***Задание для выполнения лаб. работы №1***

Написать программу, которая должна “озвучивать” клавиатуру, т.е. после запуска этой программы нажатие любой клавиши на клавиатуре будет сопровождаться звуковым сигналом. Клавиатура при этом должна оставаться работоспособной, т.е. продолжать выполнять свои основные функции в нормальном темпе.

Программа должна быть резидентной, т.е. оставаться в памяти после своего завершения.

В качестве пробного варианта длительность звукового сигнала и частоту задать константами в программе. Когда будет получена устойчивая работа программы, изменить её таким образом, чтобы длительность звукового сигнала и его частота задавались в качестве параметров при запуске программы.

Необходимо предоставить пользователю возможность “выключать” и “включать” заново звуковое сопровождение работы клавиш. Использовать для “выключения/включения” звука нестандартную комбинацию клавиш: сочетание нажатой клавиши <Shift> с какой-либо ещё, например, <Shift>+<Esc> …

**Лабораторная работа №2:   
*Управление таймером – операции в реальном времени   
(программа “будильник”)***

Категорически запрещено в процедуре – обработчике прерываний таймера использовать стандартную процедуру задержки (delay)!

***Задание для выполнения лаб. работы №2***

Написать резидентную программу, которая будет работать, как “будильник” – через заданный интервал времени издавать короткий звуковой сигнал – “тикать” (например, через 1–2 секунды). По завершении более длительного интервала времени (от нескольких секунд до нескольких минут или часов) должен раздаваться более продолжительный мелодичный звуковой сигнал, имитирующий звонок будильника.

Длительность звукового сигнала – “тиканья” – не должна быть слишком большой. Её следует задавать в программе в пределах от 1/10 до ½ доли секунды.

По окончании “звонка будильника” “тиканье” должно продолжаться. “Звонок” должен раздаваться только один раз.

В то время, когда звучит “звонок”, не должно быть слышно “тиканья” (звонок может длиться в течение нескольких секунд, и в этот интервал времени теоретически могут попасть звуки “тиканья”).

Для получения эффекта “мелодичного звонка” следует использовать несколько звуковых частот с различной продолжительностью звучания каждой из них.

Временной интервал, через который должно происходить “тиканье” (в секундах – целое число секунд), задавать с клавиатуры в качестве параметра; время, через которое должен прозвонить будильник (в минутах – их число может быть дробным), тоже задавать в качестве параметра. Отсчёт времени для “звонка” вести с момента запуска программы.

**Лабораторная работа №3: *Работа с экраном (“часы”)***

***Задание для выполнения лаб. работы №3***

Написать программу “часы”, которая будет однократно перехватывать показания системных часов и от них вести отсчет времени, используя собственный счетчик прерываний таймера (в качестве основы может быть использован обработчик прерываний таймера из предыдущей работы). Показания времени – часы, минуты, секунды – следует выводить в заданном месте экрана. Формат вывода: xx:xx:xx. Обновление показаний времени выполнять дискретно, через заданное количество секунд.

Программа должна работать резидентно. При обновлении показаний времени курсор не должен быть виден, кроме того, вывод показаний времени не должен нарушать нормальной работы пользователя.

Часы должны работать точно! Для обеспечения этого, во-первых, необходимо помнить о том, что количество импульсов (или “тиков”) таймера в секунду – число дробное. Следовательно, на одной секунде может возникать некоторая погрешность, которую возможно устранить на 5 секундах (когда пройдет целое количество импульсов). Во-вторых, необходимо учитывать, что минимальная величина, отраженная в показаниях часов – это секунды. Но программа может быть запущена в начале секунды (прошел 1 импульс) или же в конце (прошло 18 импульсов). Если начинать отсчет “тиков” таймера с нуля, может возникнуть погрешность величиной почти в секунду. Следовательно, при установке начального значения счетчика импульсов таймера необходимо учитывать ещё и доли секунды (один из параметров стандартной процедуры GetTime), переведя их в соответствующее количество импульсов.

Величину дискрета, а также координаты местоположения часов на экране задавать с клавиатуры в качестве параметров командной строки (т.е. программа должна иметь три параметра) при запуске программы.

**Лабораторная работа №4:   
*Эмуляция работы параллельных процессов на примере   
схемы “производитель–потребитель”***

***Задание для выполнения лаб. работы №4***

Написать программу, которая будет эмулировать параллельную работу некоторых потоков. Потоки должны работать циклически. В качестве модели использовать схему “производитель – потребитель”. Один поток (производитель) может помещать случайные (или какие-то определенные – например, только четные числа или квадраты целых чисел и т.п.) числа в буфер (массив заданного размера), для наглядности поток-производитель должен эти числа выводить на экран. Другой поток (потребитель) забирает числа из этого буфера. Для контроля также выполнять вывод на экран чисел, взятых потоком-потребителем из буфера. Вывод разными потоками выполнять в разные строки и/или разным цветом; дополнительно выводить на экран индикатор того, какой именно поток работает в настоящий момент, а также содержимое буфера и текущий процент его заполненности.

