**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ**

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ**

**«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук

Образовательная программа бакалавриата «Программная инженерия»

|  |  |  |
| --- | --- | --- |
| СОГЛАСОВАНО  Научный руководитель,  преподаватель департамента программной инженерии ФКН, кандидат компьютерных наук    \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Р.А.Нестеров  \_\_\_\_\_\_\_\_\_\_\_\_\_11 \_мая\_\_ 2023 г. |  | УТВЕРЖДАЮ  Академический руководитель образовательной программы «Программная инженерия»  профессор департамента программной инженерии, кандидат технических наук    \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.В. Шилов  \_\_\_\_\_\_\_\_\_\_\_\_11\_мая\_\_\_2023 г. |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | ***Подп. и дата*** |  | | ***Инв. № дубл.*** |  | | ***Взам. Инв. №*** |  | | ***Подп. и дата*** |  | | ***Инв. № подл.*** |  | | **ВСТРАИВАЕМЫЙ ПРОФИЛИРОВЩИК**  **ПРОГРАММНОГО КОДА НА ЯЗЫКЕ С++**  **Руководство оператора**  **ЛИСТ УТВЕРЖДЕНИЯ**  **RU.17701729.04.04-01 34 01-1-ЛУ**  Исполнитель  студент группы БПИ214  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ Е.К..Фортов/  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_11\_мая\_\_2023 г. | |
|  |  |

УТВЕРЖДЕН

RU.17701729.04.04-01 34 01-1-ЛУ

|  |  |
| --- | --- |
| ***Подп. и дата*** |  |
| ***Инв. № дубл.*** |  |
| ***Взам. инв. №*** |  |
| ***Подп. и дата*** |  |
| ***Инв. № подл*** |  |

**ВСТРАИВАЕМЫЙ ПРОФИЛИРОВЩИК**

**ПРОГРАММНОГО КОДА НА ЯЗЫКЕ С++**

**Руководство оператора  
  
RU.17701729.04.04-01 34 01-1-ЛУ**

**Листов 17**

Оглавление

[1. НАЗНАЧЕНИЕ ПРОГРАММЫ 3](#_Toc72273819)

[1.1. Функциональное назначение программы 3](#_Toc72273820)

[1.2. Эксплуатационное назначение 3](#_Toc72273821)

[1.3. Состав функций](#_Toc72273822) 4

[2. УСЛОВИЯ ВЫПОЛНЕНИЯ ПРОГРАММЫ](#_Toc72273823) 5

[2.1. Минимальный состав аппаратурных средств 6](#_Toc72273824)

[2.2. Минимальный состав программных средств 6](#_Toc72273825)

[2.3. Требования к персоналу (пользователю) 6](#_Toc72273826)

[3. ВЫПОЛНЕНИЕ ПРОГРАММЫ 7](#_Toc72273827)

[3.1. Загрузка программы 7](#_Toc72273828)

[3.2. Запуск программы 7](#_Toc72273829)

[3.3. Выполнение программы](#_Toc72273830) 7

[3.3.1. Получение справки по командам до начала пользования](#_Toc72273831) 7

[3.3.2. Начало замера времени](#_Toc72273832) 7

[3.3.3. Конец замера времени](#_Toc72273833) 8

[3.3.4. Вывод результата последнего замера времени (вручную)](#_Toc72273834) 9

[3.3.5. Получение информации о профайлере 1](#_Toc72273835)0

[3.3.6. Вывод локальных логов в консоль 1](#_Toc72273836)1

[3.3.7. Очистка логов 1](#_Toc72273837)2

[3.3.8. Изменение размерности логов](#_Toc72273838) 13

[3.4. Завершение программы 1](#_Toc72273851)4

[4. СООБЩЕНИЯ ОПЕРАТОРУ](#_Toc72273852) 15

[СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ](#_Toc72273853) 16

# 1. НАЗНАЧЕНИЕ ПРОГРАММЫ

## 1.1. Функциональное назначение программы

Функциональным назначением программы является предоставление программисту возможности быстро и удобно проводить временные замеры частей своей программы, написанной на С++, во время выполнения программы. Кроме того, разрабатываемая программа создает файлы с результатами проведенных измерений для дальнейшего построения графиков в сторонней программе. Эти данные можно использовать для построения статистики работы исследуемой программы.

