


**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ  
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ  
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук  
Образовательная программа бакалавриата «Программная инженерия»


**СОГЛАСОВАНО**

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.


**ВСТРАИВАЕМЫЙ ПРОФИЛИРОВЩИК  
ПРОГРАММНОГО КОДА НА ЯЗЫКЕ C++**

Техническое задание

**ЛИСТ УТВЕРЖДЕНИЯ**

RU.17701729.04.04-01 ТЗ 01-1-ЛУ

Исполнитель  
студент группы БПИ214

—  — / Е.К.Фортов/  
— 11 мая 2023 г.

Подп. и дата	
Инв. № дубл.	
Взам. Инв. №	
Подп. и дата	
Инв. № подл.	

УТВЕРЖДЕН  
RU.17701729.04.04-01 ТЗ 01-1-ЛУ

**ВСТРАИВАЕМЫЙ ПРОФИЛИРОВЩИК ПРОГРАММНОГО КОДА НА ЯЗЫКЕ C++**

**Техническое задание**

**RU.17701729.04.04-01 ТЗ 01-1-ЛУ**

**Листов 20**

<i>Подп. и дата</i>	
<i>Инв. № дубл.</i>	
<i>Взам. инв. №</i>	
<i>Подп. и дата</i>	
<i>Инв. № подл</i>	

## Оглавление

1. ВВЕДЕНИЕ.....	3
1.1. Наименование программы.....	3
1.2. Краткая характеристика области применения.....	3
2. ОСНОВАНИЕ ДЛЯ РАЗРАБОТКИ.....	4
2.1. Документы, на основании которых ведётся разработка.....	4
2.2. Наименование темы разработки.....	4
3. НАЗНАЧЕНИЕ РАЗРАБОТКИ.....	5
3.1. Функциональное назначение.....	5
3.2. Эксплуатационное назначение.....	5
4. ТРЕБОВАНИЯ К ПРОГРАММЕ.....	6
4.1. Требования к функциональным характеристикам.....	6
4.1.1. Требования к составу выполняемых функций.....	6
4.1.2. Требования к организации входных данных.....	7
4.1.3. Требования к организации выходных данных.....	7
4.2. Требования к интерфейсу.....	8
4.3. Требования к надежности.....	8
4.3.1. Требования к обеспечению надежного (устойчивого) функционирования программы.....	8
4.3.2. Время восстановления после отказа.....	8
4.3.3. Отказы из-за некорректных действий оператора.....	9
4.4. Условия эксплуатации.....	10
4.5. Требования к составу и параметрам технических средств.....	10
4.6. Требования к информационной и программной совместимости.....	10
4.6.1. Требования к информационным структурам и методам решения.....	10
4.6.2. Требования к программным средствам, используемым программой.....	10
4.6.3. Требования к исходным кодам и языкам программирования.....	10
4.7. Требования к маркировке и упаковке.....	10
4.8. Требования к транспортировке и хранению.....	11
4.8.1. Требования к транспортировке и хранению программных документов, предоставленных в электронном виде.....	11
4.8.2. Требования к транспортировке и хранению программных документов, представленных в печатном виде.....	11
5. ТРЕБОВАНИЯ К ПРОГРАММНОЙ ДОКУМЕНТАЦИИ.....	12
5.1. Предварительный состав программной документации.....	12
5.2. Специальные требования к программной документации.....	12
6. ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ.....	13
6.1. Ориентировочная экономическая эффективность.....	13
6.2. Предполагаемая потребность.....	13
6.3. Экономические преимущества разработки по сравнению с отечественными или зарубежными аналогами.....	13
7. СТАДИИ И ЭТАПЫ РАЗРАБОТКИ.....	15
7.1. Необходимые стадии разработки, этапы и содержание работ.....	15
7.2. Сроки разработки и исполнители.....	16
8. ПОРЯДОК КОНТРОЛЯ И ПРИЁМКИ.....	17
СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ.....	18

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.04-01 ТЗ				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

## 1. ВВЕДЕНИЕ

### 1.1. Наименование программы

Наименование программы – «Встраиваемый Профилировщик Программного Кода на Языке C++ «FAST\_PROFILE»» («Embedded Profiler of C++ Program Code «FAST\_PROFILE»»).

### 1.2. Краткая характеристика области применения

«FAST\_PROFILE» дает возможность даже начинающим программистам быстро, точно и удобно проводить замеры времени исполнения кода на C++, что избавляет от необходимости писать свои конструкции для данной цели, а также от необходимости скачивать отдельное приложение с GUI и разбираться в том, как правильно использовать ту или иную функцию, читая объемную документацию профилировщика.

