Лабораторная работа 2. Работа с временем и системной информацией

**Вариант2: 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);

Выполнение:

Код:

#include <iostream>

#include <cctype>

#include <cwctype>

#include <cstring>

#include <cwchar>

#include <windows.h>

int countwords(const char\* str) {

if (str == nullptr || \*str == '\0') {

return 0;

}

int count = 0;

bool inWord = false;

while (\*str) {

// Проверяем, является ли символ буквой или цифрой

if (std::isalpha(static\_cast<unsigned char>(\*str)) ||

std::isdigit(static\_cast<unsigned char>(\*str))) {

if (!inWord) {

count++;

inWord = true;

}

}

else {

// Любой не-буквенный символ считается разделителем

inWord = false;

}

str++;

}

return count;

}

void reversestring(char\* str) {

if (str == nullptr || \*str == '\0') {

return;

}

char\* start = str;

char\* end = str + std::strlen(str) - 1;

while (start < end) {

char temp = \*start;

\*start = \*end;

\*end = temp;

start++;

end--;

}

}

BOOL isstringpalindrome(const char\* str) {

if (str == nullptr || \*str == '\0') {

return TRUE;

}

const char\* left = str;

const char\* right = str + std::strlen(str) - 1;

while (left < right) {

// Пропускаем не-буквенные символы слева

while (left < right && !std::isalpha(static\_cast<unsigned char>(\*left))) {

left++;

}

// Пропускаем не-буквенные символы справа

while (left < right && !std::isalpha(static\_cast<unsigned char>(\*right))) {

right--;

}

if (left >= right) {

break;

}

// Сравниваем символы без учета регистра

if (std::tolower(static\_cast<unsigned char>(\*left)) !=

std::tolower(static\_cast<unsigned char>(\*right))) {

return FALSE;

}

left++;

right--;

}

return TRUE;

}

int countwords(const wchar\_t\* str) {

if (str == nullptr || \*str == L'\0') {

return 0;

}

int count = 0;

bool inWord = false;

while (\*str) {

if (std::iswalpha(\*str) || std::iswdigit(\*str)) {

if (!inWord) {

count++;

inWord = true;

}

}

else {

inWord = false;

}

str++;

}

return count;

}

void reversestring(wchar\_t\* str) {

if (str == nullptr || \*str == L'\0') {

return;

}

wchar\_t\* start = str;

wchar\_t\* end = str + std::wcslen(str) - 1;

while (start < end) {

wchar\_t temp = \*start;

\*start = \*end;

\*end = temp;

start++;

end--;

}

}

BOOL isstringpalindrome(const wchar\_t\* str) {

if (str == nullptr || \*str == L'\0') {

return TRUE;

}

const wchar\_t\* left = str;

const wchar\_t\* right = str + std::wcslen(str) - 1;

while (left < right) {

while (left < right && !std::iswalpha(\*left)) {

left++;

}

while (left < right && !std::iswalpha(\*right)) {

right--;

}

if (left >= right) {

break;

}

if (std::towlower(\*left) != std::towlower(\*right)) {

return FALSE;

}

left++;

right--;

}

return TRUE;

}

int main() {

// Тестирование ANSI-функций

std::cout << "ANSI functions test:" << std::endl;

const char\* ansiText = "Hello, World! This is a test string.";

char ansiBuffer[100];

// Безопасное копирование строки

strcpy\_s(ansiBuffer, sizeof(ansiBuffer), ansiText);

std::cout << "Original: " << ansiBuffer << std::endl;

std::cout << "Word count: " << countwords(ansiBuffer) << std::endl;

reversestring(ansiBuffer);

std::cout << "Reversed: " << ansiBuffer << std::endl;

const char\* palindrome = "A man, a plan, a canal, Panama!";

std::cout << "Is palindrome '" << palindrome << "': "

<< (isstringpalindrome(palindrome) ? "YES" : "NO") << std::endl;

std::cout << std::endl;

// Тестирование Unicode-функций

std::wcout << L"Unicode functions test:" << std::endl;

const wchar\_t\* unicodeText = L"Привет, Мир! Это тестовая строка.";

wchar\_t unicodeBuffer[100];

// Безопасное копирование широкой строки

wcscpy\_s(unicodeBuffer, sizeof(unicodeBuffer) / sizeof(wchar\_t), unicodeText);

std::wcout << L"Original: " << unicodeBuffer << std::endl;

std::wcout << L"Word count: " << countwords(unicodeBuffer) << std::endl;

reversestring(unicodeBuffer);

std::wcout << L"Reversed: " << unicodeBuffer << std::endl;

const wchar\_t\* unicodePalindrome = L"А роза упала на лапу Азора";

std::wcout << L"Is palindrome '" << unicodePalindrome << L"': "

<< (isstringpalindrome(unicodePalindrome) ? L"YES" : L"NO") << std::endl;

return 0;

}

**Контрольные вопросы**

1. В чем принципиальное отличие кодировок ANSI и Unicode?

**ANSI** (обычно подразумевает Windows-1252 или другие однобайтовые кодировки):

* Использует 1 байт на символ (256 возможных символов)
* Поддерживает только ограниченный набор символов (в основном латиницу)
* Несовместимость между разными языковыми версиями

**Unicode** (UTF-16 в Windows):

* Использует 2 байта на символ (65,536 возможных символов)
* Поддерживает все языки мира в единой кодировке
* Обеспечивает кросс-языковую совместимость

1. Для чего нужны макросы TCHAR, LPCTSTR, TEXT()?

Эти макросы обеспечивают кроссплатформенность между ANSI и Unicode:

* **TCHAR** - автоматически становится char или wchar\_t в зависимости от настройки проекта
* **LPCTSTR** - указатель на константную строку TCHAR (Long Pointer to Constant TCHAR String)
* **TEXT()** - макрос для создания строковых литералов правильного типа

1. Какая функция WinAPI используется для преобразования Unicode-строки в ANSI-строку? Опишите ее основные параметры.

**WideCharToMultiByte** - преобразует Unicode строку в многобайтовую (ANSI).

Основные параметры:

* CP\_ACP - кодовая страница (текущая ANSI)
* dwFlags - флаги преобразования (обычно 0)
* lpWideCharStr - исходная Unicode строка
* cchWideChar - длина исходной строки
* lpMultiByteStr - буфер для результата
* cbMultiByte - размер буфера
* lpDefaultChar, lpUsedDefaultChar - обработка неизвестных символов

1. Как в программе на Win32 получить текстовое описание последней произошедшей ошибки?

Используется функция **FormatMessage()**:

Пример:

DWORD error = GetLastError();

LPVOID errorMsg;

FormatMessage(

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

NULL,

error,

MAKELANGID(LANG\_NEUTRAL, SUBLANG\_DEFAULT),

(LPTSTR)&errorMsg,

0,

NULL

);

LocalFree(errorMsg);

1. Почему для современных приложений рекомендуется использовать Unicode?
2. **Глобализация** - поддержка всех языков мира
3. **Упрощение разработки** - единая кодировка для всех языков
4. **Совместимость** с современными API (многие Windows API существуют только в Unicode-версии)
5. **Безопасность** - избежание проблем с преобразованием кодировок
6. **Производительность** - Windows внутренне использует Unicode
7. **Будущая-proof** - соответствие современным стандартам