Лабораторная работа 2. Работа со строками. Использование ANSI, UNICODE строк в Windows-программах

Теоретические сведения:

**Тема: Работа со строками. Использование ANSI, UNICODE строк в Windows-программах, преобразование, вывод. Обработка ошибок в Win32.**

**1. Кодировки символов: ANSI и Unicode**

В Windows существует два основных представления символов и строк:

* **ANSI (Multi-Byte Character Set - MBCS):**
  + Использует тип данных char.
  + Каждый символ занимает 1 байт.
  + Для кодирования национальных алфавитов используется кодовая страница (Code Page), что приводит к проблемам с совместимостью между разными языковыми системами.
  + Строковые литералы записываются в двойных кавычках: "Это ANSI-строка".
* **Unicode (Wide Character Set):**
  + Использует тип данных wchar\_t, который занимает 2 байта (UTF-16LE в Windows).
  + Позволяет представлять практически все символы всех письменностей мира в одной кодировке.
  + Является стандартом для современных Windows-приложений.
  + Строковые литералы предваряются буквой L: L"Это Unicode-строка".
* **UTF-8:**
  + Это популярная кодировка Unicode с переменной длиной символа (от 1 до 4 байт).
  + В WinAPI для представления UTF-8 используется тот же тип char, что и для ANSI, но с другой кодовой страницей (CP\_UTF8).
  + Строковые литералы для UTF-8 в чистом виде в C++ не поддерживаются, но их можно создавать с помощью преобразований.

**2. Макросы Windows.h для кроссплатформенности**

Библиотека Windows.h предоставляет набор макросов и типов данных для упрощения написания кода, который можно скомпилировать как в ANSI, так и в Unicode-варианте.

* **TCHAR:** Условный тип символа. Заменяется на char при сборке с \_MBCS и на wchar\_t при сборке с \_UNICODE.
* **LPSTR / LPCSTR:** Указатель на ANSI-строку (char\* / const char\*).
* **LPWSTR / LPCWSTR:** Указатель на Unicode-строку (wchar\_t\* / const wchar\_t\*).
* **LPTSTR / LPCTSTR:** Условный указатель на строку. Заменяется на LPSTR/LPCSTR (ANSI) или LPWSTR/LPCWSTR (Unicode) в зависимости от настроек проекта.
* **Текстовые литералы:** Макрос \_T() или TEXT() используется для условных строковых литералов.  
  TEXT("Hello") будет развернут в "Hello" для ANSI и в L"Hello" для Unicode.

**3. Преобразование между кодировками**

WinAPI предоставляет функции для преобразования строк:

* **MultiByteToWideChar:** Преобразует ANSI-строку в Unicode-строку.
* **WideCharToMultiByte:** Преобразует Unicode-строку в ANSI-строку (или в строку в любой другой кодовой странице, например, UTF-8).

**4. Вывод строк в Windows**

* **Для консольных приложений:**
  + printf(), cout — для вывода ANSI-строк (char\*).
  + wprintf(), wcout — для вывода Unicode-строк (wchar\_t\*).
* **Для GUI-приложений:**
  + Функции наподобие MessageBox имеют две версии: MessageBoxA (ANSI) и MessageBoxW(Unicode). Обычно мы используем макрос MessageBox, который подставляется в нужную версию.

**5. Обработка ошибок в Win32**

Почти все функции WinAPI возвращают код ошибки в случае неудачи.

* **Функция GetLastError():** Возвращает код последней ошибки, возникшей при вызове API-функции.
* **Функция FormatMessage():** Позволяет преобразовать код ошибки, полученный от GetLastError(), в читаемое текстовое сообщение. Это основной инструмент для диагностики проблем.

**Задание к лабораторной работе**

Написать указанные функции в двух вариантах: для ANSI-строк (используя char) и для строк Unicode (используя wchar\_t).