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.04-01 ТЗ				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

## **2. ОСНОВАНИЕ ДЛЯ РАЗРАБОТКИ**

### **2.1. Документы, на основании которых ведётся разработка**

Основанием для разработки является учебный план подготовки бакалавров по направлению 09.03.04 «Программная инженерия» и утвержденная академическим руководителем тема курсового проекта».

### **2.2. Наименование темы разработки**

Наименование темы разработки – «Встраиваемый Профилировщик Программного Кода на Языке C++ «FAST\_PROFILE»».

Программа выполняется в рамках темы курсового проекта — «Встраиваемый Профилировщик Программного Кода на Языке C++», в соответствии с учебным планом подготовки бакалавров по направлению 09.03.04 «Программная инженерия».

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.04-01 ТЗ				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

### 3. НАЗНАЧЕНИЕ РАЗРАБОТКИ

#### 3.1. Функциональное назначение

Функциональным назначением программы является предоставление программисту возможности быстро и удобно проводить временные замеры частей своей программы, написанной на C++, 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.

#### 3.2. 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.

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.04-01 ТЗ				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

## 4. ТРЕБОВАНИЯ К ПРОГРАММЕ

### 4.1. Требования к функциональным характеристикам

#### 4.1.1. Требования к составу выполняемых функций

Программа должна давать 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;
- удалять логи (как локально, так и глобально)

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.04-01 ТЗ				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

#### 4.1.2. Требования к организации входных данных

Входные данные — это исходный код, в который встраиваются нужные команды профилировщика (исходный код может быть представлен в файлах с расширениями .cpp & .h). Исходный код должен отрабатывать корректно, без предупреждений компилятора и ошибок.

В свою очередь, команды профилировщика, внедряемые в код тестируемой программы, — это макросы или отдельные функции, которые могут принимать дополнительную информацию (например, комментарии к тому или иному замеру времени) в виде строк и целочисленных значений.

Касательно требований к входным данным, пользователю необходимо ознакомиться с внутренней справкой профайлера, которая описывает, какие входные данные принимает та или иная функция.

#### 4.1.3. Требования к организации выходных данных

Результат работы профайлера на тестируемом коде должен содержать следующее (цвет вывода — по умолчанию стандартный, если не был указан другой в основном коде) :

- вывод результатов проведенных замеров (вывод в консоль + .csv & .json файлы для дальнейшей постройки графиков в сторонних сервисах)
- вывод базовой аналитики (максимальное/минимальное время тестов)
- вывод во внешний .txt файл отчета о проведенном тестировании (комментарии)
- вывод справки по командам (при вызове соответствующей команды профилировщика)

Если программа, на которой проводились замеры времени, отработала с некорректным завершением (код возврата не 0), профайлер может ничего не вывести и не сохранить проведенные исследования во внешний файл.

#### 4.2. Требования к интерфейсу

Графический интерфейс у данного сервиса фактически отсутствует, так как все команды прописываются именно в исходном файле.

Справка по командам профайлера содержится в прилагаемом .txt файле, также возможен вызов справки из исходного кода в консоль с помощью соответствующей команды в исходном файле.

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.04-01 ТЗ				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата



### 4.3. Требования к надежности

#### 4.3.1. Требования к обеспечению надежного (устойчивого) функционирования программы

Для устойчивой работы программы необходимо соблюдать ряд организационно-технических мер:

- 1) Убедиться, что написанная программа будет отрабатывать корректно на входных данных, на которых будут проводиться замеры времени исполнения частей кода. Это значит, что при компиляции тестируемой программы необходимо включить следующие флаги: -fsanitize=address,undefined -fno-sanitize-recover=all -Wall -Wextra -Werror -std=c++14 -pedantic; Компилировать необходимо компилятором gcc версии не ниже 14 или компилятором clang версии не ниже 3.4;
- 2) Иметь правильно скомпилированные и находящиеся в нужном месте библиотеки, который использует данный профайлер;
- 3) Компиляция исходного кода должна производиться с флагами оптимизации (-O2, -O3 или -Ofast);

#### 4.3.2. Время восстановления после отказа

Если отказ был спровоцирован внешними факторами (например, поломка энергоблока компьютера или неисправность других его внутренних компонентов), то время исправления ситуации не регламентируется.

