ЛР 5

Теор часть:

\*\*Реестр и журналы (Windows). Доступ к реестру Windows. Работа с журналами Windows. Другие вспомогательные средства управления.\*\*

\*\*Реестр Windows:\*\*

Windows Registry (Реестр Windows) - это центральное хранилище системных и прикладных настроек операционной системы Windows. Он представляет собой иерархическую базу данных, содержащую информацию о конфигурации системы, установленных приложениях и компонентах. Реестр используется для хранения параметров конфигурации, настроек безопасности, информации о установленных программах, драйверах устройств и многого другого.

\*\*Структура Реестра:\*\*

Реестр организован в виде древовидной структуры, в которой узлы называются "разделами" (keys), а данные хранятся в "значениях" (values). Основные разделы включают:

- `HKEY\_CLASSES\_ROOT`: Содержит информацию о типах файлов и связях с приложениями.

- `HKEY\_CURRENT\_USER`: Содержит профиль текущего пользователя.

- `HKEY\_LOCAL\_MACHINE`: Содержит конфигурацию компьютера.

- `HKEY\_USERS`: Содержит профили всех пользователей.

- `HKEY\_CURRENT\_CONFIG`: Содержит информацию о текущей конфигурации.

\*\*Работа с Реестром:\*\*

Для работы с Реестром Windows можно использовать различные средства, включая:

1. \*\*Редактор Реестра:\*\* Встроенный инструмент для просмотра и редактирования Реестра. Он позволяет добавлять, изменять и удалять ключи и значения в Реестре.

2. \*\*API для работы с Реестром:\*\* Windows предоставляет API функции для доступа к Реестру из приложений. Эти функции позволяют программам читать и записывать данные в Реестр.

Пример API функций:

- `RegOpenKeyEx()`: Открывает существующий ключ Реестра или создает новый.

- `RegQueryValueEx()`: Читает значение из ключа Реестра.

- `RegSetValueEx()`: Устанавливает значение в ключ Реестра.

\*\*Журналы Windows (Event Logs):\*\*

Windows поддерживает систему журналов событий, которая записывает информацию о различных событиях и состояниях операционной системы. Журналы могут содержать информацию о сбоях, предупреждениях, информационных сообщениях и многом другом.

\*\*Журналы включают:\*\*

1. \*\*Application Log (Журнал приложений):\*\* Содержит события, связанные с работой приложений и программ.

2. \*\*Security Log (Журнал безопасности):\*\* Содержит информацию о попытках входа, аудитории безопасности и других безопасностных событиях.

3. \*\*System Log (Журнал системы):\*\* Содержит информацию о работе операционной системы, такие как сбои и состояние системы.

\*\*Работа с журналами:\*\*

Системные администраторы и разработчики могут использовать утилиты и API Windows для анализа журналов событий. Это позволяет мониторить состояние системы, выявлять проблемы и отслеживать действия пользователей.

\*\*Другие вспомогательные средства управления:\*\*

Помимо Реестра и журналов, Windows предоставляет различные инструменты для управления системой, такие как:

- \*\*Диспетчер задач:\*\* Позволяет просматривать и управлять текущими процессами и службами.

- \*\*Утилиты администратора:\*\* Включают инструменты для настройки безопасности, управления учетными записями, резервного копирования и восстановления системы и др.

- \*\*Групповая политика (Group Policy):\*\* Позволяет администраторам определять правила и настройки для пользователей и компьютеров в домене.

- \*\*Службы (Services):\*\* Позволяют управлять службами Windows, такими как серверы и драйверы.

Все эти средства обеспечивают широкие возможности для управления и настройки операционной системы Windows с целью обеспечения безопасности, стабильности и эффективности работы.

Разбор кода:

Данный код представляет собой программу, которая мониторит изменения в разрешении экрана, языке ввода и настройках системы, и записывает информацию о таких изменениях в журнал событий Windows Event Log. Давайте рассмотрим, что и где используется в данном коде:

1. \*\*Заголовочные файлы и глобальные переменные:\*\*

- `#include <Windows.h>`: Подключение заголовочного файла Windows API для использования функций и типов данных Windows.

- `#include <tchar.h>`: Подключение заголовочного файла для поддержки символов широкого и узкого формата (TCHAR).

