Лабораторная работа 1. Работа с временем и системной информацией

Теоретические сведения:

**Библиотека Windows.h**

Windows.h — заголовочный файл для работы с операционной системой Windows на языке программирования C/С++. Он предоставляет доступ к Windows API (WinAPI) — набору функций, макросов, типов данных и структур, которые позволяют взаимодействовать с системой на низком уровне.

Области применения:

работа с файлами и файловой системой (например, CreateFile, ReadFile, WriteFile);

создание окон и управление элементами графического интерфейса (например, CreateWindow, ShowWindow);

управление процессами и потоками (например, CreateProcess, TerminateProcess, CreateThread);

работа с сетями и коммуникациями (например, Winsock);

управление памятью и использование системных ресурсов (например, GlobalAlloc, HeapAlloc).

! windows.h должен быть включён перед любыми другими заголовочными файлами, связанными с Windows API, даже стандартными библиотеками. Это связано с тем, что windows.h включает макросы и другие компоненты, которые могут изменять, расширять или заменять функции в этих библиотеках.

Некоторые функции библиотеки windows.h

1. GetTickCount64 — функция из библиотеки Windows.h на C++. Она возвращает количество миллисекунд, прошедших с момента запуска системы.

Синтаксис:

ULONGLONG GetTickCount64();.

Возвращает: количество миллисекунд.

Пример работы:

#include <windows.h>

#include <iostream>

int main() {

ULONGLONG ticks = GetTickCount64(); // Возвращает количество миллисекунд

std::cout << "Система работает уже: " << ticks << " миллисекунд." << std::endl;

Чтобы получить время в удобном формате, необходимо перевести миллисекунды в часы, минуты и секунды.

2. MessageBox

Функция создает всплывающее окно с заданным сообщением, заголовком и кнопками

Пример:

#include <windows.h>

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow) {

MessageBox(NULL, L"Привет, мир!", L"Простой пример", MB\_OK | MB\_ICONINFORMATION);

return 0;

}

Где

* NULL: Указывает, что окно сообщения не принадлежит какому-то конкретному родительскому окну.
* L"Привет, мир!": Текст сообщения, которое будет отображаться в окне. Префикс L указывает, что это строка в формате Unicode.
* L"Простой пример": Заголовок окна сообщения.
* MB\_OK | MB\_ICONINFORMATION: Флаги, определяющие вид окна. MB\_OK добавляет кнопку "ОК", а MB\_ICONINFORMATION добавляет значок информации.

Флаги MessageBox

Для установки свойств окна необходимо использовать следующие флаги:

Определите один из следующих флажков, чтобы указать кнопки, содержащиеся в окне сообщений:

* **MB\_OK** - Окно сообщение содержит одну командную кнопку: OK. Это по умолчанию.
* **MB\_OKCANCEL** - Окно сообщение содержит две командных кнопки: OK и Отменить (Cancel).
* **MB\_RETRYCANCEL** - Окно сообщение содержит две командных кнопки: Повторить (Retry) и Отменить (Cancel).
* **MB\_YESNO** - Окно сообщение содержит две командных кнопки: Да (Yes) и Нет (No).
* **MB\_YESNOCANCEL** - Окно сообщение содержит три командных кнопки: Да (Yes), Нет (No) и Отменить (Cancel).

Определите один из следующих флажков, чтобы отобразить пиктограмму в окне сообщений:

* **MB\_ICONEXCLAMATION, MB\_ICONWARNING** - В окне сообщений появляется пиктограмма восклицательного знака.
* **MB\_ICONINFORMATION, MB\_ICONASTERISK** - В окне сообщений появляется пиктограмма, состоящая из символа i нижнего регистра в круге.
* **MB\_ICONQUESTION** - В окне сообщений появляется пиктограмма в виде знака вопроса.
* **MB\_ICONSTOP,MB\_ICONERROR,MB\_ICONHAND** - В окне сообщений появляется пиктограмма в виде стоп-сигнала.

Определите один из следующих флажков, чтобы указать заданную по умолчанию кнопку:

* **MB\_DEFBUTTON1** - Первая кнопка - основная кнопка. MB\_DEFBUTTON1 - значение по умолчанию, если не определена кнопка MB\_DEFBUTTON2, MB\_DEFBUTTON3 или MB\_DEFBUTTON4.
* **MB\_DEFBUTTON2** - Вторая кнопка - основная кнопка.
* **MB\_DEFBUTTON3** - Третья кнопка - основная кнопка.
* **MB\_DEFBUTTON4** - Четвертая кнопка - основная кнопка.

Определите один из следующих флажков, чтобы указать модальность диалогового окна:

* **MB\_APPLMODAL** - Пользователь должен ответить окну сообщений перед продолжением работы в окне, которое идентифицировано параметром hWnd. Однако, пользователь может перемещаться в окнах других прикладных программ и работать в этих окнах.В зависимости от иерархии окон в прикладной программе, пользователь может получить возможность, чтобы перемещаться в другие окна в пределах прикладной программы. Все дочерние окна родителя окна сообщений автоматически блокируются, однако выскакивающие окна - нет.MB\_APPLMODAL - значение по умолчанию, если не определен флажок, ни MB\_SYSTEMMODAL, ни MB\_TASKMODAL.
* **MB\_SYSTEMMODAL** - То же самое, что и MB\_APPLMODAL за исключением того, что окно сообщений имеет стиль WS\_EX\_TOPMOST. Используйте системно - модальные окна сообщений, чтобы уведомлять пользователя о серьезных, потенциально опасных ошибках, которые требуют немедленного внимания (например, запуск программы при нехватке памяти). Этот флажок не имеет никакого влияния на способность пользователя взаимодействовать с другими окнами, а не те, которые связаны с hWnd.
* **MB\_TASKMODAL** - То же самое, что и MB\_APPLMODAL за исключением того, что все окна верхнего уровня, принадлежащие текущей задаче, заблокированы, если параметр hWnd имеет значение ПУСТО (NULL). Используйте этот флажок, когда вызывающая прикладная программа или библиотека не имеют доступного дескриптора окна, но все еще должны сохранять вводимые данные для других окон в текущей прикладной программе без приостановки работы других прикладных программ.

