Отчёт

по лабораторной работе №1

по теме «Разработка универсальных приложений для разных типов кодировок символьной информации»

по предмету «Операционные системы»

студентов группы ПИ-14-4

Шутеева Ильи и Гладуша Ивана

Цель: Научиться обрабатывать тексты для таких типов кодировок, как ASCII и UNICODE, причём сама программа не должна зависеть от выбранного способа.

Для обеспечения универсальности кодировки используются типы char и wchar\_t. Тип char используется при кодировке ASCII и он занимает в памяти один байт. Тип wchat\_t используется при использовании кодировки UNICODE. Этому типу данных выделяется два байта. Соответственно: 1byte = 8 bit, значит переменная типа char может принимать 256 разных значений; 2 byte = 16 bit, значит переменная типа wchar\_t может принимать 65536 разных значений.

Для обеспечения универсальности кодировки используются следующие функции:

1. TCHAR \* \_tsetlocale ( int category, const TCHAR \*locale ) – функция установки локального режима. Необходима для выведения в консоль символов Кириллицы. Она принимает следующие параметры:

* category – Категория, на которую влияет языковой стандарт (обычно LC\_ALL).
* Locale – Указатель языкового стандарта. Используется \_T("Russian") для русского языка и \_T("Ukrainian") для украинского.

1. int MultiByteToWideChar (UINT CodePage, DWORD dwFlags, LPCSTR lpMultiByteStr, int cbMultiByte, LPWSTR lpWideCharStr, int cchWideChar) – функция для конвертации массива char символов в массив wchar\_t символов. Функция возвращает количество символов в ряде-результате.
   * CodePage – кодовая страница, обычно задаётся CP\_ACP.
   * dwFlags ­– флаги, указывающие тип преобразования. Обычно устанавливается значение 0.
   * lpMultiByteStr – строка, которую мы преобразовываем.
   * cbMultiByte – размер в байтах преобразуемой строки. Если стоит данный параметр равен -1, то обрабатывается вся строка.
   * lpWideCharStr – буфер, в который мы сохраняем преобразованную строку.
   * cchWideChar – количество символов в буфере lpWideCharStr.
2. int WideCharToMultiByte (UINT CodePage, DWORD dwFlags, LPWSTR lpWideCharStr, int cchWideChar, LPCSTR lpMultiByteStr, int cbMultiByte, LPCSTR lpDefaultChar, LPBOOL lpUserDefaultChar) – функция для конвертации массива wchar\_t символов в массив char символов. Функция возвращает количество символов в ряде-результате.
   * CodePage – кодовая страница, обычно задаётся CP\_ACP.
   * dwFlags ­– флаги, указывающие тип преобразования. Обычно устанавливается значение 0.
   * lpMultiByteStr – строка, которую мы преобразовываем.
   * cchWideChar – количество символов в lpWideCharStr.
   * lpWideCharStr – буфер, в который мы сохраняем преобразованную строку.
   * cbMultiByte – размер в байтах конечного результата.
   * lpDefaultChar – Адрес символа, которым заменяется символ, который преобразуется, если он не может быть отображён. Равен NULL, если используется значение по умолчанию.
   * lpUserDefaultChar – указатель на флажок, который указывает на то, используется ли символ по умолчанию в предыдущем параметре. Может быть NULL.

Функции, которые используются для вывода информации в консольном режиме на русском языке:

1. cout – поток вывода байтов.
2. wprintf(const wchar\_t \*Text) – Распечатка в консоли строки Text.
3. MessageBoxW(HWND    hWnd,LPCTSTR lpText, LPCTSTR lpCaption, UINT    uType) – Вывод информации во всплывающем окне.

Текст программы:   
  
// Lab1.cpp: определяет точку входа для консольного приложения.