- `#include <evntrace.h>`: Подключение заголовочного файла для работы с журналом событий Windows Event Log.

- `LRESULT CALLBACK WndProc(HWND hwnd, UINT msg, WPARAM wParam, LPARAM lParam);`: Прототип функции WndProc, которая будет обрабатывать сообщения окна.

- `int mainScreenWidth = -1; int mainScreenHeight = -1; HKL mainInputLocale = NULL;`: Глобальные переменные для отслеживания разрешения экрана и языка ввода.

2. \*\*Функция `LogEventToEventLog()`:\*\*

- Эта функция создает запись в журнале Windows Event Log с указанным сообщением.

- `RegisterEventSource()`: Регистрирует источник события для журнала событий.

- `ReportEvent()`: Создает запись события в журнале.

- `DeregisterEventSource()`: Освобождает ресурсы после работы с журналом.

- `MessageBox()`: Отображает диалоговое окно с сообщением.

3. \*\*Функции `HandleResolutionChange()`, `HandleInputLocaleChange()`, `HandleSystemParamChange(UINT uiAction)`:\*\*

- Эти функции обрабатывают изменения разрешения экрана, языка ввода и другие системные параметры.

- Они вызывают функцию `LogEventToEventLog()`, чтобы создать запись в журнале при изменениях.

4. \*\*Функция `WinMain()`:\*\*

- Это точка входа в приложение Windows.

- Создается окно и обрабатываются сообщения Windows.

- В основном цикле (`while (GetMessage(&msg, NULL, 0, 0))`) проверяются сообщения и вызываются функции обработки изменений.

5. \*\*Функция `WndProc()`:\*\*

- Эта функция обрабатывает сообщения окна (события окна).

- В случае получения определенных сообщений (например, `WM\_DISPLAYCHANGE`, `WM\_INPUTLANGCHANGE`, `WM\_SETTINGCHANGE`), вызываются соответствующие функции обработки изменений.

6. \*\*Использование таймера:\*\*

- В функции `WndProc()` создается таймер с помощью `SetTimer()`, который каждую секунду (1000 миллисекунд) вызывает функции `HandleResolutionChange()` и `HandleInputLocaleChange()`.

- Таймер устанавливается в `WM\_CREATE` и уничтожается в `WM\_DESTROY`.

Таким образом, код мониторит изменения в разрешении экрана, языке ввода и настройках системы с использованием сообщений Windows и записывает информацию о таких изменениях в журнал событий Windows Event Log.

Возможные вопросы:

Вопросы, которые могут быть заданы по данному коду, могут включать темы относящиеся к использованию Windows API, работе с журналами событий, мониторингу изменений системных параметров и таймерам. Вот несколько примеров вопросов с ответами:

1. \*\*Что делает эта программа?\*\*

- Ответ: Программа мониторит изменения в разрешении экрана, языке ввода и других системных параметрах и записывает информацию о таких изменениях в журнал событий Windows Event Log.

2. \*\*Зачем используется функция `LogEventToEventLog()`?\*\*

- Ответ: Эта функция используется для создания записи в журнале Windows Event Log с указанным сообщением, чтобы отслеживать и регистрировать изменения и события.

3. \*\*Что означают параметры `eventCategory` и `eventId` в функции `ReportEvent()`?\*\*

- Ответ: `eventCategory` определяет категорию события, а `eventId` - идентификатор события в журнале.

4. \*\*Какие системные параметры мониторятся в этой программе?\*\*

- Ответ: Программа мониторит изменения разрешения экрана (WM\_DISPLAYCHANGE), языка ввода (WM\_INPUTLANGCHANGE) и другие системные параметры (WM\_SETTINGCHANGE).

5. \*\*Почему в коде использован таймер?\*\*

- Ответ: Таймер устанавливает периодическую проверку изменений разрешения экрана и языка ввода, обновляя информацию каждую секунду. Таким образом, программа может реагировать на изменения в реальном времени.

6. \*\*Что делает функция `HandleResolutionChange()`?\*\*

- Ответ: Функция `HandleResolutionChange()` сравнивает текущее разрешение экрана с предыдущим состоянием и, если разрешение изменилось, записывает сообщение в журнал событий.

7. \*\*Какие действия выполняются при запуске программы?\*\*

- Ответ: При запуске программы создается окно, устанавливается таймер для мониторинга изменений, и начинается обработка сообщений Windows.

8. \*\*Что происходит при уничтожении окна?\*\*

- Ответ: При уничтожении окна программа останавливает таймер и завершает выполнение.

9. \*\*Какие события и сообщения могут быть записаны в журнал событий Windows Event Log?\*\*

- Ответ: Программа записывает события об изменениях разрешения экрана, языке ввода и других системных параметрах, такие как изменение обоев рабочего стола.

10. \*\*Какова цель мониторинга и записи этих изменений?\*\*

- Ответ: Главная цель - предоставление информации и уведомлений об изменениях в системе, что может быть полезным для администраторов и пользователей для отслеживания состояния системы.

Эти вопросы и ответы позволяют понять основные аспекты работы программы и использование Windows API для мониторинга и регистрации событий в Windows Event Log.

Общие вопросы по теме ЛР5:

Когда рассматривается тема "Реестр и журналы (Windows). Доступ к реестру Windows. Работа с журналами Windows. Другие вспомогательные средства управления", могут быть заданы следующие вопросы, не прямо связанные с данным конкретным кодом, но относящиеся к общей теме:

1. \*\*Что представляет собой реестр Windows, и какие данные в нем хранятся?\*\*

- Ответ: Реестр Windows - это централизованное хранилище системной конфигурации и настроек. В нем хранятся данные о программных и аппаратных ресурсах, настройках операционной системы и установленных приложениях.

2. \*\*Какие виды операций можно выполнять с реестром Windows?\*\*

- Ответ: С реестром Windows можно выполнять операции чтения, записи, обновления и удаления данных. Операции выполняются с использованием специальных API, предоставляемых операционной системой.

3. \*\*Что такое журналы Windows Event Log?\*\*

- Ответ: Журналы Windows Event Log - это система регистрации событий, позволяющая записывать информацию о событиях, происходящих в операционной системе, приложениях и службах. Эти журналы могут быть использованы для мониторинга и анализа произошедших событий.

4. \*\*Какие типы событий могут быть записаны в журналы Windows Event Log?\*\*

- Ответ: В журналах Windows Event Log можно записывать различные типы событий, такие как информационные, предупреждения, ошибки и аудиторские события. Эти события описывают действия и состояния системы.

5. \*\*Каким образом системные средства мониторинга могут использовать журналы Windows Event Log?\*\*

- Ответ: Системные средства мониторинга могут анализировать события, записанные в журнале, для выявления проблем, мониторинга производительности и безопасности системы. Это позволяет операторам и администраторам получать информацию о состоянии системы.

6. \*\*Какие специальные API предоставляют доступ к журналам Windows Event Log?\*\*

- Ответ: Для доступа к журналам Windows Event Log используются специальные API, такие как Win32 API и PowerShell cmdlets, предоставляющие функциональность для чтения и записи событий в журналы.

7. \*\*Какие другие вспомогательные средства и инструменты могут использоваться для управления системой и мониторинга событий?\*\*

- Ответ: В дополнение к реестру и журналам Windows Event Log, существуют другие средства и инструменты, такие как Group Policy, Task Scheduler и Performance Monitor, которые позволяют администраторам управлять системой, планировать задачи и мониторить ресурсы и производительность.

8. \*\*Какие задачи могут быть решены с помощью управления реестром Windows?\*\*

- Ответ: Управление реестром Windows может использоваться для настройки параметров операционной системы, установки приложений, управления драйверами и службами, а также для многих других задач, связанных с настройкой и конфигурацией системы.

9. \*\*Почему важно вести мониторинг и регистрацию событий в системе?\*\*

- Ответ: Мониторинг и регистрация событий в системе важны для выявления проблем, отслеживания действий пользователей и служб, а также для обеспечения безопасности и производительности системы.

10. \*\*Какие действия можно предпринять на основе информации, полученной из журналов Windows Event Log?\*\*

- Ответ: На основе информации из журналов Windows Event Log можно предпринимать действия, такие как устранение проблем, оптимизация производительности, мониторинг безопасности и аудит системы, а также анализ событий для определения трендов и планирования обслуживания.

ЛР6

Теор часть:

\*\*Средства синхронизации и взаимного исключения (Windows)\*\*

Средства синхронизации и взаимного исключения играют важную роль в разработке многозадачных и многопоточных приложений под операционной системой Windows. Эти механизмы предназначены для координации доступа нескольких потоков к общим ресурсам и обеспечения правильного выполнения параллельных задач. Важно понимать, какие средства предоставляет Windows для этой цели и как их использовать.

\*\*Средства синхронизации в Windows включают в себя:\*\*

1. \*\*Мьютексы (Mutexes):\*\*

Мьютексы предоставляют механизм блокировки доступа к ресурсам для других потоков. Они используются для обеспечения взаимного исключения, позволяя только одному потоку одновременно выполнять критические секции кода.

2. \*\*Семафоры (Semaphores):\*\*

Семафоры позволяют контролировать доступ к ресурсам для определенного числа потоков. Например, семафор счетчика равного 1 может использоваться как мьютекс.

3. \*\*Критические секции (Critical Sections):\*\*

Критические секции - это специальные объекты, предоставляемые Windows API, для организации взаимного исключения. Они обеспечивают блокировку доступа к ресурсам только для потоков, выполняющихся в одном процессе.

4. \*\*События (Events):\*\*

События используются для уведомления одного или нескольких потоков о наступлении события. Потоки могут ждать наступления события и затем продолжить выполнение.

\*\*Изучение и использование средств синхронизации в Windows:\*\*

- \*\*Создание мьютексов и семафоров:\*\*

Для создания мьютексов и семафоров можно использовать функции Windows API, такие как `CreateMutex` и `CreateSemaphore`. Эти функции возвращают дескрипторы объектов синхронизации, с которыми потоки могут взаимодействовать.

- \*\*Использование критических секций:\*\*

Критические секции могут быть использованы с помощью объектов `CRITICAL\_SECTION`, предоставляемых Windows API. Для входа в критическую секцию используется функция `EnterCriticalSection`, а для выхода - `LeaveCriticalSection`.

- \*\*Работа с событиями:\*\*

События можно создавать с помощью функции `CreateEvent`. Потоки могут ожидать наступления события с использованием `WaitForSingleObject` или `WaitForMultipleObjects`.

- \*\*Синхронизация доступа к ресурсам:\*\*

Потоки, конкурирующие за доступ к общим ресурсам, могут использовать средства синхронизации для обеспечения взаимного исключения. Это предотвращает возникновение состояний гонки данных и обеспечивает целостность данных.

- \*\*Избегание дедлоков:\*\*

При использовании средств синхронизации, важно учитывать возможность дедлоков, когда потоки блокируют друг друга. Для предотвращения дедлоков можно использовать стратегии, такие как порядок блокировки и использование таймаутов.

\*\*Мьютексы (Mutexes):\*\*

Мьютексы - это средство синхронизации в операционных системах, включая Windows. Они представляют собой объекты, которые могут находиться в разных состояниях: заблокированном и разблокированном. Мьютексы используются для обеспечения взаимного исключения, что означает, что только один поток может владеть мьютексом в заданный момент времени. Если другой поток попытается получить доступ к заблокированному мьютексу, он будет заблокирован до тех пор, пока первый поток не освободит мьютекс. Мьютексы применяются для синхронизации доступа к общим ресурсам, таким как файлы, память или другие критические участки кода.

\*\*Семафоры (Semaphores):\*\*

Семафоры - это абстрактные счетчики, которые используются для контроля доступа нескольких потоков к ресурсам. Семафор имеет начальное значение и может быть увеличен (семафор "поднимается") или уменьшен (семафор "опускается"). Если семафор имеет значение больше нуля, поток может уменьшить его и продолжить выполнение. Если семафор равен нулю, поток будет заблокирован до тех пор, пока значение семафора не станет больше нуля. Семафоры позволяют ограничивать количество потоков, имеющих доступ к определенному ресурсу.

\*\*Фильтры событий (Event Flags - "фенч"):\*\*

Фильтры событий, также известные как флаги событий, представляют собой бинарные значения, которые могут находиться в двух состояниях: установлены или сброшены. Эти объекты используются для уведомления одного или нескольких потоков о наступлении события. Потоки могут ожидать, когда фильтр событий устанавливается, и после этого продолжать выполнение. Фильтры событий часто используются для сигнализации между потоками и синхронизации задач, когда одна задача должна дождаться выполнения другой.

В контексте Windows, эти средства синхронизации предоставляются Windows API и предоставляют программистам мощные инструменты для управления и синхронизации потоков в многозадачных приложениях.

\*\*Итоги:\*\*

Изучение и использование средств синхронизации и взаимного исключения в Windows являются важной частью разработки многозадачных и многопоточных приложений. Правильное использование этих средств позволяет обеспечить правильную координацию и синхронизацию потоков, предотвращая состояния гонки данных и обеспечивая безопасный доступ к общим ресурсам.

Разбор кода:

Этот код представляет собой пример простой игры "Змейка" с использованием библиотеки Windows API. Давайте разберем, что и где используется в этом коде:

1. Включение заголовочных файлов:

```cpp

#include <windows.h>

#include <deque>

#include <fstream>

#include <thread>

#include <mutex>

#include <string>

#include <chrono>

#include <fstream>

#include "resource.h"

```

Здесь включены необходимые заголовочные файлы, включая файлы для работы с Windows API, потоками, мьютексами, и т. д.

2. Объявление и инициализация глобальных переменных и структур данных:

```cpp

const int gridSize = 25;

const int screenWidth = 1100;

const int screenHeight = 800;

const wchar\_t\* saveFileName = L"D:\\Study\\OSiSP\\LR1\\LR1\\snake\_save.bin";

const wchar\_t\* sharedMemoryName = L"MySharedMemory";

std::thread inputThread;

std::thread updateGameThread;

HWND hWnd;

HWND restartButton;

HHOOK g\_hook = NULL;

// И другие переменные.

```

В этой части кода объявлены глобальные переменные, такие как размер игровой сетки, размер экрана, пути к файлам сохранения, обработчики окон и потоки для управления игрой.

3. Определение структуры `SnakeSegment` для хранения координат сегментов змейки.

4. Объявление мьютекса, области разделяемой памяти и других средств синхронизации.

5. Различные функции, такие как `SetProcessPriority`, `CreateMemoryMapping`, `CloseMemoryMapping`, `SaveGame`, и `LoadGame`, которые выполняют различные задачи, включая установку приоритета процесса, создание разделяемой памяти, сохранение и загрузку состояния игры.

6. Функции для отрисовки игровых элементов, такие как `DrawCell`, `PaintGame`, и `UpdateGame`.

7. Функции для обработки пользовательского ввода и обновления состояния игры.

8. Главная функция `WinMain`, которая выполняет инициализацию окна, создание потоков, цикл обработки сообщений, и освобождение ресурсов при завершении игры.

Теперь давайте рассмотрим вопросы, которые могут быть заданы по этому коду и теме:

\*\*Вопросы, связанные с кодом:\*\*

1. Как реализована отрисовка игрового поля и элементов (змейки, еды) в функции `PaintGame`?

2. Для чего используется мьютекс `snakeMutex` в коде?

3. Что происходит при нажатии клавиши "R"?

4. Какие функции отвечают за обработку пользовательского ввода?

5. Как реализована синхронизация потоков в данной программе?

6. Какая структура данных используется для хранения координат сегментов змейки?

7. Как работает разделяемая память в этой программе?

8. Что делает функция `SetProcessPriority`?

9. Какие действия выполняются в функции `UpdateGame`?

\*\*Вопросы, связанные с темой "Средства синхронизации и взаимного исключения (Windows)":\*\*

1. Какие средства синхронизации используются в данном коде?

2. Как работают мьютексы и зачем они нужны в многозадачных приложениях?

3. Для чего предназначены условные переменные и как они применяются в коде?

4. Каким образом обеспечивается синхронизация доступа к общим данным в этой программе?

5. В чем заключается использование разделяемой памяти в контексте синхронизации?

Эти вопросы помогут понять, как работает данный код и какие средства синхронизации и средства взаимного исключения используются для обеспечения корректного выполнения многопоточного приложения.

Вопросы по коду:

В данном коде присутствуют средства синхронизации и взаимного исключения для обеспечения безопасной работы многопоточной программы. Давайте рассмотрим, где и как они используются:

1. \*\*Мьютексы:\*\*

- Мьютекс `snakeMutex` используется для синхронизации доступа к общей структуре данных `snake` (змейке). Он блокируется при добавлении и удалении сегментов змейки, чтобы предотвратить гонки данных между потоками. Например, он используется при обновлении состояния игры в функции `UpdateGame`.

2. \*\*Условные переменные:\*\*

- Условная переменная `conditionVariable` используется для сигнализации между потоками. Она связана с мьютексом и используется, чтобы потоки могли ждать определенного условия перед продолжением выполнения. Однако, в данном коде условная переменная не используется явно, и её наличие может быть связано с предполагаемым использованием в будущих версиях программы.

3. \*\*Инициализация мьютекса и условной переменной:\*\*

- Инициализация мьютекса происходит с помощью функции `InitializeCriticalSection(&cs)`.

- Инициализация условной переменной происходит с помощью функции `InitializeConditionVariable(&conditionVariable)`.

4. \*\*Блокировка и разблокировка мьютекса:\*\*

- Мьютекс блокируется с помощью `lock\_guard<mutex> lock(snakeMutex)` в разных частях кода, чтобы гарантировать, что доступ к данным змейки выполняется только одним потоком одновременно. Например, в функциях `UpdateGame` и `SaveGame`.

Таким образом, мьютексы и условные переменные используются в данном коде для обеспечения правильной синхронизации и взаимного исключения при работе с общими данными, такими как состояние змейки и разделяемая память.

Общие вопросы:

Общие вопросы по теме "Средства синхронизации и взаимного исключения в Windows" могут включать следующее:

1. \*\*Что такое средства синхронизации в операционной системе Windows?\*\*

- Средства синхронизации - это механизмы, предоставляемые операционной системой для координации и синхронизации работы потоков в многозадачных приложениях.

2. \*\*Зачем нужны средства синхронизации?\*\*

- Средства синхронизации необходимы для предотвращения гонок данных и обеспечения правильной синхронизации доступа к общим ресурсам, таким как переменные или структуры данных, из нескольких потоков.

3. \*\*Какие средства синхронизации предоставляет Windows?\*\*

- Windows предоставляет различные средства синхронизации, такие как мьютексы, семафоры, условные переменные, критические секции, события, и другие.

4. \*\*Что такое мьютексы в Windows?\*\*

- Мьютексы (mutex) - это средство синхронизации, которое позволяет одному потоку одновременно захватывать и освобождать ресурс (критическую секцию), чтобы предотвратить одновременный доступ нескольких потоков.

5. \*\*Что такое условные переменные в Windows?\*\*

- Условные переменные (conditional variables) позволяют потокам ожидать определенных условий и сигнализировать о выполнении этих условий другим потокам.

6. \*\*Чем отличаются мьютексы от критических секций?\*\*

- Мьютексы могут быть использованы между процессами, в то время как критические секции предназначены для использования только внутри одного процесса.

7. \*\*Какие функции используются для работ с мьютексами и критическими секциями в Windows?\*\*

- Для работы с мьютексами используются функции, такие как `CreateMutex`, `WaitForSingleObject`, и `ReleaseMutex`. Для критических секций - `InitializeCriticalSection`, `EnterCriticalSection`, и `LeaveCriticalSection`.

8. \*\*Зачем нужны условные переменные, и какие функции используются для их работы?\*\*

- Условные переменные позволяют потокам ожидать определенных условий и сигнализировать об их выполнении. Для работы с условными переменными используются функции `InitializeConditionVariable`, `SleepConditionVariableCS`, и `WakeConditionVariable`.

9. \*\*Какие средства синхронизации выбирать в зависимости от ситуации?\*\*

- Выбор средства синхронизации зависит от конкретных требований задачи. Например, мьютексы подходят для обеспечения взаимного исключения, а условные переменные - для ожидания определенных условий.

10. \*\*Как обеспечивается безопасная работа многопоточных приложений с использованием средств синхронизации?\*\*

- Для обеспечения безопасности при работе с многопоточными приложениями, разработчики используют средства синхронизации, чтобы избежать гонок данных и непредсказуемого поведения приложений.