## 1.2. Эксплуатационное назначение

При написании программы по технической документации разработчик должен оценивать время исполнения программного кода на разных входных данных. Для этого программисту иногда удобно использовать готовый профилировщик кода - программу, которая упрощает проведение замеров времени исполнения отдельных частей кода. К сожалению, использование существующих профилировщиков может приводить к затруднениям по причине довольно длительной настройки, а также внедрения большого числа дополнительных инструкций и команд.

Более того, большинство из них предлагают только графический интерфейс, что делает невозможным тестирование программного кода в консоли, например, если программа тестируется на удаленном сервере с доступом только по протоколу ssh. «FAST\_PROFILE» решает эти проблемы следующим образом:

1. Подключается одной командой через заголовочный файл
2. Для начала/окончания замеров времени, а также для других функций, таких как построение графиков проведенных замеров и вывод в консоль/файлы логов, используются макросы.

Встраиваемый профилировщик кода упрощает проведение временных тестов программного кода. В результате экономится время как разработчика, так и тестировщика.

## 1.3. Состав функций

Программа даёт пользователю возможность выполнять следующие функции:

* начинать замер времени;
* заканчивать замер времени;
* выводить результаты замеров в консоли;
* выводить результаты замеров в отдельный файл (логировать в .txt, .json, .csv);
* создавать файлы с результатами проведенных измерений (на некоторых различных входных данных, код программы остается таким же) для дальнейшего построения графиков на основе этих данных в форматах .csv, .json;
* считать и выводить базовую аналитику результатов (выделение наибольшего и наименьшего времени исполнения того или иного участка кода);
* подсвеченный синтаксис результатов, чтобы отличить вывод программы от вывода логов профайлера
* указывать точность замеров времени (вывод в секундах, миллисекундах, микросекундах, наносекундах);
* добавлять комментарии к i-ому замеру времени при вызове той или иной функции профилировщика;
* выводить справку по командам;
* дополнять функциональность путем наследования главного класса профайлера;
* подключать весь профилировщик путем добавления одного файла с помощью директивы include;

# 2. УСЛОВИЯ ВЫПОЛНЕНИЯ ПРОГРАММЫ

## 2.1. Минимальный состав аппаратурных средств

Для бесперебойной работы программного продукта требуется компьютер с:

1. Объемом свободной встроенной памяти не меньше 55 МБ,
2. Объёмом оперативной памяти не меньше 1 ГБ.

## 2.2. Минимальный состав программных средств

Для бесперебойной работы программного продукта требуется компьютер с:

1. Установленной версией компилятора gcc - 14, clang — 3.4
2. Операционной системой со стабильной сборкой, выпущенной не позднее 2015 года

## 2.3. Требования к персоналу (пользователю)

Необходимое количество персонала – 1 человек.

Необходимая квалификация персонала – пользователь.

# 3. ВЫПОЛНЕНИЕ ПРОГРАММЫ

В данном разделе описан пример работы с программой.

## 3.1. Загрузка программы

Для загрузки данного ПО надо скачать файл profiler.hpp и папку include и положить эти файлы в ту же директорию, где лежит код, который будет профилирован.

## 3.2. Запуск программы

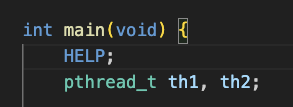
Для подключения профайлера к основной программе необходимо прописать #include «profiler.hpp» (см. рис. 1):

