|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.01 Информатика и вычислительная техника;**

**ОТЧЕТ**

|  |  |
| --- | --- |
| **По рубежному контролю №** | 1 |

**Название:** Алгоритм Adler-32

**Дисциплина:** Анализ алгоритмов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ7-52Б |  |  | Д.М.Блохин |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | Л.Л.Волкова |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2020

Оглавление

[Введение 3](#_Toc64365372)

[1.Аналитическая часть 4](#_Toc64365373)

[1.1 Описание алгоритма 4](#_Toc64365374)

[1.2 Вывод 4](#_Toc64365375)

[2.Конструкторская часть 5](#_Toc64365376)

[2.1 Разработка алгоритма 5](#_Toc64365377)

[2.2 Вывод 6](#_Toc64365378)

[3.Технологическая часть 7](#_Toc64365379)

[3.1 Средства реализации 7](#_Toc64365380)

[3.2 Листинг кода 7](#_Toc64365381)

[3.3 Пример работы 8](#_Toc64365382)

[3.4 Вывод 8](#_Toc64365383)

[4.Экспериментальная часть 9](#_Toc64365384)

[4.1 Сравнение стандартной и модифицированной реализации алгоритмов 9](#_Toc64365385)

[4.2 Вывод 9](#_Toc64365386)

[Заключение 10](#_Toc64365387)

[Список использованной литературы 11](#_Toc64365388)

# **Введение**

Adler-32 – хеш-функция, разработанная Марком Адлером, является модификацией контрольной суммы Fletcher. Вычисляет значение контрольной суммы в соответствии с RFC 1950 для массива байт или его фрагмента. Данный алгоритм расчёта контрольной суммы отличается от CRC32 производительностью. Adler-32 используется в библиотеке Zlib[1]. Rolling checksum версия функции используется в утилите rsync.

Так же как и в случае контрольной суммы Fletcher[2], при разработке суммы Adler стояла задача получения контрольной суммы с эффективностью обнаружения ошибок сравнимой с CRC. Хотя показатели поиска ошибок контрольных сумм Adler[3] и Fletcher практически такие же как и у относительно слабых CRC, они ведут себя гораздо хуже, чем хорошие CRC, в некоторых важных случаях.

**1.Аналитическая часть**

В данном разделе будут рассмотрена теоретическая часть хеш-функции Adler-32.

* 1. **Описание алгоритма**

Контрольная сумма Adler-32 получается путём вычисления двух 16-битных контрольных сумм **A** и **Б** и конкатенации их бит в 32-битное целое. **А** равняется сумме всех байт в строке плюс один, а **Б** является суммой всех отдельных значений **А** на каждом шаге. В начале выполнения функции Adler-32, **А** инициализируется единицей, а **Б** нулем. Суммы берутся по модулю 65521 (самое большое простое число меньшее чем 216). Байты записываются в сетевом порядке, **Б** занимает 2 старших байта.

* 1. **Вывод**

В данном разделе была рассмотрена хеш-функции Adler-32.

**2.Конструкторская часть**

В данной части будут представлены схемы реализуемых алгоритмов.

**2.1 Разработка алгоритма**

На рисунках 1 и 2 представлены реализации хеш-функции Adler-32.

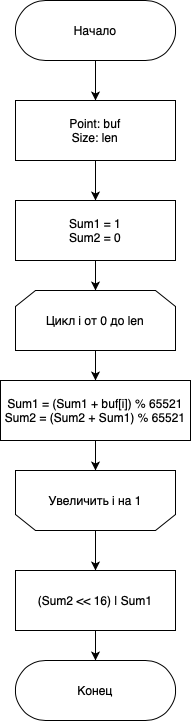


Рисунок 1: Стандартный алгоритм хеш-функции Adler-32.

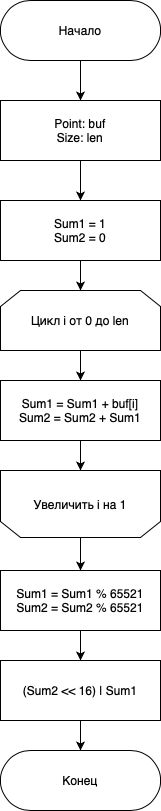


Рисунок 2: Модифицированный алгоритм хещ-функции Adler-32

**2.2 Вывод**

В данном разделе были рассмотрены схемы реализуемых алгоритмов.

# **3.Технологическая часть**

В данном разделе будут описаны средства реализации и листинг кода.