**1. Функция addslashes**

cpp

*// ANSI-версия*

void addslashes\_a(char\* str) {

*// ... код функции ...*

}

*// Unicode-версия*

void addslashes\_w(wchar\_t\* str) {

*// ... код функции ...*

}

**Алгоритм:** Необходимо пройти по исходной строке, и перед каждым вхождением символа '(одинарная кавычка) или " (двойная кавычка) вставлять символ \ (обратный слэш). Важно обеспечить сдвиг остальной части строки.

**2. Функция ltrim**

cpp

*// ANSI-версия*

void ltrim\_a(char\* str) {

*// ... код функции ...*

}

*// Unicode-версия*

void ltrim\_w(wchar\_t\* str) {

*// ... код функции ...*

}

**Алгоритм:** Функция должна найти первый непробельный символ в строке и сдвинуть всю оставшуюся часть строки (вместе с нулевым терминатором \0) в начало переданного буфера.

**3. Функция comparestrings**

cpp

*// ANSI-версия*

int comparestrings\_a(const char\* str1, const char\* str2) {

*// ... код функции ...*

}

*// Unicode-версия*

int comparestrings\_w(const wchar\_t\* str1, const wchar\_t\* str2) {

*// ... код функции ...*

}

**Возвращаемые значения:**

* 1 - если длина str1 (количество символов) меньше длины str2.
* 2 - если длина str2 меньше длины str1.
* 3 - если длины строк равны.

**Требования к реализации**

1. Создать проект в Microsoft Visual Studio типа "Консольное приложение".
2. Написать реализации всех шести функций (три для ANSI, три для Unicode).
3. В функции main() продемонстрировать работу всех функций на примерах. Создать тестовые строки (как ANSI, так и Unicode), вызвать для них функции и вывести результаты до/после обработки.
4. Продемонстрировать преобразование строк между кодировками с помощью WideCharToMultiByte (например, преобразовать Unicode-результат в UTF-8 для вывода).
5. Реализовать простейшую обработку ошибок: если какая-либо из функций WinAPI (например, для преобразования кодировки) вернула ошибку, получить ее код с помощью GetLastError(), вывести его и расшифровать с помощью FormatMessage().

**Примеры кода для вдохновления**

**Скелет для демонстрации в main()**

cpp

#include <windows.h>

#include <iostream>

#include <string>

*// Прототипы ANSI-функций*

void addslashes\_a(char\* str);

void ltrim\_a(char\* str);

int comparestrings\_a(const char\* str1, const char\* str2);

*// Прототипы Unicode-функций*

void addslashes\_w(wchar\_t\* str);

void ltrim\_w(wchar\_t\* str);

int comparestrings\_w(const wchar\_t\* str1, const wchar\_t\* str2);

int main() {

*// ===== ДЕМОНСТРАЦИЯ ANSI-ФУНКЦИЙ =====*

std::cout << "=== ANSI Functions Demo ===" << std::endl;

char ansiStr[256] = " Hello, 'World'!";

std::cout << "Original ANSI string: \"" << ansiStr << "\"" << std::endl;

ltrim\_a(ansiStr);

std::cout << "After ltrim\_a: \"" << ansiStr << "\"" << std::endl;

addslashes\_a(ansiStr);

std::cout << "After addslashes\_a: \"" << ansiStr << "\"" << std::endl;

*// ===== ДЕМОНСТРАЦИЯ UNICODE-ФУНКЦИЙ =====*

std::wcout << L"\n=== Unicode Functions Demo ===" << std::endl;

wchar\_t unicodeStr[256] = L" Hello, 'World'!";

std::wcout << L"Original Unicode string: \"" << unicodeStr << L"\"" << std::endl;

ltrim\_w(unicodeStr);

std::wcout << L"After ltrim\_w: \"" << unicodeStr << L"\"" << std::endl;

addslashes\_w(unicodeStr);

std::wcout << L"After addslashes\_w: \"" << unicodeStr << L"\"" << std::endl;

