**Отчёт по практическому заданию №1**

1. Открыли проект и установили точки остановы, где хотим проверить загрузку ЦП.

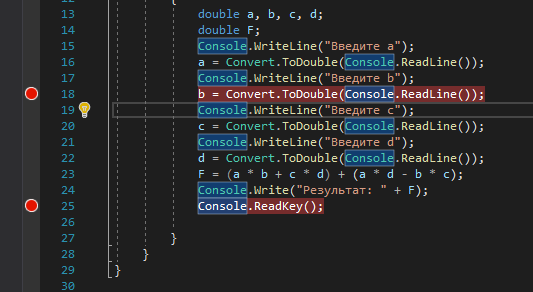


Рис. 1

2. Ставим вторую точку остановы в конце функции

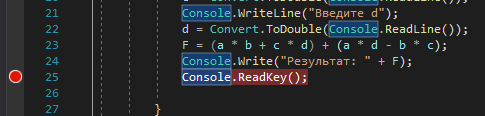


Рис. 2

3. Открыли окно **Отладка>Окна>Показать средства диагностики**

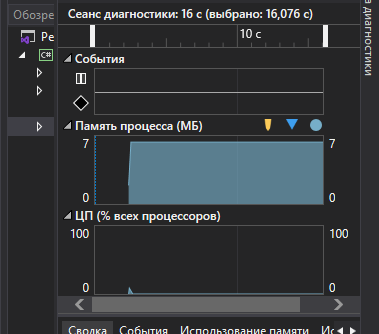


Рис. 3

4. Вы можете выбрать, что следует просмотреть, [Использование памяти](https://docs.microsoft.com/ru-ru/visualstudio/profiling/memory-usage?view=vs-2022) 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**.

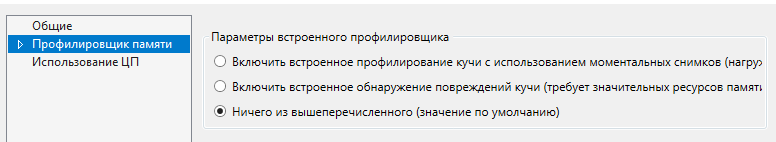


Рис. 4

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



Рис. 5

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

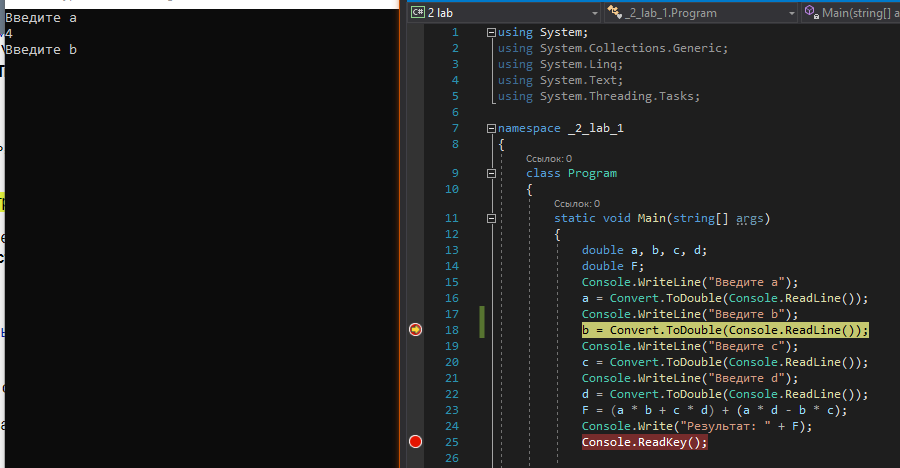


Рис. 6

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

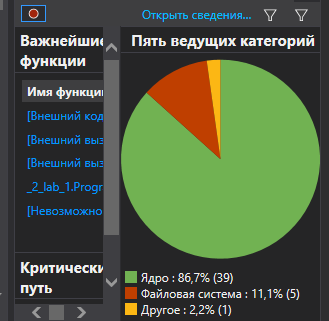


Рис. 7

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.

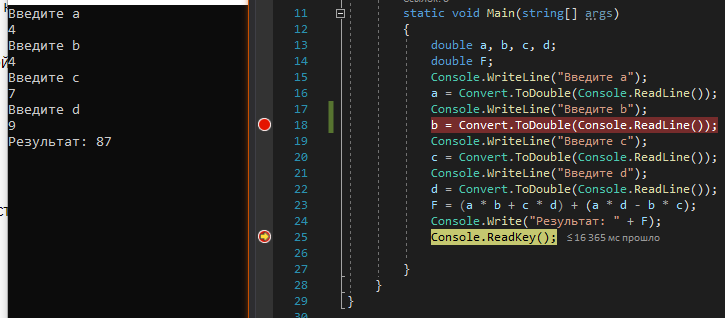


Рис. 8

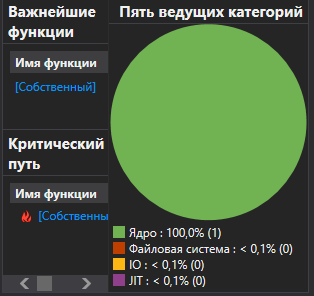


Рис. 9

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).

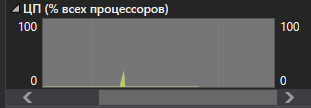


Рис. 10

### Шаг 2 Анализ данных о загрузке ЦП

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

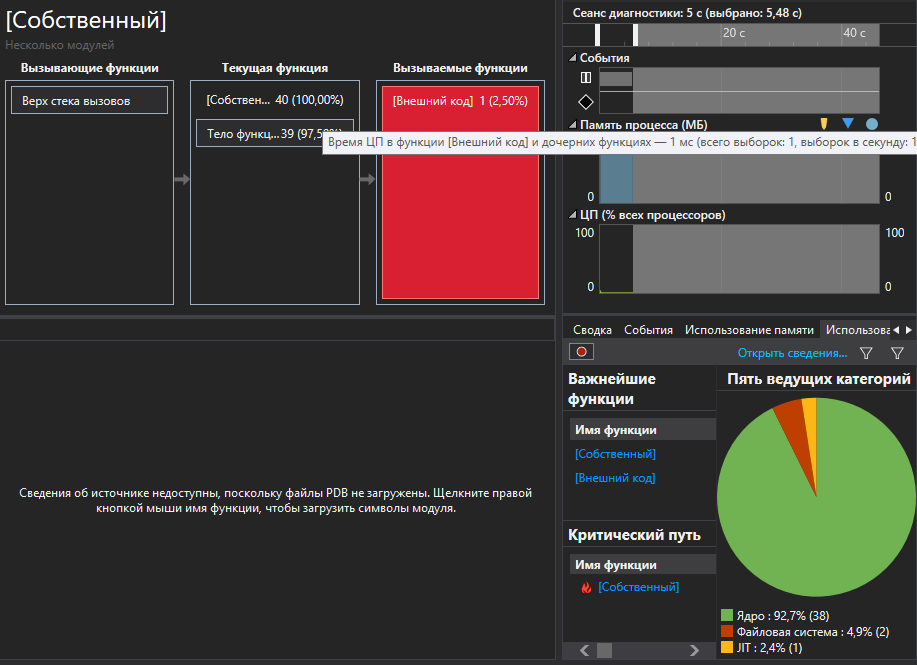


Рис. 11

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**.

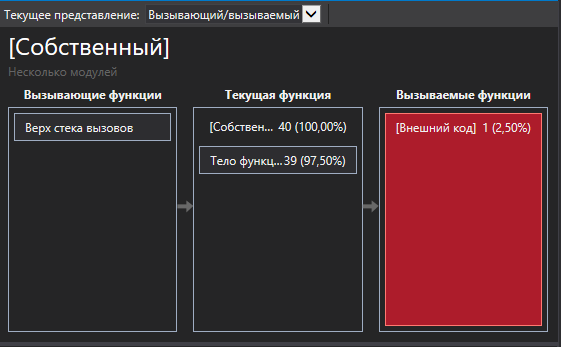


Рис. 12

В этом представлении выбранная функции отображается в заголовке и в поле **Текущая функция** (в этом примере GetNumber). Функция, вызывавшая текущую функцию, отображается в левой части окна в разделе **Вызывающие функции**, а все функции, вызываемые текущей функцией, отображаются в поле **Вызываемые функции** справа. (Можно выбрать любое поле, чтобы изменить текущую функцию.)

В этом представлении показано общее время (мс) и доля общего времени выполнения приложения, затраченного на выполнение функции. В поле **Тело функции** также показан общий объем времени (и доля времени), затраченного в теле функции за исключением времени, затраченного в вызываемых и вызывающих функциях. (В этом примере в теле функции затрачено 39 из 40 мс, а оставшиеся 1 мс затрачены во внешнем коде, вызванном этой функцией).

2. 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.

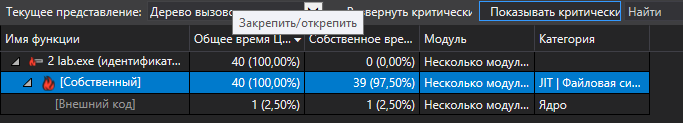


Рис. 13

Чтобы увидеть вызовы функций, которые используют самый высокий процент ЦП в представлении дерева вызовов, нажмите **Развернуть критический путь**.



Рис. 14

**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**.

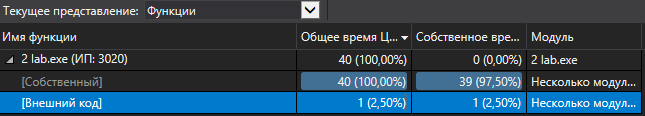


Рис. 15

https://github.com/viks17/praktika.git