**Контрольные вопросы:**

1. Какие программы относятся к системному ПО?

2. Что такое операционная система?

3. Что такое системный вызов и зачем он нужен?

4. В чем отличие 32 и 64 разрядных систем?

5.Библиотека Windows.h, ее функции и особенности

**Задание к лабораторной работе:**

**Вариант 1**

Написать программу, выводящую на экран:

1. Время работы ОС в формате "дни, часы, минуты, секунды"
2. MessageBox с текущим временем и датой
3. Имя компьютера и текущего пользователя

**Вариант 2: Системный мониторинг**

Написать программу, которая выводит в консоль и (опционально) в MessageBox следующую информацию:

1. **Время работы системы** в формате "дни:часы:минуты:секунды".
2. **Размер и состояние оперативной памяти:** общий объем физической памяти и объем доступной памяти в мегабайтах.
3. **Информацию о диске C:\:** общий объем и объем свободного места в гигабайтах.
4. **Версию операционной системы** (например, "Microsoft Windows 10 Pro").

**Теоретическое описание (дополнительно к исходному):**

* **Функция GlobalMemoryStatusEx** (из Windows.h) для получения информации об оперативной памяти.
* **Функция GetDiskFreeSpaceEx** (из Windows.h) для получения информации о свободном месте на диске.
* **Функция GetVersionEx** (из Windows.h) или новый способ с помощью RtlGetVersion для определения версии ОС. *(Примечание: GetVersionEx может возвращать некорректные данные в новых версиях Windows из-за манифеста приложения)*.

**Вариант 3: Интерактивный таймер**

Написать программу, которая:

1. При запуске записывает точное время своего запуска.
2. Выводит сообщение MessageBox с кнопками **"OK"** и **"Cancel"** и текстом "Программа запущена. Остановить?".
3. Если пользователь нажимает "OK", программа закрывается, выводя в консоль время своей работы (от запуска до закрытия).
4. Если пользователь нажимает "Cancel", программа показывает новое окно MessageBox с вопросом "Вы уверены?" (**MB\_YESNO**). При выборе "Нет" возвращается к предыдущему сообщению. При выборе "Да" — закрывается, также выводя время работы.

**Теоретическое описание (дополнительно к исходному):**

* Работа с циклами и обработкой возвращаемых значений из MessageBox (например, IDOK, IDCANCEL, IDYES, IDNO).
* Использование функции GetLocalTime или GetSystemTime (из Windows.h) для получения точного времени начала работы и вычисления интервала.

**Вариант 4: "Системный смотритель"**

Создать программу, которая каждые **N** секунд (где N вводит пользователь при запуске) проверяет состояние системы и выводит предупреждение, если:

1. **Слишком долго работает система** (например, больше 7 дней). Использовать GetTickCount64().
2. **Мало свободного места на системном диске** (менее 5 ГБ). Использовать GetDiskFreeSpaceEx.
3. Программа должна работать в фоновом режиме, выводя предупреждения с помощью MessageBox с соответствующими иконками (**MB\_ICONWARNING**, **MB\_ICONERROR**).

**Теоретическое описание (дополнительно к исходному):**

* Функция Sleep (из Windows.h) для приостановки выполнения программы на заданный интервал времени (в миллисекундах).
* Организация бесконечного цикла для мониторинга.

**Вариант 5: Кроссплатформенный вывод (сложный)**

Написать программу, которая определяет, под управлением какой ОС она запущена (Windows или Linux, используя макросы препроцессора), и выводит системную информацию соответствующим образом.

* **Для Windows:** Использовать Windows.h (GetComputerName, GetUserName, GetTickCount64).
* **Для Linux:** Использовать стандартные функции C++ (например, std::chrono для времени работы) и системные вызовы (например, чтение файлов /proc/uptime, /etc/hostname, getenv("USER")).

**Задание:**

1. Вывести имя компьютера.
2. Вывести имя текущего пользователя.
3. Вывести время работы системы.

**Теоретическое описание (дополнительно к исходному):**

* Использование директив препроцессора #ifdef, #else, #endif для условной компиляции.
* Основы работы с файловыми потоками в C++ (std::ifstream) для чтения системной информации в Linux.

**Вариант 6: GUI-версия (использование WinAPI окон)**

Создать не консольное приложение, а диалоговое окно с использованием WinAPI.

1. При запуске программы создается окно с несколькими полями (Static Text) и кнопкой "Обновить".
2. В полях отображается:
   * Текущее дата и время (обновляется по нажатию кнопки).
   * Имя компьютера.
   * Имя пользователя.
   * Время работы системы.
3. Добавить кнопку "О программе", которая вызывает MessageBox с информацией о разработчике.

**Теоретическое описание (дополнительно к исходному):**

* Основы создания диалоговых окон через функцию DialogBox.
* Использование процедуры окна (DLGPROC) для обработки сообщений (например, нажатия кнопок).
* Функции для работы с элементами управления: SetDlgItemText, GetDlgItemText.

**Требование к отчету:**

-Код программы

-Скриншоты работы

-Теоретическое описание функций для вывода текущего времени и даты, которые использованы в программе

- Ответы на контрольные вопросы