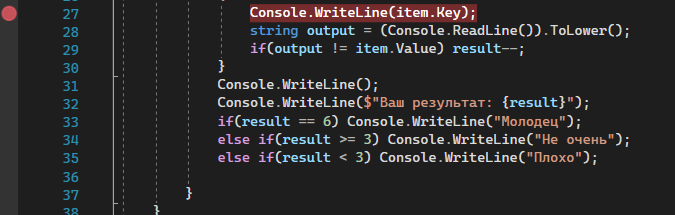
Задание. Изучить и апробировать возможности отладчика. Для этого:

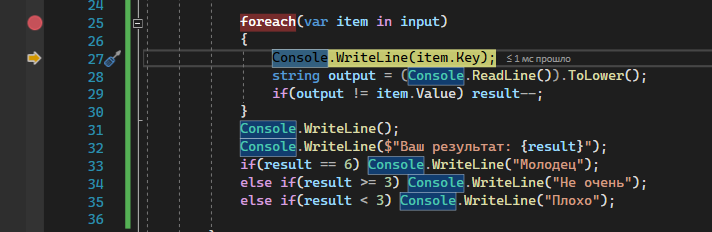
Исследовать возможности выбранного Вами интегрированного отладчика. Изучите меню и справочную информацию среды программирования. В отчет включить список реализованных в ней функций отладки и краткое их описание (в каких случаях и как используются). Можно в виде таблицы, например для C#:

|  |  |  |
| --- | --- | --- |
| Функция | | Назначение |
| № | Содержание |
| 1 | Установка и редактирование точек останова, их типа, просмотр списка | Позволяет временно прервать выполнение и исследовать состояние программы (логи, состояние памяти, регистров процессора, стека и т. п.), с тем чтобы определить, правильно ли она себя ведёт |
| 2 | Пошаговое выпол­нение программы | Приостановка выполнения программы на каждом шаге, позволяет посмотреть состояние программы |
| 3 | Шаг с обходом | Позволяет пропустить функцию, показывает результат после выполнения действия функции |
| 4 | Быстрое выполнение до точки в коде с помощью мыши | Использование кнопки Выполнение до щелкнутого аналогично установке временной точки останова. |
| 5 | Выполнить до текущей позиции | Эта команда запускает отладку и задает временную точку останова на текущей строке кода. |
| 6 | Проверка переменных с помощью подсказок по данным | Проверка состояний переменных. В режиме приостановки в отладчике наведите указатель мыши на объект, чтобы увидеть его текущее значение или значение по умолчанию. |
| 7 | Проверка переменных с помощью окон "Видимые" и "Локальные" | В окне Видимые отображаются переменные вместе с текущим значением и типом. Окно Видимые содержит все переменные, используемые в текущей строке или предыдущей строке. В окне Локальные показаны переменные, которые находятся в текущей области. |
| 8 | Установка контрольного значения | В окне Контрольное значение можно указать переменную (или выражение), которую необходимо отслеживать. У вас есть контрольное значение, заданное для объекта, и по мере перемещения по отладчику вы можете наблюдать за изменением его значения. |
| 9 | Просмотр стека вызовов | В окне Стек вызовов показан порядок вызова методов и функций. В верхней строке показана текущая функция. Во второй строке показана функция или свойство, из которого она вызывалась, и т. д. Стек вызовов хорошо подходит для изучения и анализа потока выполнения приложения. |
| 10 | Проверка исключения | Когда приложение выдает исключение, отладчик переходит к строке кода, вызвавшей исключение. |

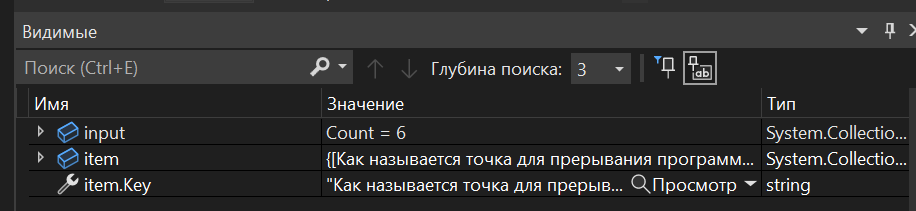
Установка точки отсанова:

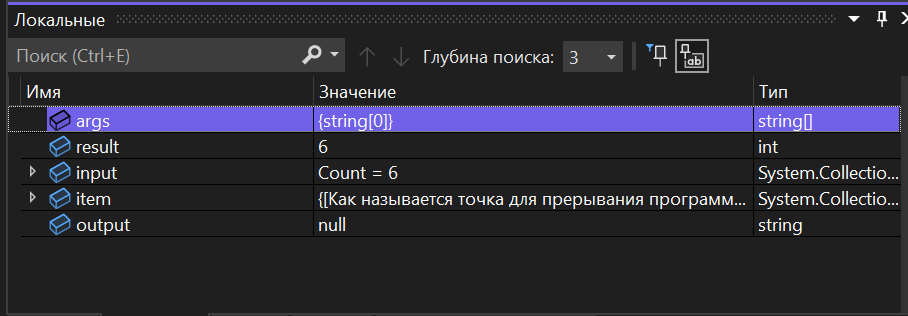


Пошаговое выполнение программы:

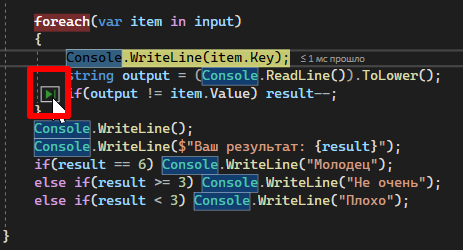


Проверка переменных видимые и локальные:

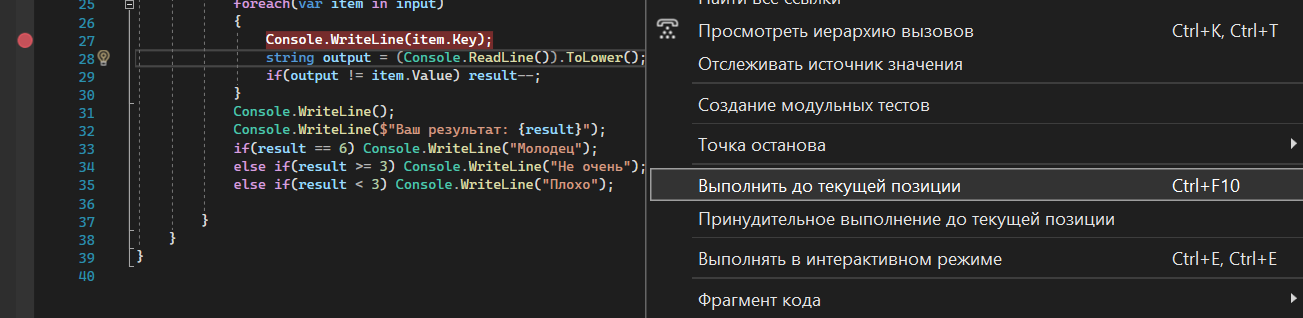




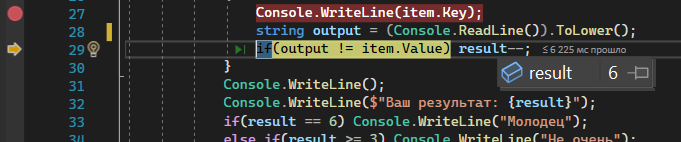
Быстрое выполнение до точки в коде с помощью мыши:



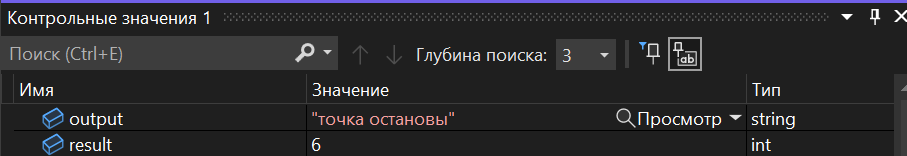
Выполнить до текущей позиции:



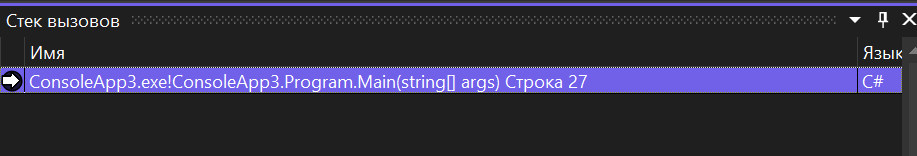
Проверка переменных с помощью подсказок по данных:



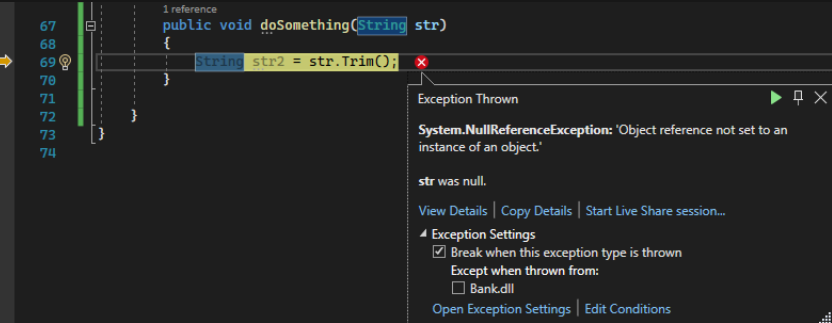
Установка контрольного значения:



Просмотр стека вызовов:



Проверка исключений:



Вывод:

Код программы:

static void Main(string[] args)

{

int result = 6;

Dictionary<string, string> input = new Dictionary<string, string>()

{

{"Как называется точка для прерывания программы?", "точка остановы"},

{"Можно ли выполнять программу пошагово?", "да"},

{"Какое сейчас время года?", "весна"},

{"Напишите название предмета", "тестирование программного обеспечения"},

{"Кто президент России?(Написать фамилию)", "путин"},

{"Кто президент США?(Написать фамилию)", "байден"}

};

foreach(var item in input)

{

Console.WriteLine(item.Key);

string output = (Console.ReadLine()).ToLower();

if(output != item.Value) result--;

}

Console.WriteLine();

Console.WriteLine($"Ваш результат: {result}");

if(result == 6) Console.WriteLine("Молодец");

else if(result >= 3) Console.WriteLine("Не очень");

else if(result < 3) Console.WriteLine("Плохо");

}