//

#include <iostream>

#include "stdafx.h"

#include <locale.h>

#include <windows.h>

#include <fstream>

using namespace std;

// 1. Программа для определения кодировки символов по умолчанию.

char\* GetProgramEncoding()

{

if (typeid(char).name() == typeid(TCHAR).name())

{

return "ANCII";

}

else

{

return "Unicode";

}

//-----OR------

/\*if (sizeof(TCHAR) == 1)

{

return "ANCII";

}

else

{

return "Unicode";

}\*/

}

int wcompare(void const \* arg1, void const \* arg2)

{

int n = lstrcmpW(\*(const wchar\_t\*\*)arg1, \*(const wchar\_t\*\*)arg2);

return n;

}

// Объединение двух строк в третью (wchar\_t).

void concatW(wchar\_t\* buffer, wchar\_t\* text1, wchar\_t\* text2)

{

int i = 0;

int len1 = wcslen(text1);

int len2 = wcslen(text2);

for ( ; i < len1; i++)

{

buffer[i] = text1[i];

}

for (; i < len1 + len2; i++)

{

buffer[i] = text2[i - len1];

}

}

// Объединение двух строк в третью (char).

void concatC(char\* buffer, char\* text1, char\* text2)

{

int i = 0;

int len1 = strlen(text1);

int len2 = strlen(text2);

for (; i < len1; i++)

{

buffer[i] = text1[i];

}

for (; i < len1 + len2; i++)

{

buffer[i] = text2[i - len1];

}

}

// Переворот мтроки (wchar\_t).

void reverseW(wchar\_t\* buffer, wchar\_t\* text1)

{

int len = wcslen(text1);

for (int i = 0; i < len - 1; i++)

{

// В данном случае вычитаем 2, потому что мы символ перехода

// на новую строку и символ конца файла оставляем на своём месте.

buffer[i] = text1[len - i - 2];

}

// Добавляем символ перехода на новую строку в конец строки.

if (text1[len - 1] == '\n')

buffer[len - 1] = text1[len - 1];

}

// Переворот мтроки (char).

void reverseC(char\* buffer, char\* text1)

{

int len = strlen(text1);

// В данном случае вычитаем 2, потому что мы переход на новую строку оставляем на своём месте.

if (text1[len - 1] == '\n')

{

for (int i = 0; i < len - 1; i++)

{

buffer[i] = text1[len - i - 2];

}

// Добавляем переход на новую строку в конце файла.

buffer[len - 1] = '\n';

}

else

{

for (int i = 0; i < len; i++)

{

buffer[i] = text1[len - i - 1];

}

}

}

void ReverceText(char\* path)

{

FILE \*file;

// Проверяем, какая кодировка в файл.

fopen\_s(&file, path, "rt");

// Первый байт метки.

char BOM1 = fgetc(file);

// Второй байт метки.

char BOM2 = fgetc(file);

// Третий байт метки.

char BOM3 = fgetc(file);

fclose(file);

// Случай для UTF-16LE

if (BOM1 == '\xff' && BOM2 == '\xfe')

{

fopen\_s(&file, path, "rt+, ccs=UTF-16LE");

wchar\_t text[30000] = L"";

while (!feof(file))

{

wchar\_t str1[256] = L"";

wchar\_t str2[256] = L"";

wchar\_t w = ' ';

for (int i = 0; w != '\n' && !feof(file); i++)

{

w = fgetwc(file);

str1[i] = w;

}

//fgetws(str1, 256, file);

reverseW(str2, str1);

concatW(text, text, str2);

}

fclose(file);

// Открываем файл для записи.

fopen\_s(&file, path, "wt, ccs=UTF-16LE");

for (int i = 0; i < wcslen(text); i++)

{

fputwc(text[i], file);

}

fclose(file);

}

// UTF-16 BigEndian

//else if (BOM1 == '\xfe' && BOM2 == '\xff')

//{

// fopen\_s(&file, path, "rt+");

// // Пропускаем первые четыре символа (BOM + '\0' + '\0')

// for (int i = 0; i < 2; i++)

// fgetwc(file);

// wchar\_t text[30000] = L"";

// while (!feof(file))

// {

// wchar\_t str1[256] = L"";

// wchar\_t str2[256] = L"";

// wchar\_t w = ' ';

// for (int i = 0; w != '\n' && !feof(file); i++)

// {

// w = fgetwc(file);

// // (костыльно)

// if (w != '\0')

// str1[i] = w;

// else

// i--;

// }

// //fgetws(str1, 256, file);

// reverseW(str2, str1);

// concatW(text, text, str2);

// }

// fclose(file);

// // Открываем файл для записи.

