|  |  |  |
| --- | --- | --- |
| Дата | Содержание выполняемой работы | Подпись  руководителя |
| 01.04.2021 | **Практическая работа №10**  Тема: Делегаты, события, потоки выполнения. Многопоточные приложения.  Наименование работы: Делегаты. События. Вызов делегатов. Наступление события.  Цель: Сформировать навыки создания программ с использованием делегатов. Сформировать навыки создания программ с оповещением о наступлении события.  Задание 1.  Изучил предложенную теорию. Кратко законспектировал.  Делегаты.  Делегат представляет собой объект, который может ссылаться на метод. Следовательно, когда создается делегат, то в итоге получается объект, содержащий ссылку на метод. Более того, метод можно вызывать по этой ссылке. Иными словами, делегат позволяет вызывать метод, на который он ссылается.  По сути, делегат — это безопасный в отношении типов объект, указывающий на другой метод приложения, который может быть вызван позднее. В частности, объект делегата поддерживает три важных фрагмента информации:   * адрес метода, на котором он вызывается; * аргументы этого метода; * возвращаемое значение этого метода.   Как только делегат создан и снабжен необходимой информацией, он может динамически вызывать методы, на которые указывает, во время выполнения. Каждый делегат в .NET Framework автоматически снабжается способностью вызывать свои методы синхронно или асинхронно. Этот факт значительно упрощает задачи программирования, поскольку позволяет вызывать метод *во вторичном потоке выполнения без ручного создания и управления объектом Thread*.  События.  Событие представляет собой автоматическое уведомление о том, что произошло некоторое действие. События действуют по следующему принципу: объект, проявляющий интерес к событию, регистрирует обработчик этого события. Когда же событие происходит, вызываются все зарегистрированные обработчики этого события. Обработчики событий обычно представлены делегатами.  События являются членами класса и объявляются с помощью ключевого слова event. Чаще всего для этой цели используется следующая форма:  event делегат\_события имя\_события;  где делегат\_события обозначает имя делегата, используемого для поддержки события, а имя\_события — конкретный объект объявляемого события.  Задание 2.  Создал программы по образцу. Листинг кода переписал в Приложение 9. Вставил скриншоты выполнения программы.  Задание 2.1  Создание приложения с использованием делегата для вызова функции. Листинг кода задания 2.1 переписал в Приложение 9. Результат выполнения программы представлен на рисунке 19.  15.png  Рисунок 19 – результат выполнения программы  Задание 2.2  Создание приложения с использованием делегата для вызова метода класса. Листинг кода переписал в Приложение 9. Результат выполнения программы представлен на рисунке 20.  16.png  Рисунок 20 – результат выполнения программы  Задание 2.3  Создание приложения с оповещением о наступлении события. Листинг кода переписал в приложение 9. Результат выполнения программы представлен на рисунке 21.  17.png  Рисунок 21 – результат выполнения программы  **Контрольные вопросы**   1. Укажите, для чего предназначен делегат.   По сути, делегат — это безопасный в отношении типов объект, указывающий на другой метод приложения, который может быть вызван позднее.   1. Укажите, как соотносятся потоки выполнения и делегаты.   Как только делегат создан и снабжен необходимой информацией, он может динамически вызывать методы, на которые указывает, во время выполнения.   1. Укажите, какие три метода автоматически поддерживаются делегатом.   Invoke(), BeginInvoke(), EndInvoke().   1. Укажите, какие методы и свойства делегатов вы знаете.   Method, Target, Combine(), GetInvokationList(), Remove(), RemoveAll().   1. Опишите, что такое событие.   Событие представляет собой автоматическое уведомление о том, что произошло некоторое действие.   1. Укажите, для чего предназначены аксессоры события.   Аксессоры предоставляют средства для управления реализацией подобного списка в приведенной ниже форме:  Event делегат\_события имя\_события {  add {  // Код добавления события в цепочку событий  }  remove {  // Код удаления события из цепочки событий  }  } |  |