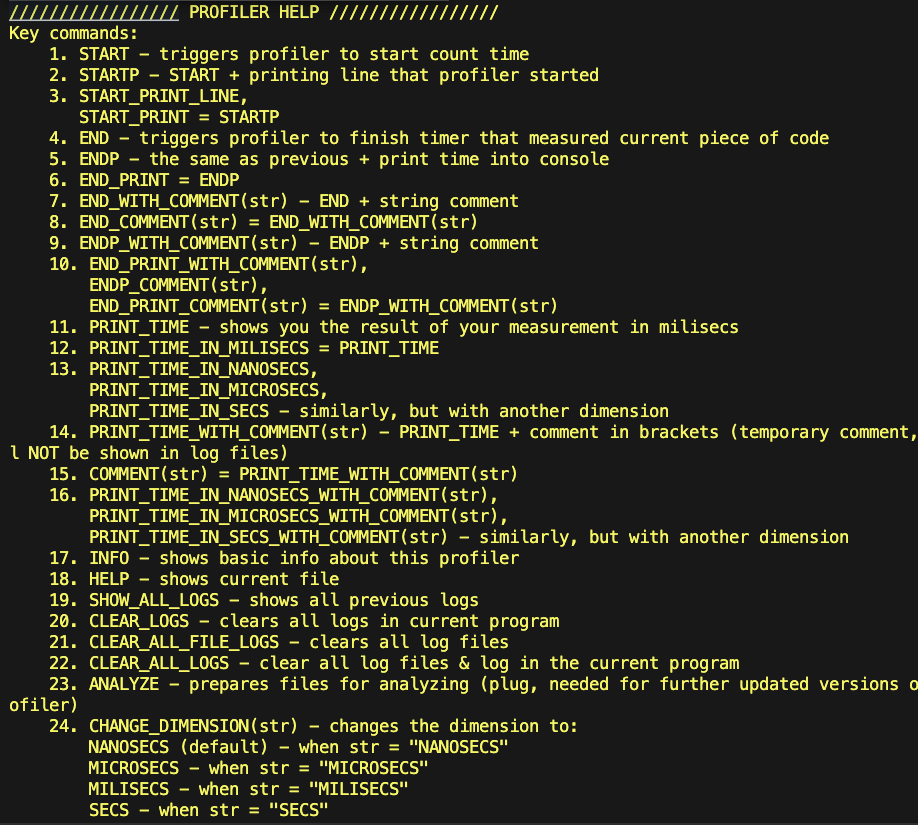


## 3.3. Выполнение программы

### 3.3.1. Получение справки по командам до начала пользования

Настоятельно рекомендую ознакомиться с основными командами профайлера, чтобы избежать в дальнейшем ошибок в замерах времени. Для получения справки по командам надо написать в коде (как можно раньше) HELP:

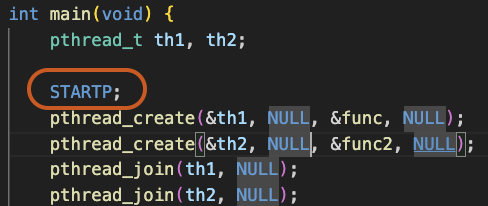




### 3.3.2. Начало замера времени

Теперь необходимо определить часть кода, которая будет тестироваться на предмет времени исполнения. Перед этим куском кода надо поставить один из макросов:

* START — начало замера времени
* STARTP — начало замера времени + вывод на экран комментария об этом
* START\_PRINT\_LINE = STARTP
* START\_PRINT = STARTP



### 

### 3.3.3. Конец замера времени

### После исследуемого участка кода необходимо поставить макрос END (конец замера времени + подсчет результата + логирование в локальные поля). При использовании команд ENDP / END\_PRINT поведение профайлера будет аналогичным команде END, только дополнительно текущий результат будет выведен в консоль (также желтым цветом). Также можно остановить замер времени с комментарием — он будет отображен в логах. Это делается с помощью команд END\_WITH\_COMMENT(str) / END\_COMMENT(str), где str — комментарий. Если требуется и оставить комментарий к замеру времени, и вывести результат на экран, следует воспользоваться одной из команд:

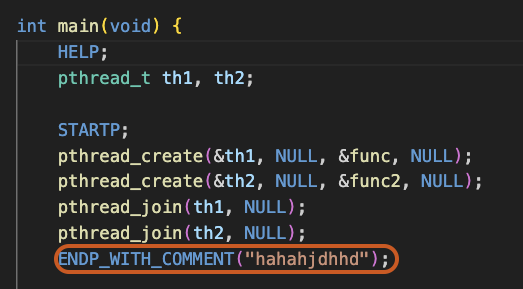
### ENDP\_WITH\_COMMENT(str)

### END\_PRINT\_WITH\_COMMENT(str)

### ENDP\_COMMENT(str)

### END\_PRINT\_COMMENT(str).

### Алгоритм работы данных команд аналогичен командам выше.

****

### 

### 3.3.4. Вывод результата последнего замера времени (вручную)

### Для (дополнительной) распечатки замеренного результата можно воспользоваться любой из команд:

### PRINT\_TIME

### PRINT\_TIME\_IN\_NANOSECS

### PRINT\_TIME\_IN\_MICROSECS

### PRINT\_TIME\_IN\_SECS

### PRINT\_TIME\_IN\_MILISECS

### Эти команды возьмут результат последнего замера времени и выведут результат в нужной размерности на экран желтым цветом (вместе с комментарием). По умолчанию используются милисекунды (команда PRINT\_TIME). Для вывода результата с комментарием используются функции:

