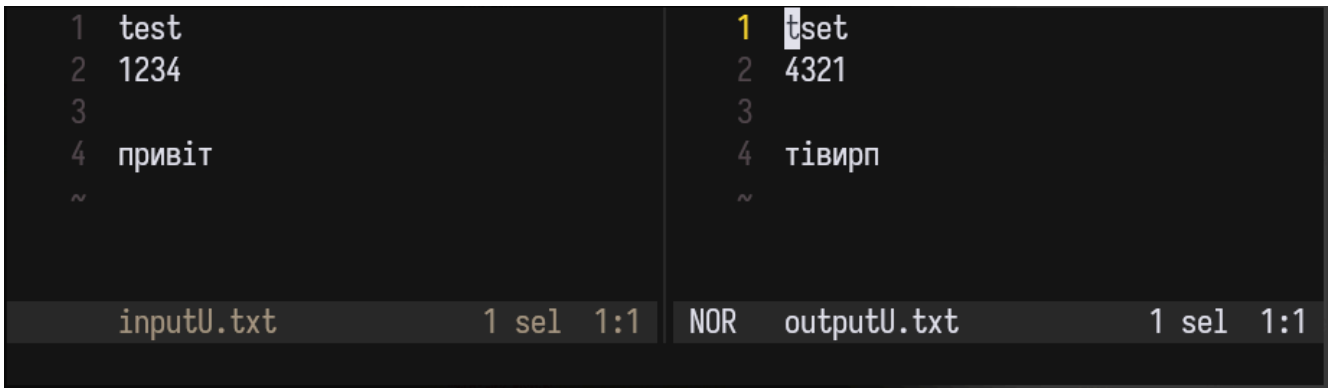

Рисунок 1.23 – Результат перетворення UNICODE файлу

1.4 Висновки

Під час виконання даної лабораторної роботи я навчився опрацьовувати тексти для типів кодування ASCII та UNICODE, таким чином, щоб програма не залежала від обраного типу.