*// ===== ДЕМОНСТРАЦИЯ ПРЕОБРАЗОВАНИЯ И ОБРАБОТКИ ОШИБОК =====*

*// ... код с использованием WideCharToMultiByte и GetLastError ...*

return 0;

}

**Шаблон обработки ошибок**

cpp

*// Пример использования FormatMessage*

DWORD errorCode = GetLastError();

if (errorCode != 0) {

LPSTR errorMessage = nullptr;

FormatMessageA(

FORMAT\_MESSAGE\_ALLOCATE\_BUFFER | FORMAT\_MESSAGE\_FROM\_SYSTEM,

NULL,

errorCode,

0, *// Автоматический выбор языка*

(LPSTR)&errorMessage, *// Буфер будет выделен системой*

0,

NULL

);

std::cerr << "Error (Code " << errorCode << "): " << errorMessage << std::endl;

LocalFree(errorMessage); *// Очистка выделенного системой буфера*

}

**Контрольные вопросы**

1. В чем принципиальное отличие кодировок ANSI и Unicode?
2. Для чего нужны макросы TCHAR, LPCTSTR, TEXT()?
3. Какая функция WinAPI используется для преобразования Unicode-строки в ANSI-строку? Опишите ее основные параметры.
4. Как в программе на Win32 получить текстовое описание последней произошедшей ошибки?
5. Почему для современных приложений рекомендуется использовать Unicode?

**Варианты заданий к лабораторной работе:**

**Вариант 1**

Функцию написать в двух вариантах: для ANSI-строк и строк UNICODE (UTF-8).

1. Написать функцию addslashes (char \* str), которая вставляет символ обратного слэша (\) перед одинарными и двойными кавычками, встречающимися в строке.
2. Написать функцию ltrim (char \* str), которая удаляет пробелы в начале строки.
3. Написать функцию comparestrings (char \* str1,char \* str2), которая сравнивает количество символов в строках. Возвращаемые значения: 1 - если str1 короче, чем str2; 2 - если str2 короче, чем str1; 3 - если количество символов в строках одинаково.

**Вариант 2: Базовые строковые операции**

Написать функции для работы со строками в двух вариантах: для ANSI-строк (char\*) и для строк Unicode (wchar\_t\*).

1. **Функция mystrlwr**: преобразует все символы строки к нижнему регистру.  
   void mystrlwr(char\* str); / void mystrlwr(wchar\_t\* str);
2. **Функция mystrrchr**: находит последнее вхождение символа в строке и возвращает указатель на него. Если символ не найден, возвращает NULL.  
   char\* mystrrchr(const char\* str, int ch); / wchar\_t\* mystrrchr(const wchar\_t\* str, wchar\_t ch);
3. **Функция isstringsame**: сравнивает две строки без учета регистра. Возвращает TRUE, если строки идентичны (игнорируя регистр), и FALSE в противном случае.  
   BOOL isstringsame(const char\* str1, const char\* str2); / BOOL isstringsame(const wchar\_t\* str1, const wchar\_t\* str2);

**Вариант 3: Обработка и анализ строк**

Написать функции для работы со строками в двух вариантах: для ANSI-строк (char\*) и для строк Unicode (wchar\_t\*).

1. **Функция countwords**: подсчитывает количество слов в строке. Слово — это последовательность символов, отделенная пробелами, табуляциями или знаками препинания (например, запятыми, точками).  
   int countwords(const char\* str); / int countwords(const wchar\_t\* str);
2. **Функция reversestring**: переворачивает строку (меняет порядок символов на противоположный).  
   void reversestring(char\* str); / void reversestring(wchar\_t\* str);
3. **Функция isstringpalindrome**: проверяет, является ли строка палиндромом (читается одинаково слева направо и справа налево, игнорируя регистр и пробелы). Возвращает TRUEили FALSE.  
   BOOL isstringpalindrome(const char\* str); / BOOL isstringpalindrome(const wchar\_t\* str);