### PRINT\_TIME\_IN\_MILISECS

### PRINT\_TIME\_WITH\_COMMENT(str)

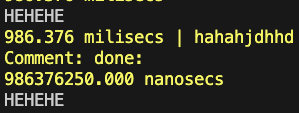
### PRINT\_TIME\_IN\_NANOSECS\_WITH\_COMMENT(str)

### PRINT\_TIME\_IN\_MICROSECS\_WITH\_COMMENT(str)

### PRINT\_TIME\_IN\_SECS\_WITH\_COMMENT(str)

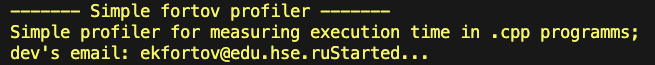
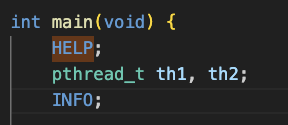
### COMMENT(str), где str — комментарией (локальный, он будет отображен только в консоли и не будет сохраняться в логах. Для сохранения комментария в логи должна быть использована команда END\_WITH\_COMMENT(str), описанная выше). Алгоритм работы этих команд аналогичен командам, описанным выше.





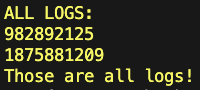
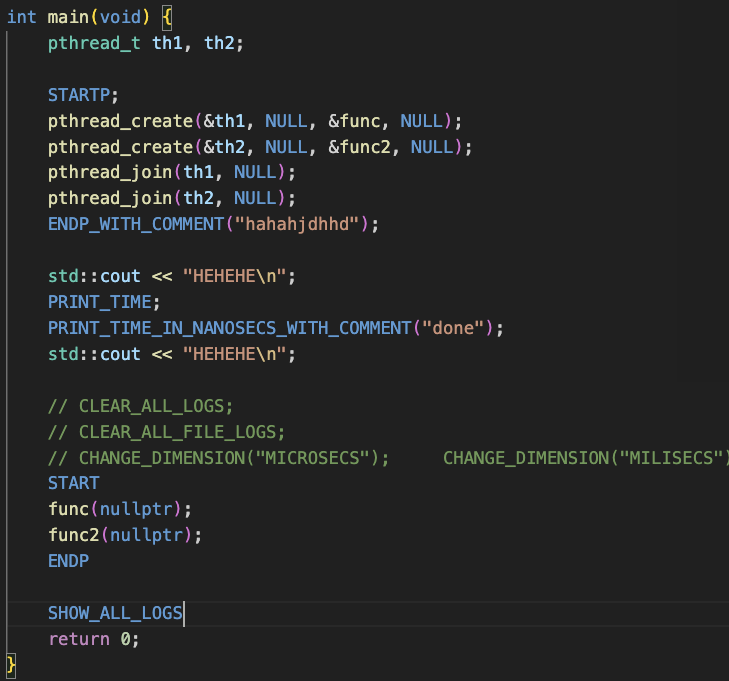
### 3.3.5. Получение информации о профайлере

### Для получения информации о данном ПО пользователь может воспользоваться макросом INFO. Этот макрос выведет литералы, описывающие что есть данный продукт, зачем он нужен, базовые команды, а также контакты его разработчика.



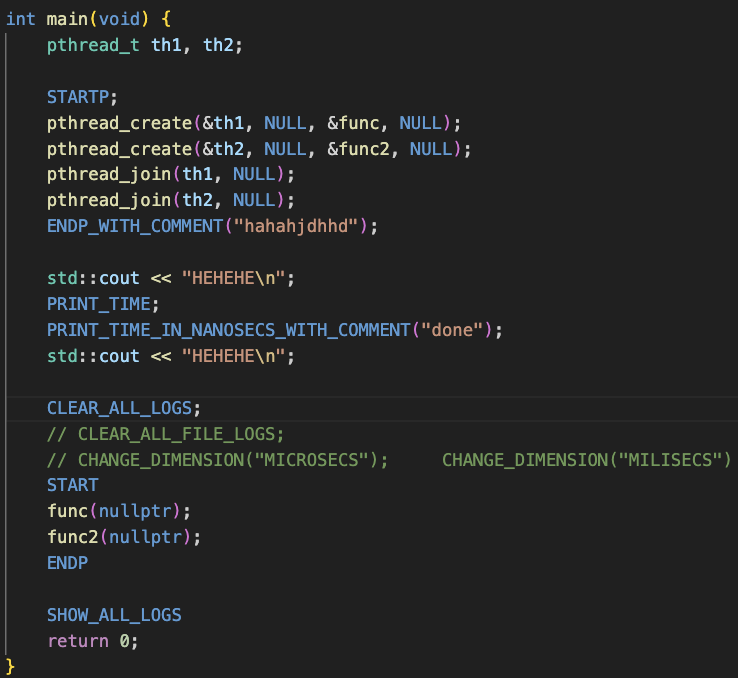
### 3.3.6. Вывод локальных логов в консоль

