### Шаг 1. Сбор данных профилирования

1)Откройте проект для отладки в Visual Studio и установите точку останова в приложении в точке, где вы хотите проверить загрузку ЦП.

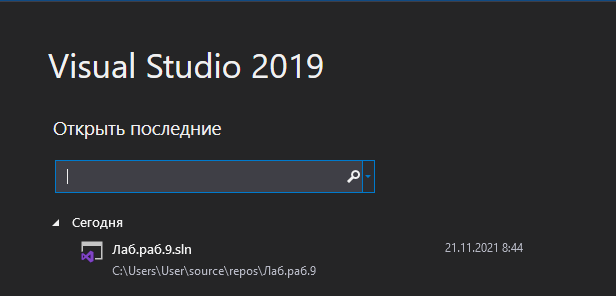


Рис 1

2)Установите вторую точку останова в конце функции или области кода, который требуется проанализировать.

С помощью двух точек останова можно ограничить сбор данных частями кода, которые требуется проанализировать.

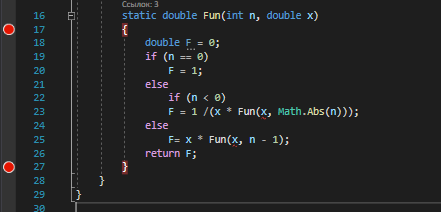


Рис 2

3) Окно **Средства диагностики** появится автоматически, если вы не отключали эту функцию. Чтобы снова открыть окно, щелкните **Отладка** > **Окна** > **Показать средства диагностики**

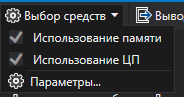


Рис 3

4) Вы можете выбрать, что следует просмотреть, Использование памяти или **Загрузка ЦП** (либо оба средства), с помощью параметра **Выбор средств** на панели инструментов. В Visual Studio Enterprise также можно включить или отключить IntelliTrace, выбрав **Сервис** > **Параметры** > **IntelliTrace**

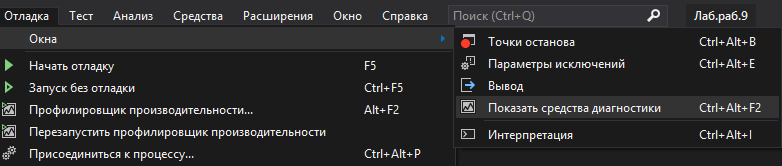


Рис 4

5)Щелкните **Отладка** > **Начать отладку** (**Запустить** на панели инструментов или **F5**).

По завершении загрузки приложения отображается представление "Сводка" средств диагностики. Если вам нужно открыть окно, щелкните **Отладка** > **Окна** > **Показать средства диагностики**.

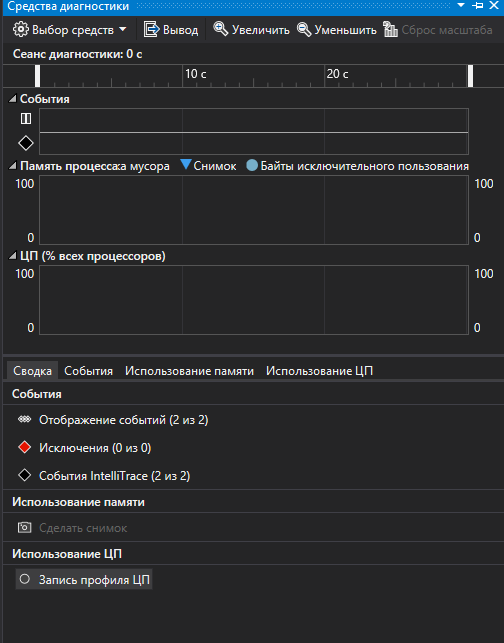


Рис 5

6) Запустите сценарий, который вызвал срабатывание первой точки останова.

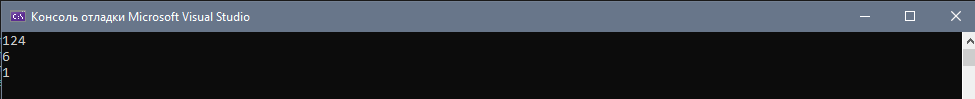


Рис 6

7 )Приостановив отладчик, включите сбор данных о загрузке ЦП, а затем откройте вкладку **Загрузка ЦП**

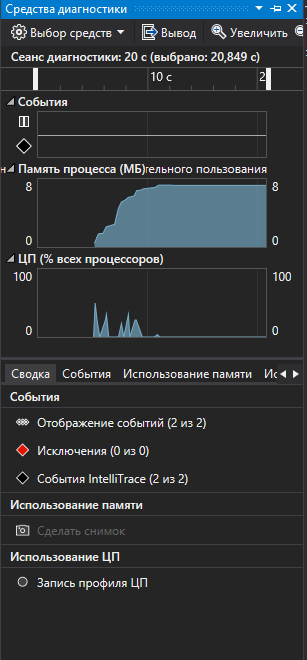


Рис 7

8) Нажмите клавишу F5, чтобы запустить приложение до второй точки останова.

Теперь у вас есть данные о производительности приложения именно для той области кода, которая выполняется между двумя точками останова.

Профилировщик начинает подготавливать данные потока. Дождитесь завершения этой операции

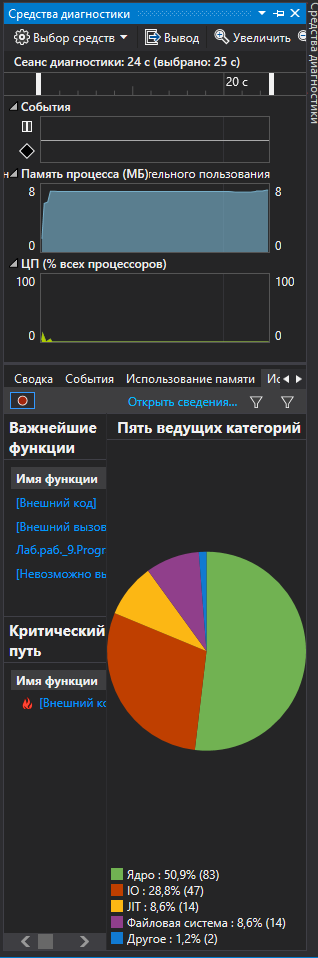


Рис 8

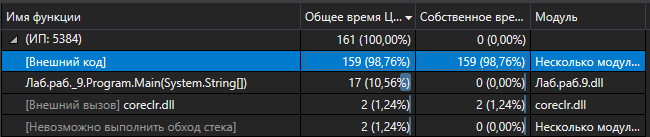


Рис 9

9)Если вы хотите выбрать для анализа более конкретную область кода, выберите область на временной шкале ЦП (это должна быть область, в которой отображаются данные профилирования

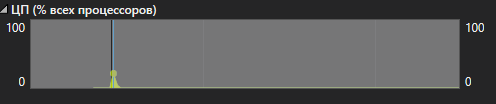


Рис 10

**Шаг 2. Анализ данных о загрузке ЦП**

1)В списке функций изучите функции, которые выполняют большую часть работы

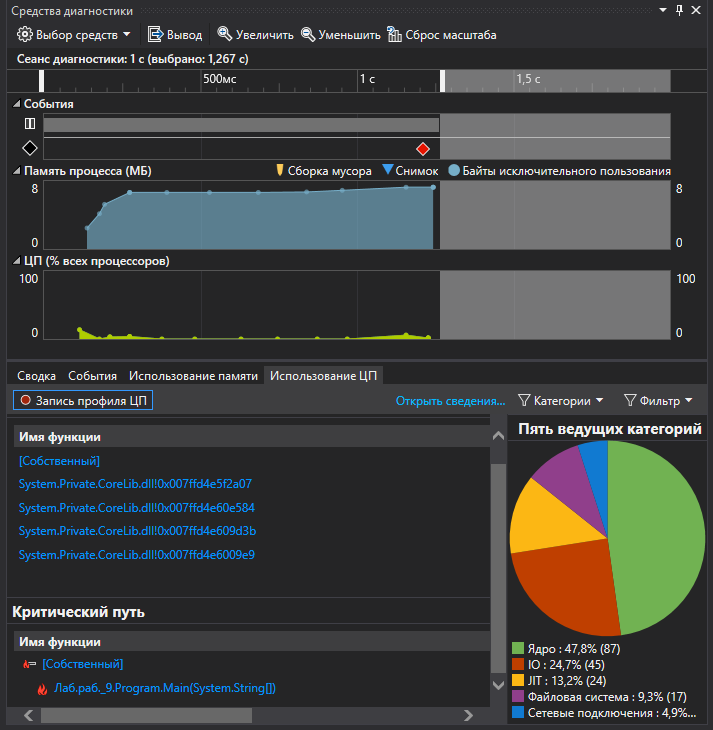


Рис 11

2) В списке функций дважды щелкните одну из функций вашего приложения, которая выполняет много работы.

При двойном щелчке функции в левой панели откроется представление **Вызывающий/вызываемый**

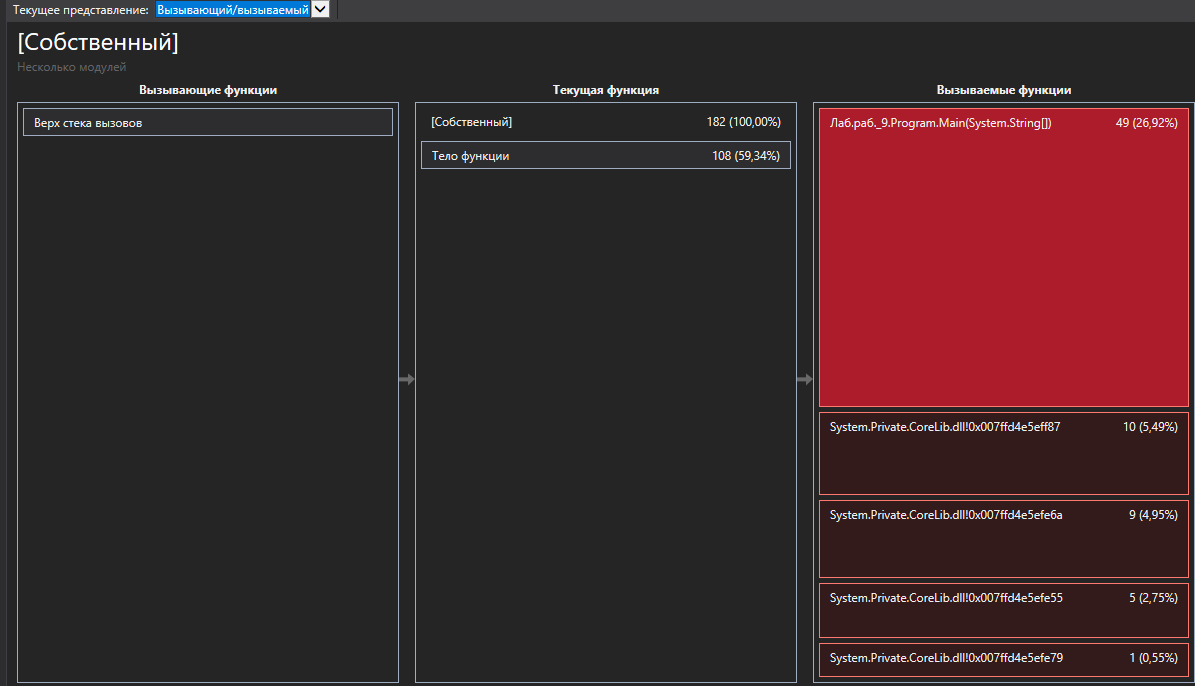
****

Рис 12

**3)** Чтобы увидеть более обобщенное представление, показывающее порядок, в котором вызываются функции, выберите в раскрывающемся списке в верхней части панели пункт **Дерево вызовов**.

Каждая нумерованная область на рисунке соответствует определенному шагу в процедуре.

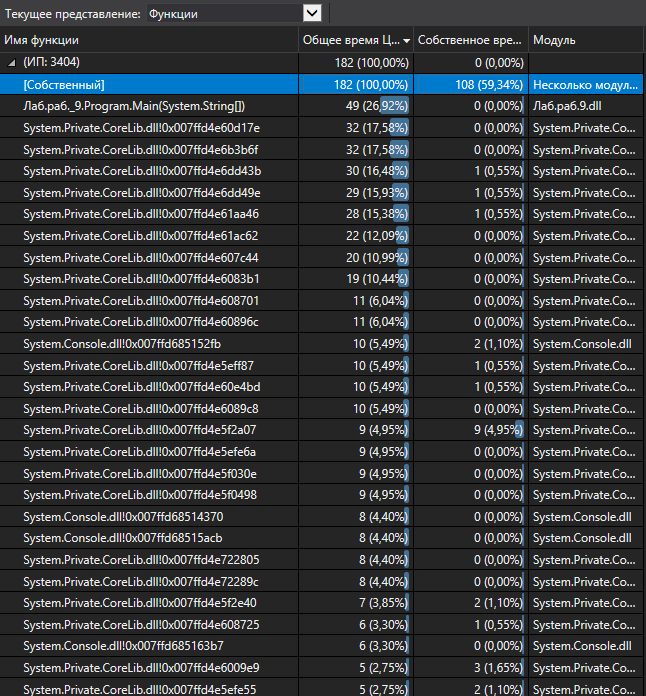


Рис 13

**ТАБЛИЦА 1**

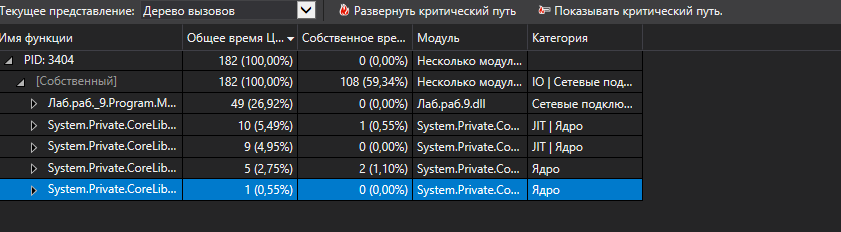


Рис 14

# Измерение производительности приложения посредством анализа использования ЦП

Просмотр внешнего кода

Внешний код — это функции в компонентах системы и платформы, которые исполняются вашим кодом. Внешний код включает функции, которые запускают и останавливают приложение, отрисовывают пользовательский интерфейс, управляют потоками и предоставляют приложению другие низкоуровневые службы. В большинстве случаев внешний код вас интересовать не будет, поэтому средство "Загрузка ЦП" собирает внешние функции пользовательского метода в один узел **[Внешний код]** .

Если вы захотите посмотреть пути к вызовам внешнего кода, выберите **Показать внешний код** в списке **Представление фильтра** и выберите **Применить**.

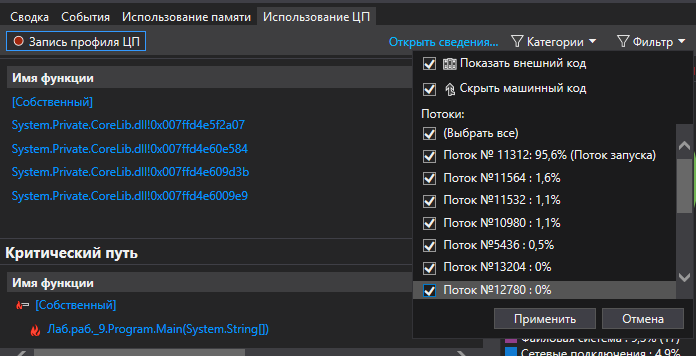


Рис 15