Если отказ был спровоцирован внутренними факторами (например, пользователь случайно удалил системный файл и ОС теперь работает некорректно), то время восстановления не должно быть больше времени, необходимого для исправления ошибки с ОС.

#### 4.3.3. Отказы из-за некорректных действий оператора

Отказ программы возможен также вследствие некорректных действий пользователя при неправильном использовании (например, исходный тестируемый код отрабатывает с ошибкой или предупреждением или в runtime возникло неопределенное поведение). Чтобы такого не допускать, необходимо ознакомиться с пунктом 4.3.1;

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.04-01 ТЗ				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

Также отказ возможен при некорректном пользовании операционной системой. В таком случае время на восстановления профайлера не должно превышать времени, необходимого для устранения поломки ОС.

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.04-01 ТЗ				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

#### 4.4. Условия эксплуатации

Компьютер предназначен для эксплуатации в помещениях с искусственно-регулируемыми климатическими условиями, например, в отапливаемых и вентилируемых помещениях категории 4.1 согласно ГОСТ 15150-69 [4].

Программа не требует специального обслуживания.

Программа предназначена для пользования одним человеком. Необходимая квалификация – пользователь (ознакомившийся с краткой справкой профайлера).

#### 4.5. Требования к составу и параметрам технических средств

Для бесперебойной работы программного продукта требуется компьютер с:

- установленной версией компилятора gcc - 14, clang — 3.4
- операционной системой со стабильной сборкой, выпущенной не позднее 2015 года
- объемом свободной встроенной памяти не меньше 55 МБ,
- объёмом оперативной памяти не меньше 1 ГБ.

#### 4.6. Требования к информационной и программной совместимости

##### 4.6.1. Требования к информационным структурам и методам решения

Требования к информационным структурам и методам решения не предъявляются.

##### 4.6.2. Требования к программным средствам, используемым программой

Для работы программного продукта требуется gcc компилятор версии не ниже 14 или clang компилятор версии не ниже 3.4; необходимые флаги см. в пункте 4.3.1

##### 4.6.3. Требования к исходным кодам и языкам программирования

Программа должна быть написана на языке программирования C++ версии не выше 14. В качестве среды разработки программы может быть использован любой редактор кода. Допускается писать код только в .cpp и .h файлах.

#### 4.7. Требования к маркировке и упаковке

Требования к маркировке и упаковке не предъявляются.

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.04-01 ТЗ				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

#### **4.8. Требования к транспортировке и хранению**

##### **4.8.1. Требования к транспортировке и хранению программных документов, предоставленных в электронном виде**

Программные документы загружаются в электронном виде в информационную образовательную среду LMS (Learning Management System) НИУ ВШЭ. Требования к хранению и транспортировке не предъявляются.

##### **4.8.2. Требования к транспортировке и хранению программных документов, представленных в печатном виде**

Программные документы, предоставляемые в печатном виде, должны соответствовать общим правилам учета и хранения программных документов, предусмотренных стандартами Единой системы программной документации и соответствовать требованиям ГОСТ 19.602-78 [13].

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.04-01 ТЗ				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

## **5. ТРЕБОВАНИЯ К ПРОГРАММНОЙ ДОКУМЕНТАЦИИ**

### **5.1. Предварительный состав программной документации**

- 1) «Встраиваемый Профилировщик Программного Кода на Языке C++». Техническое задание (ГОСТ 19.201-78 [8])
- 2) «Встраиваемый Профилировщик Программного Кода на Языке C++». Программа и методика испытаний (ГОСТ 19.301-79 [9])
- 3) «Встраиваемый Профилировщик Программного Кода на Языке C++». Текст программы (ГОСТ 19.401-78 [10])
- 4) «Встраиваемый Профилировщик Программного Кода на Языке C++». Пояснительная записка (ГОСТ 19.404-79 [11])
- 5) «Встраиваемый Профилировщик Программного Кода на Языке C++». Руководство оператора (ГОСТ 19.505-79 [12])

### **5.2. Специальные требования к программной документации**

- 1) Все документы к программе должны быть выполнены в соответствии с ГОСТ 19.106-78 [7] и ГОСТами к каждому виду документа (см. п. 5.1.);
- 2) Пояснительная записка должна быть загружена в систему Антиплагиат через LMS (Learning Management System) НИУ ВШЭ.
- 3) Вся документация и программа также сдаются в электронном виде в формате .pdf или .docx. в архиве формата .rar или .zip.
- 4) За три дня до защиты комиссии все материалы курсового проекта:
  - техническая документация,
  - программный проект,
  - исполняемый файл,
  - отзыв руководителя,
  - лист Антиплагиата

должны быть загружены одним или несколькими архивами в проект дисциплины «Курсовой проект, 2 курс ПИ» в личном кабинете в информационной образовательной среде LMS (Learning Management System) НИУ ВШЭ.

