Коллоквиум 1

Михайлова Александра 11 группа

2 вариант

# 1. Приведите WinAPI, необходимое для решения лабораторной работы №2

В лабораторной №2 реализуется многопоточная программа с потоками main, min\_max и average.

Основные действия:

-создание потоков;

-ожидание завершения;

-паузы (Sleep);

-синхронный вывод результатов в консоль.

Используемое WinAPI:

1. Создание потока

HANDLE CreateThread

( LPSECURITY\_ATTRIBUTES lpThreadAttributes,

SIZE\_T dwStackSize,

LPTHREAD\_START\_ROUTINE lpStartAddress,

LPVOID lpParameter,

DWORD dwCreationFlags,

LPDWORD lpThreadId );

Используется для запуска min\_max и average потоков. lpParameter передаёт указатель на структуру с данными.

1. Ожидание завершения потока

WaitForSingleObject(HANDLE hThread, DWORD dwMilliseconds);

Применяется в main для ожидания завершения min\_max и average. Второй параметр — INFINITE.

1. Пауза

Sleep(DWORD dwMilliseconds);

Используется в min\_max (на 7 мс) и average (на 12 мс) после операций сравнения/сложения, имитируя задержку.

1. Завершение потока

ExitThread(DWORD dwExitCode);

1. Закрытие дескрипторов

CloseHandle(hThread);

Пример вызова:

HANDLE hThread = CreateThread(NULL, 0, ThreadFunc, &args, 0, NULL); WaitForSingleObject(hThread, INFINITE); CloseHandle(hThread);

# 2. Что такое процесс в ОС Windows

Процесс в Windows — изолированная среда выполнения программы, включающая:

-виртуальное адресное пространство;

-один или более потоков;

-таблицу дескрипторов (файлы, семафоры и т.п.);

-информацию о состоянии.

В контексте лабораторных: В ЛР2 — один процесс, в нём создаются три потока (main, min\_max, average). В ЛР4 — создаются отдельные процессы Sender и Receiver, которые обмениваются сообщениями через общий файл, используют события и работают независимо.

3. Что такое Критическая секция

Критическая секция (Critical Section) — механизм синхронизации между потоками одного процесса. Предназначен для исключительного доступа к общим данным.

Использование в лабораторной №3: Потоки marker получают общий доступ к массиву целых чисел. Чтобы избежать гонок при записи и чтении, вокруг обращения к массиву используются критические секции.

API:

CRITICAL\_SECTION cs;

InitializeCriticalSection(&cs);

EnterCriticalSection(&cs);

LeaveCriticalSection(&cs);

DeleteCriticalSection(&cs);

4. Что такое Семафор

Семафор — объект синхронизации с внутренним счётчиком, ограничивающим количество потоков/процессов, одновременно получающих доступ к ресурсу.

В отличие от критических секций:

-Работает как межпроцессный механизм;

-Может использоваться в разных процессах (Sender и Receiver);

-Позволяет задать количество допустимых одновременных доступов.

API:

HANDLE CreateSemaphore(...);

WaitForSingleObject(hSemaphore, INFINITE);

ReleaseSemaphore(hSemaphore, 1, NULL);

Использование: В ЛР4 (Sender ↔ Receiver) могла быть использована очередь сообщений и синхронизация через события или семафоры, хотя формально в задании указан общий файл, а не семафор — это альтернатива для организации более надёжной синхронизации.

# 5. Сравнительный анализ C++98 с/без Boost и новых стандартов с/без Qt

В лабораторных используется WinAPI и C++98 без Boost:

-Все потоки создаются через CreateThread;

-Синхронизация: WaitForSingleObject, Sleep, CRITICAL\_SECTION;

-Нет std::thread, std::mutex (не используется C++11);

-Ввод-вывод и логика — вручную, без готовых шаблонов/объектов.

|  |  |  |  |
| --- | --- | --- | --- |
| Параметр | C++98 + WinAPI (как в лабораторных) | C++11+ (std) | C++ + Qt |
| Потоки | CreateThread() | std::thread | QThread |
| Синхронизация | CRITICAL\_SECTION, WaitForSingleObject | std::mutex, std::condition\_variable | QMutex, QSemaphore |
| Удобство | Низкое | Среднее | Высокое |
| Безопасность | Нужно вручную управлять ресурсами | RAII через lock\_guard | Высокоуровневая обёртка |
| Переносимость | олько Windows | Кроссплатформенно | Кроссплатформенно |
| Зависимости | Нет | Стандарт | Qt-библиотека |

Пример C++11:

std::thread t1(calcMinMax, std::ref(array));

std::thread t2(calcAverage, std::ref(array));

t1.join();

t2.join();

Вместо WaitForSingleObject и CreateThread всё делается проще.