**Вариант 4: Строки и системные вызовы**

Написать функции для работы со строками в двух вариантах: для ANSI-строк (char\*) и для строк Unicode (wchar\_t\*). **Особое внимание уделить обработке ошибок WinAPI.**

1. **Функция getdrivesinfo**: формирует строку, содержащую список всех доступных в системе дисков (например, "C:\ D:\ E:"). Использовать функцию WinAPI GetLogicalDriveStrings.  
   BOOL getdrivesinfo(char\* outputBuffer, DWORD bufferSize); / BOOL getdrivesinfo(wchar\_t\* outputBuffer, DWORD bufferSize);
2. **Функция getusername**: получает имя текущего пользователя системы и записывает его в переданный буфер. Использовать функцию WinAPI GetUserName.  
   BOOL getusername(char\* outputBuffer, DWORD bufferSize); / BOOL getusername(wchar\_t\* outputBuffer, DWORD bufferSize);
3. **Функция getsystemdirectory**: получает путь к системной директории Windows (например, C:\Windows\System32) и записывает его в переданный буфер. Использовать функцию WinAPI GetSystemDirectory.  
   BOOL getsystemdirectory(char\* outputBuffer, DWORD bufferSize); / BOOL getsystemdirectory(wchar\_t\* outputBuffer, DWORD bufferSize);

**Задание по обработке ошибок:** Для каждой функции, использующей WinAPI, необходимо проверять код возврата. В случае ошибки:

* Вызывать GetLastError.
* С помощью FormatMessage получить текстовое описание ошибки.
* Выводить это описание на экран или записывать в буфер для ошибок.

**Вариант 5: "Умные" строковые операции (повышенная сложность)**

Написать функции для работы со строками в двух вариантах: для ANSI-строк (char\*) и для строк Unicode (wchar\_t\*).

1. **Функция strreplace**: ищет в строке все вхождения подстроки oldSubstr и заменяет их на подстроку newSubstr. Функция должна корректно обрабатывать изменение длины строки.  
   BOOL strreplace(char\* str, const char\* oldSubstr, const char\* newSubstr, DWORD bufferSize);/ BOOL strreplace(wchar\_t\* str, const wchar\_t\* oldSubstr, const wchar\_t\* newSubstr, DWORD bufferSize);
2. **Функция parsecsventry**: обрабатывает строку в формате CSV (Comma-Separated Values), извлекает из нее n-е поле (по индексу, начиная с 0). Поля могут быть заключены в кавычки.  
   BOOL parsecsventry(const char\* csvLine, int fieldIndex, char\* outputBuffer, DWORD bufferSize);/ BOOL parsecsventry(const wchar\_t\* csvLine, int fieldIndex, wchar\_t\* outputBuffer, DWORD bufferSize);
3. **Функция getstringencoding**: анализирует строку и пытается определить ее кодировку. Возвращает:
   * 1 - если строка, скорее всего, в ANSI.
   * 2 - если строка, скорее всего, в UTF-16LE (Unicode).
   * 3 - если строка, скорее всего, в UTF-8 (без BOM).
   * 0 - если не удалось определить.  
     *Алгоритм может быть простым: проверка на BOM, анализ patterns байт.*  
     int getstringencoding(const char\* str, int length); / int getstringencoding(const wchar\_t\* str, int length);

**Структура отчета для любого варианта:**

1. **Титульный лист.**
2. **Цель работы.**
3. **Теоретическое обоснование** (кратко описать различия ANSI/Unicode, используемые функции WinAPI, методы обработки ошибок).
4. **Блок-схемы алгоритмов** (опционально).
5. **Листинг кода** с комментариями.
6. **Примеры работы программы** (скриншоты консоли с демонстрацией всех функций на различных тестовых данных).
7. **Демонстрация обработки ошибок** (скриншоты, показывающие вывод сообщений об ошибках с помощью FormatMessage).
8. **Ответы на контрольные вопросы** (общие для всех вариантов, см. первый вариант лабораторной работы).
9. **Выводы по работе.**