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.04-01 ТЗ				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

## 6. ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ

### 6.1. Ориентировочная экономическая эффективность

В рамках данной работы расчет экономической эффективности не предусмотрен.

### 6.2. Предполагаемая потребность

Данный профилировщик могут использовать все разработчики/тестировщики с компилятором gcc версии не ниже 14 или компилятором clang версии не ниже 3.4, которым нужно быстро протестировать работу части своей программы на предмет времени выполнения. Данный профайлер предлагает простое, быстрое и легковесное решение данной проблемы, упрощая жизнь разработчикам и тестировщикам.

### 6.3. Экономические преимущества разработки по сравнению с отечественными или зарубежными аналогами

На момент создания программы наиболее используемыми аналогами в области профилировщиков являются: Callgrind, Google perftools и EasyProfiler.

Общий недостаток всех этих продуктов — их сложность в использовании, которое требует установки соответствующей программы с графическим интерфейсом, далее ее линковка с тестируемой программой, далее вызов определенных конструкций из командной строки. Даже после завершения работы профайлера сложно правильно понять и трактовать полученные результаты. Более того, невозможно понять, как именно работал данный профайлер.

Есть и другие недостатки данных решений — например, это привязка к конкретной операционной системе. Например, callgrind является частью инструмента valgrind, текущих версий которого нет ни для Mac OS, ни для Windows. Google perftools, в свою очередь, не может корректно отрабатывать на малых программах, так как использует довольно старые библиотеки операций над временем в C++. EasyProfiler делает ставку на GUI, без которого работа сервиса становится непонятна из-за большого количества выводимых данных и большого количества команд.

Также, Callgrind и Google perftools для работы требуют прописывать свои флаги компиляции, отличные от тех, чем пользуется программист при компилировании своей программы. Это затрудняет понимание компилирующих опций, усложняя процесс debug-a. EasyProfiler, в свою очередь, использует только стандартные флаги компиляции, которые лишь задают путь к необходимым для работы профайлера библиотекам, не

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.04-01 ТЗ				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

засоряя тем самым терминал. Это весомое преимущество данного сервиса перед другими, и задача разрабатываемого в данной работе профайлера будет «унаследовать» этот принцип, максимально его оптимизировав (по возможности используя только уже встроенные в стандартную библиотеку языка C++ библиотеки).

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.04-01 ТЗ				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

## 7. СТАДИИ И ЭТАПЫ РАЗРАБОТКИ

### 7.1. Необходимые стадии разработки, этапы и содержание работ

Стадии и этапы разработки были выявлены с учетом ГОСТ 19.102-77 [6]:

Таблица 1 – Стадии разработки, этапы и содержание работ

Стадии разработки	Этапы работ	Содержание работ
I. Техническое задание	Обоснование необходимости разработки программы	Постановка задачи
		Сбор исходных материалов
		Выбор и обоснование критериев эффективности и качества разрабатываемой программы
	Разработка и утверждение технического задания	Определение требований к программе
		Определение стадий, этапов и сроков разработки программы и документации на нее
		Определение необходимости проведения научно-исследовательских работ на последующих стадиях
		Согласование и утверждение технического задания
II. Рабочий проект	Разработка программы	Программирование и отладка программы
	Разработка программной документации	Разработка программных документов в соответствии с требованиями ГОСТ 19.101-77 [5]
	Испытания программы	Разработка, согласование и утверждение порядка и методики испытаний
		Проведение предварительных испытаний
		Корректировка программы и программной документации по результатам испытаний

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.04-01 ТЗ				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата



Продолжение Таблицы 1

Стадии разработки	Этапы работ	Содержание работ
III. Внедрение	Подготовка и защита программного продукта.	Утверждение даты защиты программного продукта.
		Подготовка программы и программной документации для презентации и защиты.
		Представление разработанного программного продукта руководителю и получение отзыва.
		Загрузка Пояснительной записки в систему Антиплагиат через LMS (Learning Management System) НИУ ВШЭ
		Загрузка материалов курсового проекта в LMS (Learning Management System) НИУ ВШЭ, проект дисциплины «Курсовой проект, 2 курс ПИ» (см. п. 5.2)
		Защита программного продукта (курсового проекта) комиссии.

