**Пояснения к лабораторной 1**

Общее для всех вариантов:

Язык: C++.

Среда: Microsoft Visual Studio.

Ключевая библиотека: Windows.h (для взаимодействия с WinAPI). Все предложенные функции (GetTickCount64, MessageBox, GlobalMemoryStatusEx, GetDiskFreeSpaceEx и т.д.) являются частью WinAPI и прекрасно доступны в Visual Studio.

Пояснения по конкретным вариантам:\*

**Вариант 1 (Исходный) и Вариант 2 (Системный мониторинг):**

Идеально для *консольного приложения*.

Функции GlobalMemoryStatusEx, GetDiskFreeSpaceEx и попытка получить версию ОС (GetVersionEx или RtlGetVersion) — все это часть WinAPI и полностью поддерживается.

**Вариант 3 (Интерактивный таймер):**

Отлично подходит для *консольного приложения*.

Обработка возвращаемых значений MessageBox (например, if (MessageBox(...) == IDOK)) — это стандартная практика.

**Вариант 4 ("Системный смотритель"):**

Тип проекта: *Консольное приложение*. Чтобы оно работало именно в фоновом режиме без постоянного висящего окна консоли, вы можете изменить настройки линковки проекта на "Подсистема: Windows (/SUBSYSTEM:WINDOWS)". Это скроет окно консоли, и программа будет работать как фоновый процесс, а свои предупреждения выводить через MessageBox.

Функция Sleep() из Windows.h работает идеально.

**Вариант 5 (Кроссплатформенный вывод):**

Это самый сложный, но абсолютно реализуемый вариант.

В Visual Studio вы будете использовать *Директивы препроцессора* для условной компиляции.

cpp

#ifdef \_WIN32 // Если компилируем под Windows

#include <windows.h>

// Используем WinAPI функции

#elif \_\_linux\_\_ // Если компилируем под Linux

#include <fstream>

#include <chrono>

#include <unistd.h>

#include <cstdlib>

// Используем стандартные и Linux-специфичные методы

#endif

Чтобы протестировать Linux-часть кода, вам понадобится либо:

1. Windows Subsystem for Linux (WSL):\* Установить Ubuntu (или другой дистрибутив) через WSL и компилировать код там с помощью компилятора g++.

2. Виртуальная машина с Linux.

3. Двойная загрузка Windows/Linux.

Замечание: Этот вариант требует deepestшего понимания C++ и различий между ОС.

**Вариант 6 (GUI-версия):**

Для этого варианта в Visual Studio нужно выбрать *другой тип проекта*.

Лучший выбор: "Проект *Windows Desktop*" -> "*Windows Desktop Application (Windows)*" или, что еще проще, "*Проект Windows Desktop*" -> "*Windows Desktop Wizard*", где можно выбрать тип "Empty project" и поставить галочку "Empty application".

Visual Studio имеет встроенные редакторы ресурсов для удобного создания диалоговых окон, добавления кнопок, полей текста и т.д. через интерфейс drag-and-drop. Код для элементов будет сгенерирован автоматически.

Этот вариант напрямую использует механизм сообщений Windows и процедуры окон, что является классическим способом создания GUI на C++ под Windows.