1. Изучение событийной архитекторы Windows-приложений, механизма обработки сообщений, механизма перерисовки окна.

* Разработать программу, позволяющую передвигать с помощью клавиатуры и мыши спрайт (окрашенный прямоугольник или эллипс) внутри рабочей области окна. Для движения спрайта без мерцаний необходимо использовать двойную буферизацию.
* Обеспечить работу колесика мыши: прокручивание двигает спрайт по вертикали; с удерживаемой клавишей Shift прокручивание колесика двигает спрайт по горизонтали.
* Заменить спрайт на картинку с непрямоугольным контуром, т.е. на прозрачном фоне. Допускается использование GDI+ API.

2. Работа с таймером.

* Придать спрайту из лабораторной работы 1 самостоятельное движение с отскоком от границ окна. Реализовать периодический пересчет координат спрайта двумя способами:
  + с помощью оконного таймера (SetTimer);
  + с помощью ожидаемого таймера (Waitable Timer) и пользовательского оконного сообщения (WM\_USER). *Подсказка: учтите, что для вызова обработчика pfnCompletionRoutine ожидаемый таймер использует механизм* [*APC - Asynchronous Procedure Calls*](https://docs.microsoft.com/en-us/windows/win32/sync/asynchronous-procedure-calls)*. Обратите внимание на условие, при котором вызывается user-mode APC.*

Переключение используемого вида таймера при демонстрации лабораторной работы - на усмотрение студента (директивы препроцессора, два отдельных проекта, …)

3. Изучение создания и использования динамически-загружаемых библиотек DLL.

* Разработать DLL с функцией рисования правильного N-угольника, который можно вписать в окружность заданного радиуса с центром в заданной позиции. [N = (номер-в-списке mod 10) + 3].
  + Сигнатура функции: **void draw(HDC hdc, int x, int y, int radius)**.
* Разработать программу, которая импортирует DLL из папки plugins и вызывает функцию draw для рисования фигуры в позиции курсора при нажатии ЛКМ. Импорт DLL осуществить двумя способами: неявно (*\_\_declspec(dllimport)*) и явно (функцией *LoadLibrary*).
* Предусмотреть добавление в папку plugins других DLL, экспортирующих функцию с такой же сигнатурой, с возможностью переключения между библиотеками во время работы программы: по нажатию на ПКМ выгрузить текущую и загрузить следующую DLL. После смены DLL должна измениться рисуемая фигура.

4. Изучение использования встроенного пула потоков и механизмов синхронизации.

* Разработать потокобезопасную коллекцию на основе списка. Необходимо использовать один из примитивов синхронизации, выбор обосновать:
  + Критическая секция,
  + Семафор,
  + Мьютекс.
* ***Доп. задание по желанию:*** *реализовать* ***неблокирующую*** *потокобезопасную коллекцию.*
* Разработать программу, которая осуществляет поиск заданной строки во всех файлах указанного каталога на верхнем уровне вложенности.
* Главный поток перечисляет файлы в каталоге, для каждого из них создаёт задачу и запускает её в пуле потоков (*SubmitThreadpoolWork*). В качестве аргумента задаче передается путь к файлу.
* Задача считает количество вхождений заданной строки в файле и сохраняет результат вместе с именем файла в разработанную потокобезопасную коллекцию.
* Главный поток дожидается завершения всех задач и выводит результаты поиска в формате: *“<имя-файла>: <количество-вхождений-строки>”*. Для ожидания использовать *Условную переменную* и *Критическую секцию* (см. слайды 80-81):
  + заводится счётчик незавершённых задач;
  + перед добавлением новой задачи главный поток увеличивает счётчик на 1;
  + при завершении задачи она самостоятельно уменьшает счётчик на 1;
  + главный поток ожидает, когда счётчик обнулится.

5. Создание, установка и отладка драйвера Windows режима ядра.

* [Следуя данному руководству](https://learn.microsoft.com/en-us/windows-hardware/drivers/gettingstarted/writing-a-very-small-kmdf--driver), создать и скомпилировать драйвер, который передаёт в отладочный вывод номер группы и фамилию студента.
* Настроить виртуальную среду для развёртывания и отладки драйвера (Hyper-V, VMWare Workstation, Oracle VirtualBox или др.).
* Подключиться к виртуальной среде из Visual Studio в режиме отладки. Продемонстрировать результат работы драйвера.

Текстовый урок по выполнению лабораторной: <https://docs.google.com/document/d/1tPGgoMzL1AHMjSsH8sO63x_8cbHiMVLuQPmiR8M2hA4/edit?usp=sharing>

Видеоурок: <https://www.youtube.com/watch?v=LBb7C6NpZ7E>

6. Изучение методов перехвата API-вызовов в режиме ядра.

* Разработать драйвер (виртуального устройства), который отслеживает запуск некоторого процесса X. При запуске этого процесса драйвер запускает другой процесс Y. Как только процесс X завершается по какой-то причине, драйвер завершает процесс Y.

ИЛИ

* Разработать драйвер (виртуального устройства), который отслеживает изменения в реестре Windows заданным процессом и записывает их в файл на диске.