## 7.2. Сроки разработки и исполнители

Разработка должна закончиться к 25 мая 2023 года.

Исполнитель: Фортов Егор Кириллович, студент группы БПИ214 факультета компьютерных наук НИУ ВШЭ.

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.04-01 ТЗ				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

## 8. ПОРЯДОК КОНТРОЛЯ И ПРИЁМКИ

Проверка программного продукта, в том числе и на соответствие техническому заданию, осуществляется заказчиком совместно с исполнителем согласно «Программе и методике испытаний», а также пункту 5.2

Защита выполненного проекта осуществляется комиссии, состоящей из преподавателей департамента программной инженерии, в утверждённые приказом декана ФКН сроки.

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.04-01 ТЗ				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

## СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

- 1) EasyProfiler — простой профилировщик кода [Электронный ресурс] / Сергей @yse. Режим доступа: <https://habr.com/ru/post/318142/>, свободный. (дата обращения: 10.02.2023)
- 2) ValGrind – содержит профилировщик CallGrind [Электронный ресурс] / Pointfor Services. Режим доступа: <https://valgrind.org/info/platforms.html>, свободный. (дата обращения: 10.02.2023)
- 3) Особенности профилирования программ на C++ [Электронный ресурс] / @svr\_91. Режим доступа: <https://habr.com/ru/post/482040/>, свободный. (дата обращения: 10.02.2023)
- 4) ГОСТ 15150-69 Машины, приборы и другие технические изделия. Исполнения для различных климатических районов. Категории, условия эксплуатации, хранения и транспортирования в части воздействия климатических факторов внешней среды. – М.: Изд-во стандартов, 1997.
- 5) ГОСТ 19.101-77 Виды программ и программных документов. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
- 6) ГОСТ 19.102-77 Стадии разработки. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
- 7) ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
- 8) ГОСТ 19.201-78 Техническое задание. Требования к содержанию и оформлению. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
- 9) ГОСТ 19.301-79 Программа и методика испытаний. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
- 10) ГОСТ 19.401-78 Текст программы. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
- 11) ГОСТ 19.404-79 Пояснительная записка. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
- 12) ГОСТ 19.505-79 Руководство оператора. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
- 13) ГОСТ 19.602-78 Правила дублирования, учета и хранения программных документов, выполненных печатным способом. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
- 14) Lightweight profiler library for c++ [Электронный ресурс] / @yse. Режим доступа: [https://github.com/yse/easy\\_profiler?ysclid=ldz43zqbit864077970](https://github.com/yse/easy_profiler?ysclid=ldz43zqbit864077970), свободный. (дата обращения: 10.02.2023)
- 15) Статья про профилирование программ [Электронный ресурс] / Mateen Ulhaq. Режим доступа: <https://stackoverflow.com/questions/375913/how-do-i-profile-c-code-running-on-linux>, свободный. (дата обращения: 10.02.2023)
- 16) Профилировщик кода от Google [Электронный ресурс] / @alk. Режим доступа: <https://github.com/gperftools/gperftools?ysclid=ldz4785f5i699002535>, свободный. (дата обращения: 10.02.2023)

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.04-01 ТЗ				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

17) Статья про профилирование программ [Электронный ресурс] / Wikipedia.  
Режим доступа: [https://en.wikipedia.org/wiki/Profiling\\_\(computer\\_programming\)](https://en.wikipedia.org/wiki/Profiling_(computer_programming)), свободный.  
(дата обращения: 10.02.2023)

18) Статья про тестирование и профилирование программ [Электронный ресурс] / Wikipedia. Режим доступа: [https://en.wikibooks.org/wiki/Introduction\\_to\\_Software\\_Engineering/Testing/Profiling](https://en.wikibooks.org/wiki/Introduction_to_Software_Engineering/Testing/Profiling), свободный. (дата обращения: 10.02.2023)

19) Видео про профилирование программ [Электронный ресурс] / Wikipedia. Режим доступа: <https://yandex.ru/video/preview/1757229695302647836?family=yes>, свободный. (дата обращения: 10.02.2023)

20) Флаги компиляции callgrind [Электронный ресурс]; Режим доступа: <https://manpages.org/valgrind> свободный. (дата обращения: 10.02.2023)

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.04-01 ТЗ				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

## ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

[illegible]