|  |  |  |
| --- | --- | --- |
| **Дата** | **Содержание выполняемой работы** | Подпись  руководителя |
| 02.04.2021 | **Практическая работа №11**  Тема: Многопоточные приложения.  Наименование работы: Потоки. Многопоточные приложения. Вторичный поток. Класс Thread.  Делегаты ThreadStart и ParametrizedThreadStart .  Цель: Сформировать навыки создания программ с использованием основного и вторичного потоков.  Задание 1.  Изучил предложенную теорию. Кратко законспектировал.  Потоки.  Поток представляет собой независимую последовательность инструкций в программе. Каждый процесс состоит из ресурсов, таких как оконные дескрипторы, файловые дескрипторы и другие объекты ядра, имеет выделенную область в виртуальной памяти и содержит как минимум один поток. Потоки планируются к выполнению операционной системой. У любого потока имеется приоритет, счетчик команд, указывающий на место в программе, где происходит обработка, и стек, в котором сохраняются локальные переменные потока. Стек у каждого потока выглядит по-своему, но память для программного кода и куча разделяются среди всех потоков, которые функционируют внутри одного процесса.  Класс Thread.  Класс Thread является самым элементарным из всех типов пространства имен System.Threading. Этот класс представляет объектно-ориентированную оболочку вокруг заданного пути выполнения внутри определенного AppDomain. Этот тип также определяет набор методов (как статических, так и уровня экземпляра), которые позволяют создавать новые потоки внутри текущего AppDomain, а также приостанавливать, останавливать и уничтожать определенный поток.  Создание вторичных потоков.  При программном создании дополнительных потоков для выполнения некоторой единицы работы необходимо следовать строго регламентированному процессу:   1. Создать метод, который будет точкой входа для нового потока. 2. Создать новый делегат ParametrizedThreadStart (или ThreadStart), передав конструктору адрес метода, определенного на предыдущем шаге. 3. Создать объект Thread, передав в качестве аргумента конструктора ParametrizedThreadStart/ThreadStart. 4. Установить начальные характеристики потока (имя, приоритет и т.п.). 5. Вызвать метод Thread.Start(). Это запустит поток на методе, который указан делегатом, созданным на втором шаге, как только это будет возможно.   Делегаты ThreadStart и ParametrizedThreadStart.  Согласно второму шагу, можно использовать два разных типа делегатов для "указания" метода, который выполнит вторичный поток. Делегат ThreadStart относится к пространству имен System.Threading, начиная с .NET 1.0, и он может указывать на любой метод, не принимающий аргументов и ничего не возвращающий. Этот делегат пригодится, когда метод предназначен просто для запуска в фоновом режиме, без какого-либо дальнейшего взаимодействия.  Класс AutoResetEvent.  Простой и безопасный к потокам способ заставить один поток ожидать завершения другого потока, предусматривает использование класса AutoResetEvent. В потоке, который должен ждать, создадим экземпляр этого класса и передадим конструктору false, указав, что уведомления пока не было. В точке, где требуется ожидать, вызовем метод WaitOne().  Проблемы многопоточности.  Многопоточное программирование является далеко не простой задачей. При запуске множества потоков, получающих доступ к одним и тем же данным, могут возникать трудно выявляемые проблемы — состязания за ресурсы и взаимоблокировки.  Задание 2.  Создал программы по образцу. Вставил скриншот выполнения программы.  Задание 2.1  Создать приложение с вызовом функций в потоках. Листинг кода переписал в Приложение 10. Результат выполнения программы представлен на рисунке 22    Рисунок 22 – результат выполнения программы  Задание 2.2  Создать приложение по обработке массива в потоках. Листинг кода переписал в приложение 10. Результат выполнения программы представлен на рисунке 23.    Рисунок 23 – результат выполнения программы  **Контрольные вопросы**   1. Укажите, для чего предназначен поток.   Поток (thread) представляет собой независимую последовательность инструкций в программе.   1. Укажите, какие параметры имеет поток.   ThreadStart и ParametrizedThreadStart.   1. Укажите, в каких состояниях может находиться поток.   Поток может быть выполняющимся, готовым к выполнению, приостановленным, возобновлённым в дальнейшем, заблокированным, и завершённым.   1. Укажите, для чего предназначена синхронизация.   Она позволяет координировать выполнение потоков вполне определённым образом.   1. Укажите методы и свойства класса Thread.   CurrentContext, CurrentThread, GetDomain(), GetDomainID(), Sleep(), IsAlive, IsBackground, Name, Priority, ThreadState, About(), Interrupt(), Join(), Resume(), Start(), Suspedn().   1. Укажите назначение класса Thread.   Этот класс представляет объектно-ориентированную оболочку вокруг заданного пути выполнения внутри определенного AppDomain. Этот тип также определяет набор методов, которые позволяют создавать новые потоки внутри текущего AppDomain, а также приостанавливать, останавливать и уничтожать определенный поток.   1. Перечислите этапы создания вторичного потока.    1. Создать метод, который будет точкой входа для нового потока.    2. Создать новый делегат ParametrizedThreadStart, передав конструктору адрес метода, определенного на предыдущем шаге.    3. Создать объект Thread, передав в качестве аргумента конструктора ParametrizedThreadStart/ThreadStart.    4. Установить начальные характеристики потока.    5. Вызвать метод Thread.Start(). Это запустит поток на методе, который указан делегатом, созданным на втором шаге, как только это будет возможно. 2. Перечислите проблемы многопоточности.   Состязание за ресурсы и взаимоблокировка. |  |