На экране параллельная работа потоков может быть представлена следующим образом:

Верхняя строка (производитель): ячейка для вывода текущего сгенерированного числа, признак активности потока (слово, символ, цвет), сообщение о переполнении буфера в случае этого события. Возможно, ещё какая-то полезная информация, например, номер заполняемой ячейки.

Нижняя (или вторая) строка (потребитель): ячейка для вывода текущего прочитанного числа, признак активности потока (слово, символ, цвет), сообщение о пустом буфере в случае этого события. Возможно, информация о номере считываемой из буфера ячейки.

В середине экрана: сам буфер, в который числа добавляются потоком-производителем и из которого считываются (удаляются или перекрашиваются, попадая при этом в его ячейку в нижней строке экрана) потоком-потребителем. Считывание чисел можно производить по принципу стека или очереди. При считывании по принципу очереди после завершения работы потребителя какое-то количество чисел из начала буфера исчезнет, следовательно, буфер будет перемещаться по экрану и в какой-то момент его потребуется переписать заново, от начала.

Отдельной строкой или в углу экрана отображать процент заполненности буфера.

Предусмотреть обработку критических ситуаций:

1) Случай, когда потребителю предоставлено управление, а буфер данных пуст – тогда активный поток должен напрямую отдать управление производителю, а сам уйти в режим ожидания. При этом вопрос с квантом времени для производителя может быть решён по-разному. Например, остаток недоработанного потребителем кванта может быть передан производителю, либо ему может быть выделен новый квант времени.

2) Случай, когда управление предоставлено производителю, а буфер полон и записывать результаты некуда – поток-производитель должен заблокироваться до появления свободного места в буфере и запустить поток-потребитель. Вопрос с квантом может решаться аналогично.

Для того чтобы было возможно пронаблюдать работу потоков в замедленном режиме, в каждом из потоков следует поставить дополнительную задержку (стандартный delay), величину которой задавать с клавиатуры при запуске программы, в качестве параметра командной строки. При запуске без параметров выводить сообщение примерного вида: “Программа запущена со стандартной задержкой, величина которой =…” и формат запуска программы для задания желаемой задержки.

Потоки при работе чередуются случайным образом; регламентировать их работу с помощью таймера (выделять каждому кванты времени, величина которых тоже случайна – в некотором диапазоне). При этом может складываться ситуация, что один и тот же поток несколько раз подряд получит управление. Таймер по окончании выделенного потоку кванта времени изменяет статус этого потока с активного на пассивный, в результате чего внутренний цикл этого потока должен завершиться.

Внутри обработчика прерываний таймера не может находиться вызовов процедур – потоков! Вызовы процедур должны происходить в бесконечном цикле в основной программе. В обработчик прерываний таймера вообще нельзя включать никакие действия, требующие длительного выполнения, например, вызовы циклических процедур, или процедур, работающих с графикой или с диском.

Для выхода из программы предусмотреть какую-то специальную клавишу или комбинацию клавиш (выбор по желанию программиста), информация о ней должна быть известна пользователю – помещена на экране. При нажатии этой клавиши происходит окончание работы потока-производителя, а поток-потребитель закончит свою работу, только когда буфер будет исчерпан, т.е. выработанная информация будет полностью использована.

**Лабораторная работа №5:   
*Работа с видеопамятью ("экран")***

***Задание для выполнения лаб. работы №5***

Написать программу, которая будет работать с видеопамятью. Можно выполнить любое из приведенных ниже заданий. Независимо от конкретного задания программа должна работать резидентно. Текст, присутствовавший на экране до запуска программы, не должен портиться никаким образом.

В программе следует использовать два обработчика прерываний – прерываний таймера и прерываний клавиатуры. Прерывания таймера регламентируют скорость перемещения объекта, причем параметр, характеризующий величину скорости, необходимо задавать с клавиатуры при запуске программы. Информация о параметре, задающем скорость движения, должна выводиться на экран при запуске программы без параметров.

В случае смещения строк на экране (нажатие клавиши ENTER) не должно происходить никаких нежелательных эффектов – раздвоения картинки, дублирования строк или их частей…

**Вариант 5(1)** **“Свободное движение”**:

Черный (или цветной) квадратик (или иной объект) “летает” по экрану, содержащему некоторый текст, и отражается от границ экрана. Движение продолжается до нажатия клавиши <Esc> или некоторой нестандартной комбинации клавиш.

**Вариант 5(2)** **“Управляемое движение”**:

Перемещение управляемого черного (или цветного) квадратика (или иного объекта) по экрану, содержащему некоторый текст. Управление объектом следует осуществлять с помощью клавиш-стрелок в сочетании с <Ctrl>, <Shift> или <Alt>, обрабатывать их нажатие с помощью прерываний клавиатуры. Однократное нажатие управляющей комбинации клавиш меняет направление движения объекта, а скорость его движения определяется таймером. Объект должен перемещаться циклически – “уйдя” за правую границу экрана, должен появиться слева, и т.д.

**Вариант 5(3)** **“Бегущая строка”**:

По экрану, содержащему некоторую информацию, “бежит” текстовая строка. Положение строки и её текст следует задавать параметрами. Строка “бежит” справа налево, при этом по мере того, как первые символы строки скрываются за границей экрана, они должны появляться из-за его правой границы (т.е. экран как бы замкнут).