### Для вывода всех имеющихся в данной сессии логов можно воспользоваться макросом SHOW\_ALL\_LOGS, который выведет все логи на экран в том порядке, в котором пользователь совершал замеры времени.



### 3.3.7. Очистка логов

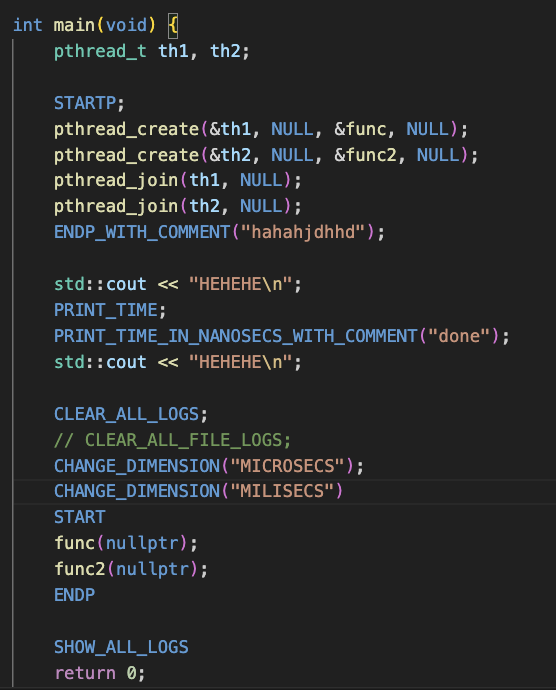
### Для очистки логов можно воспользоваться макросом CLEAR\_LOGS, который удалит все логи в текущей сессии (логи предыдущих сессий останутся в файлах). Для удаления всех файлов логов есть макрос CLEAR\_ALL\_FILE\_LOGS, который удаляет файлы log.txt, log.json, log.csv. Для удаления как текущих логов (логов текущей сессии работы программы), так и файлов логов, используется макрос CLEAR\_ALL\_LOGS, который делает то же самое, что и макросы CLEAR\_LOGS и CLEAR\_ALL\_FILE\_LOGS вместе.



### 3.3.8. Изменение размерности логов

### Для изменения размерности результатов замеров используется макрос CHANGE\_DIMENSION(str), где str — нужная размерность логов: "NANOSECS" (значение по умолчанию), "MICROSECS", "MILISECS", "SECS". Замечание: данная команда работает только для внешних логов — в файлах. Причем размерность замеров времени в файлах будет соответствовать аргументу последнего вызова данной команды. В локальных логах (вывод в консоль) для изменения стандартной размерности (милисекунды) следует пользоваться командами:умолчанию используются милисекунды (команда PRINT\_TIME). Для вывода результата с комментарием используются функции:

* PRINT\_TIME\_IN\_NANOSECS
* PRINT\_TIME\_IN\_MICROSECS
* PRINT\_TIME\_IN\_SECS
* PRINT\_TIME\_IN\_MILISECS (описаны выше).



## 

## 

## 3.4. Завершение программы

Профилировщик сам завершит свою работу вместе с завершением пользовательской программы.

# 4. СООБЩЕНИЯ ОПЕРАТОРУ

В программе допускается Undefined Behaviour, которое может возникать по причине некорректного пользования профайлером.

Например, UB может возникнуть при вводе некорректного комментария в случае смены размерности результатов.

**СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ**

1. CSV VS JSON [Электронный ресурс]. Режим доступа: https://coresignal.com/blog/json-vs-csv/, свободный. (дата обращения: 11.05.2023)

# ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Изм. | Номера листов (страниц) | | | | Всего листов (страниц) в документе | № документа | Входящий № сопроводительного документа и дата | Подпись | Дата |
| измененных | замененных | новых | аннулированных |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |