Лабораторная работа 1. Работа с временем и системной информацией

Выполнено:

**Задание:**

Написать программу, выводящую на экран:

- Время работы ОС в формате "дни, часы, минуты, секунды"

- MessageBox с текущим временем и датой

- Имя компьютера и текущего пользователя

Код программы:

#include <lmcons.h>

#include <windows.h> // Подключаем Windows API

#include <iostream> // Для работы с консольным вводом-выводом

#include <string> // Для работы со строками

// Функция для форматирования времени работы системы из миллисекунд в строку

std::wstring FormatUptime(ULONGLONG milliseconds) {

// Константы для пересчета миллисекунд

const ULONGLONG ms\_per\_second = 1000;

const ULONGLONG ms\_per\_minute = ms\_per\_second \* 60;

const ULONGLONG ms\_per\_hour = ms\_per\_minute \* 60;

const ULONGLONG ms\_per\_day = ms\_per\_hour \* 24;

// Вычисляем количество дней, часов, минут, секунд

ULONGLONG days = milliseconds / ms\_per\_day;

milliseconds %= ms\_per\_day;

ULONGLONG hours = milliseconds / ms\_per\_hour;

milliseconds %= ms\_per\_hour;

ULONGLONG minutes = milliseconds / ms\_per\_minute;

milliseconds %= ms\_per\_minute;

ULONGLONG seconds = milliseconds / ms\_per\_second;

// Формируем и возвращаем отформатированную строку

return std::to\_wstring(days) + L" дн, " +

std::to\_wstring(hours) + L" ч, " +

std::to\_wstring(minutes) + L" мин, " +

std::to\_wstring(seconds) + L" сек";

}

int main() {

// 1. Вывод времени работы ОС в консоль

ULONGLONG systemUptimeMs = GetTickCount64();

std::wcout << L"Время работы системы: " << FormatUptime(systemUptimeMs) << std::endl;

// 2. Получение текущего локального времени и даты

SYSTEMTIME systemTime;

GetLocalTime(&systemTime); // Заполняем структуру SYSTEMTIME текущим временем

// Форматируем дату и время в строку для MessageBox

wchar\_t dateTimeStr[100];

swprintf\_s(dateTimeStr,

L"Дата: %02d.%02d.%d\nВремя: %02d:%02d:%02d",

systemTime.wDay, systemTime.wMonth, systemTime.wYear,

systemTime.wHour, systemTime.wMinute, systemTime.wSecond

);

// 3. Получение имени компьютера и текущего пользователя

wchar\_t computerName[MAX\_COMPUTERNAME\_LENGTH + 1];

DWORD size = sizeof(computerName) / sizeof(computerName[0]);

GetComputerName(computerName, &size);

wchar\_t userName[UNLEN + 1];

DWORD userNameSize = sizeof(userName) / sizeof(userName[0]);

GetUserName(userName, &userNameSize);

// Форматируем строку с системной информацией

wchar\_t infoStr[200];

swprintf\_s(infoStr,

L"%s\n\nИмя компьютера: %s\nТекущий пользователь: %s",

dateTimeStr, computerName, userName

);

// 4. Вывод всей информации в MessageBox

MessageBoxW(

NULL,

infoStr,

L"Системная информация",

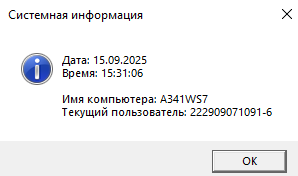
MB\_OK | MB\_ICONINFORMATION

);

return 0;

}

Вывод:



**Контрольные вопросы:**

1. Какие программы относятся к системному ПО?

Системное программное обеспечение (ПО) — это программы, предназначенные для обеспечения работы самого компьютера и выполнения прикладных задач. К нему относятся:

* Операционные системы (Windows, Linux, macOS): Управляют аппаратными ресурсами и предоставляют среду для выполнения прикладных программ.
* Драйверы устройств: Обеспечивают взаимодействие ОС и конкретного hardware (видеокарты, принтера, сканера).
* Утилиты: Программы для обслуживания и настройки системы (архиваторы, антивирусы, дефрагментаторы диска, утилиты очистки).
* Системные библиотеки: Коллекции готовых функций и процедур для разработчиков (например, DLL в Windows, Windows.h предоставляет к ним доступ).
* Средства разработки: Компиляторы, интерпретаторы, отладчики, которые трансформируют код программы в машинные инструкции.

1. Что такое операционная система?

Операционная система (ОС) — это самый главный, фундаментальный комплекс программ, который управляет всеми ресурсами компьютера (процессором, памятью, дисками, устройствами ввода-вывода) и предоставляет интерфейс для взаимодействия пользователя и прикладных программ с аппаратурой. Основные функции ОС:

* Управление процессами: Запуск, приостановка, завершение программ, распределение времени процессора между ними.
* Управление памятью: Выделение и освобождение оперативной памяти для процессов.
* Управление файловой системой: Работа с данными на дисках (создание, чтение, запись, удаление файлов и каталогов).
* Управление устройствами: Взаимодействие с периферийными устройствами через драйверы.
* Предоставление пользовательского интерфейса: Командная строка или графический интерфейс (GUI).
* Обеспечение безопасности: Разграничение прав доступа пользователей к ресурсам.

1. Что такое системный вызов и зачем он нужен?

Системный вызов — это запрос от программы, работающей в пользовательском режиме, к ядру операционной системы, работающему в привилегированном режиме, с требованием выполнить какую-либо операцию, требующую высокого уровня доступа к аппаратуре или системным ресурсам.

Зачем он нужен: для обеспечения безопасности и стабильности системы. Прямой доступ прикладных программ запрещен. Вместо этого программа делает запрос ядру ОС через системный вызов. Ядро, будучи доверенным компонентом, проверяет права программы и корректность запроса, и только затем выполняет нужную операцию (например, чтение из файла на диске или выделение памяти). Функции WinAPI (как GetTickCount64), которые мы используем в программе, являются обертками над более низкоуровневыми системными вызовами.

1. В чем отличие 32 и 64 разрядных систем?

Разрядность системы определяет, какой объем данных процессор может обработать за один такт и, как следствие, какой объем памяти он может адресовать.

|  |  |  |
| --- | --- | --- |
| Характеристика | 32-разрядная (x86) | 64-разрядная (x64) |
| Адресуемая оперативная память (ОЗУ) | Теоретически до 4 ГБ, на практике ~3.25-3.5 ГБ | Теоретически до 16 ЭБ (эксабайт), на практике огромные объемы (Терабайты) |
| Обработка данных | Работает с 32-битными порциями данных за такт | Работает с 64-битными порциями данных за такт, что часто повышает производительность |
| Совместимость | Может запускать только 32-битные приложения | Может запускать как 64-битные, так и большинство 32-битных приложений (в режиме совместимости) |
| Регистры процессора | Использует 8 основных регистров общего назначения | Использует 16 регистров общего назначения, что позволяет эффективнее работать с данными |

5.Библиотека Windows.h, ее функции и особенности

Windows.h — это главный заголовочный файл для языка C/C++ в среде Windows, который предоставляет доступ к функциям Windows API (WinAPI).

Функции: WinAPI позволяет программам напрямую взаимодействовать с операционной системой Windows для выполнения задач, которые стандартные библиотеки C/C++ сделать не могут:

* Создание и управление окнами, кнопками, меню (GUI).
* Работа с файлами, реестром, процессами и потоками.
* Управление памятью, устройствами, сетью.
* Получение системной информации (как в нашей лабораторной работе).

Особенности:

* Низкоуровневый доступ: даёт почти полный контроль над системой, но требует больше кода и осторожности по сравнению с высокоуровневыми фреймворками (например, .NET или Qt).
* Совместимость: позволяет создавать программы, совместимые с разными версиями Windows.
* Порядок включения: как верно замечено в теоретических сведениях, #include <windows.h> рекомендуется размещать до стандартных заголовочных файлов (например, <iostream>). Это связано с тем, что windows.h определяет макросы (например, min и max), которые могут конфликтовать с именами из стандартной библиотеки, если подключить их в неправильном порядке.