// fopen\_s(&file, path, "wt");

// for (int i = 0; i < wcslen(text); i++)

// {

// fputwc(text[i], file);

// }

// fclose(file);

//}

// UTF-8

else if (BOM1 == '\xef' && BOM2 == '\xbb' && BOM3 == '\xbf')

{

fopen\_s(&file, path, "rt+, ccs=UTF-8");

wchar\_t text[30000] = L"";

while (!feof(file))

{

wchar\_t str1[256] = L"";

wchar\_t str2[256] = L"";

wchar\_t w = ' ';

for (int i = 0; w != '\n' && !feof(file); i++)

{

w = fgetwc(file);

str1[i] = w;

}

//fgetws(str1, 256, file);

reverseW(str2, str1);

concatW(text, text, str2);

}

fclose(file);

// Открываем файл для записи.

fopen\_s(&file, path, "wt, ccs=UTF-8");

for (int i = 0; i < wcslen(text); i++)

{

fputwc(text[i], file);

}

fclose(file);

}

// Случай для ASCII

else

{

fopen\_s(&file, path, "rt");

char text[30000] = "";

while (!feof(file))

{

char str1[256] = "";

char str2[256] = "";

fgets(str1, 256, file);

reverseC(str2, str1);

concatC(text, text, str2);

}

fclose(file);

// Открываем файл для записи.

fopen\_s(&file, path, "wt");

fprintf(file, text);

fclose(file);

}

}

int main()

{

// Устанавливаем локальный режим для того, чтобы выводить символы кирилицы.

\_tsetlocale(LC\_ALL, \_T("Russian"));

// 1. Тип кодировки по умолчанию (ASCII, потому что в настройках был задан этот тип кодировки).

cout << "Тип кодировки по умолчанию: ";

cout << GetProgramEncoding() << endl;

// 2. Определение типа кодировки по заданному макросу в командной строке.

// Properties → C/C++ → Command Line (Unicode)

// 3. Переключение режима задания символа на ASCII.

// Properties → General → Character Set → Use Multi-Byte Character Set

// 4. Проверка типа кодировки по заданному макросу в командной строке (после компиляции).

// Properties → C/C++ → Command Line (Unicode)

// 5.

char asciiNames[3][256] =

{

"Шутеев Илья Вячеславович",

"Шутеева Татьяна Николаевна",

"Шутеев Вячеслав Вадимович"

};

cout << "Объявлен массив в ASCII кодировке:" << endl;

for (int i = 0; i < 3; i++)

{

cout << asciiNames[i] << endl;

}

cout << "\n";

// 6.

wchar\_t unicodeNames[3][256] = { L"",L"",L"" };

for (int i = 0; i < 3; i++)

{

int len = MultiByteToWideChar(

CP\_ACP,

0,

asciiNames[i],

-1,

NULL,

0

);

MultiByteToWideChar(

CP\_ACP,

0,

asciiNames[i],

-1,

unicodeNames[i],

len

);

}

// 7.

cout << "Массив в UNICODE кодировке:" << endl;

for (int i = 0; i < 3; i++)

{

//\_tprintf работает только с типом char.

wprintf(unicodeNames[i]);

wprintf(L"\n");

// MessageBox работает только с типом char.

MessageBoxW(0, unicodeNames[i], L"Name", MB\_OK);

}

cout << "\n";

// 8.

qsort(unicodeNames, 3, sizeof(wchar\_t\*), wcompare);

// 9.

for (int i = 0; i < 3; i++)

{

int len = WideCharToMultiByte(

CP\_ACP,

0,

unicodeNames[i],

-1,

NULL,

0,

NULL,

NULL

);

WideCharToMultiByte(

CP\_ACP,

0,

unicodeNames[i],

-1,

asciiNames[i],

len,

NULL,

NULL

);

}

// 10.

cout << "Отсортированный массив в ASCII кодировке:" << endl;

for (int i = 0; i < 3; i++)

{

cout << asciiNames[i] << endl;

}

//11. Считывание из файла. Для некоторых вариантов работает некорректно.

// D:\\ОС\\Лабы\\Lab1\\Debug\\111.txt

//ReverceText("D:\\ОС\\Лабы\\Lab1\\Debug\\UTF-16BE.txt");

ReverceText("D:\\ОС\\Лабы\\Lab1\\Debug\\UTF-8.txt");

ReverceText("D:\\ОС\\Лабы\\Lab1\\Debug\\UTF-16LE.txt");

ReverceText("D:\\ОС\\Лабы\\Lab1\\Debug\\ASCII.txt");

}

Выводы:

В ходе лабораторной работы я научился обрабатывать тексты для таких типов кодировок, как ASCII и UNICODE. Разработанная мной программа для переставления символов в файле в обратном порядке, которая не зависит от кодировки, в которой закодирован текст файла.