ДОДАТОК А

Вміст файлу «main.cpp»

```
#include <algorithm>
#include <iostream>
#include <locale.h>
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <stringapiset.h>
#include <tchar.h>
#include <wchar.h>
#include <windows.h>

void reverse_file(const char *input_filename, const char
*output_filename);
void reverse_string(char *str, size_t len, int codepage);
int get_codepage(const char *filename);

int main() {
  // Зміна локалізації на українську для сумісності з UNICODE
  setlocale(LC_ALL, "Ukrainian");
  // Зміна кодової сторінки консолі на 1251 для сумісності з ASCII
  SetConsoleOutputCP(1251);
  SetConsoleCP(1251);

  int tchar_size = sizeof(TCHAR);
  printf("TCHAR size: %d\n", tchar_size);

  char ua_strings[3][60] = {"Ситник Сергій Олексійович",
                            "Ситник Єгор Сергійович",
                            "Ситник Ольга Володимирівна"};

  printf("\nua_strings:\n");
  for (int i = 0; i < 3; i++) {
    printf("%d: %s\n", i + 1, ua_strings[i]);
  }
```

```
  printf("\nua_wstrings:\n");
  wchar_t ua_wstrings[3][60] = {};
  for (int i = 0; i < 3; i++) {
    MultiByteToWideChar(CP_ACP, 0, ua_strings[i], -1, ua_wstrings[i],
60);

    printf("- - - - - - - - - - - - - - - - - - - - - -\n");
    printf("%d:\n", i + 1);

    _tprintf(_T("_tprintf:\t%ls\n"), ua_wstrings[i]);
    std::wcout << L"std::wcout:\t" << ua_wstrings[i] << std::endl;
    // MessageBoxW(NULL, ua_wstrings[i], L"MessageBox", MB_OK);
  }

  // Копіювання масиву рядків для перевірки сортування
  wchar_t ua_wstrings_to_qsort[3][60] = {};
  wchar_t *ua_wstrings_to_sort[3] = {};
  for (int i = 0; i < 3; i++) {
    // Копіювання масиву для qsort
    wcscpy(ua_wstrings_to_qsort[i], ua_wstrings[i]);

    // Копіювання масиву для std::sort
    ua_wstrings_to_sort[i] = (wchar_t *)malloc(sizeof(wchar_t) * 60);
    wcscpy(ua_wstrings_to_sort[i], ua_wstrings[i]);
  }

  printf("\nqsort:\n");
  qsort(ua_wstrings_to_qsort, 3, sizeof(wchar_t) * 60,
        (int (*)(const void *, const void *))wcscoll);
  for (int i = 0; i < 3; i++) {
    _tprintf(_T("%ls\n"), ua_wstrings_to_qsort[i]);
  }

  printf("\nstd::sort:\n");
  std::sort(ua_wstrings_to_sort, ua_wstrings_to_sort + 3,
            [](const wchar_t *a, const wchar_t *b) -> bool {
              return wcscoll(a, b) < 0;
            });
```

```
  for (int i = 0; i < 3; i++) {
    _tprintf(_T("%ls\n"), ua_wstrings_to_sort[i]);
  }

  printf("\nua_strings_from_wstrings:\n");
  char ua_strings_from_wstrings[3][60] = {};
  for (int i = 0; i < 3; i++) {
    WideCharToMultiByte(CP_ACP, 0, ua_wstrings[i], -1,
                        ua_strings_from_wstrings[i], 60, NULL, NULL);

    printf("%d: %s\n", i + 1, ua_strings_from_wstrings[i]);
  }

  printf("\nfile convertions:\n");
  reverse_file("inputA.txt", "outputA.txt");
  reverse_file("inputU.txt", "outputU.txt");

  return 0;
}

void reverse_string(char *str, size_t len, int codepage) {
  if (str == NULL || len == 0)
    return;

  wchar_t *wstr = (wchar_t *)malloc(sizeof(wchar_t) * (len + 1));
  if (wstr == NULL) {
    fprintf(stderr, "Error encountered while allocating memory\n");
    return;
  }

  MultiByteToWideChar(codepage, 0, str, len + 1, wstr, len + 1);
  size_t wlen = wcslen(wstr);

  for (size_t i = 0, j = wlen - 1; i < j; i++, j--) {
    wchar_t temp = wstr[i];
    wstr[i] = wstr[j];
    wstr[j] = temp;
  }
```

```
  WideCharToMultiByte(codepage, 0, wstr, len, str, len, NULL, NULL);

  free(wstr);
}

void reverse_file(const char *input_filename, const char
*output_filename) {
  FILE *input_file = fopen(input_filename, "r");
  if (input_file == NULL) {
    fprintf(stderr, "Cant open input file '%s'\n", input_filename);
    return;
  }

  FILE *output_file = fopen(output_filename, "w");
  if (output_file == NULL) {
    fprintf(stderr, "Cant create output file '%s'\n", output_filename);
    fclose(input_file);
    return;
  }

  int codepage = get_codepage(input_filename);

  char buffer[4096];

  while (fgets(buffer, sizeof(buffer), input_file) != NULL) {
    size_t len = strlen(buffer);

    if (len > 0 && buffer[len - 1] == '\n') {
      buffer[len - 1] = '\0';
      len--;
    }

    reverse_string(buffer, len, codepage);

    fprintf(output_file, "%s\n", buffer);
  }
```

```
  fclose(input_file);
  fclose(output_file);

  printf("Successfully reversed file '%s' to '%s'\n", input_filename,
         output_filename);
};

int get_codepage(const char *filename) {
  FILE *file = fopen(filename, "r");
  if (file == NULL) {
    fprintf(stderr, "Cant open file '%s'\n", filename);
    return -1;
  }

  int codepage;

  if (fgetc(file) == 0xEF && fgetc(file) == 0xBB && fgetc(file) == 0xBF)
{
    codepage = CP_UTF8;
  } else {
    codepage = CP_ACP;
  }

  fclose(file);

  return codepage;
}
```

ДОДАТОК Б

Вміст файлу «Makefile»

```
unicode: main.cpp
  x86_64-w64-mingw32-g++ -Wall -Wextra -DUNICODE -D_UNICODE -o main.exe
main.cpp

ascii: main.cpp
  x86_64-w64-mingw32-g++ -Wall -Wextra -DMBCS -o main.exe main.cpp
```