## **3.1 Средства реализации**

В качестве языка программирования для реализации программы был выбран язык C++, потому что язык C++ имеет высокую вычислительную производительность[4]. Для замера процессорного времени была использована функция был использован написанный класс Timer.

## **3.2 Листинг кода**

На листингах 3.1 – 3.2 представлены коды реализации алгоритмов хэш-функции Adler-32.

Листинг 3.1: Стандартная реализация

#define BASE 65521

uint32\_t adler32(**unsigned char** \*buf, **int** len){  
 uint32\_t sum1 = 1;  
 uint32\_t sum2 = 0;  
  
 **for** (**int** i = 0; i < len; i++){  
 sum1 = (sum1 + buf[i]) % BASE;  
 sum2 = (sum2 + sum1) % BASE;  
 }  
  
 **return** (sum2 << 16) | sum1;  
}

Листинг 3.2: Модифицированная реализация

#define BASE 65521

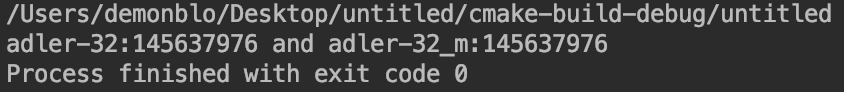
uint32\_t adler32\_m(**unsigned char** \*buf, **int** len){  
 uint32\_t sum1 = 1;  
 uint32\_t sum2 = 0;  
  
 **for** (**int** i = 0; i < len; i++){  
 sum1 = (sum1 + buf[i]);  
 sum2 = (sum2 + sum1);  
 }  
  
 sum1 %= BASE;  
 sum2 %= BASE;  
 **return** (sum2 << 16) | sum1;  
}

## **3.3 Пример работы**

На рисунках 1 и 2 представлены примеры работы программы, демонстрирующие правильное выполнение алгоритмов.

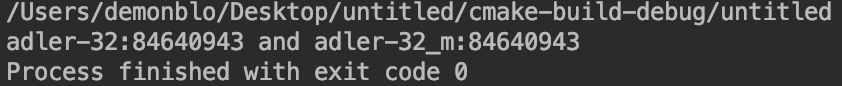
Входная строка: Then took the other, as just as fair And having perhaps the better claim Because it was grassy and wanted wear; Though as for that the passing there Had worn them really about the same.

Рисунок 1: Пример работы программы



Входная строка: Then took the other, as just as fair And having perhaps the better claim Because it was grassy and wanted wear; Though as for that the passing there Had worn them really about the same.Then took the other, as just as fair And having perhaps the better claim Because it was grassy and wanted wear; Though as for that the passing there Had worn them really about the same.

Рисунок 2: Пример работы программы со строкой большей размерности



Из примеров работы программы можно сделать вывод, что оба алгоритма работают корректно.

**3.4 Вывод**

В данном разделе были представлены средства реализации и листинг реализованных алгоритмов.

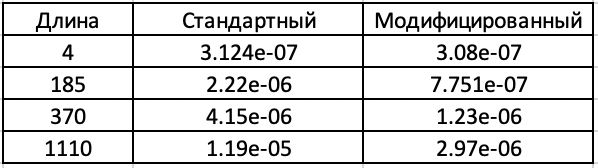
**4.Экспериментальная часть**

В данной части работы будет приведен анализ алгоритмов на основе экспериментальных данных.

**4.1 Сравнение стандартной и модифицированной реализации алгоритмов**

Для сравнения времени работы двух реализаций алгоритмов хеш-функции Adler-32 длина входной строки менялась и имела 4 различных значения. Эксперимент для более точного результата повторялся 50 раз. Итоговый результат рассчитывался как средний из полученных результатов. Результаты измерений показаны в таблице 4.1.

Рисунок 4.1: Время работы реализаций алгоритмов хэш-функции Adler-32



Можно сделать вывод, что при небольшой длине входной строки разницы во времени работы практически нет, но с увеличением строки разница во времени существенно растет, так при 185 символах модифицированный алгоритм выполнялся в 2.83 раза быстрее, а при длине в 1110 символов выполнялся в 4 раза быстрее, то есть разница во времени постепенно растет.

**4.2 Вывод**

В данной части работы был приведен анализ алгоритмов на основе экспериментальных данных.

**Заключение**

В ходе выполнения рубежного контроля был изучен и оптимизирован алгоритм хеш-функции Adler-32. Для этого были реализованы две реализации алгоритма хеш-функции. Также была оценена эффективность двух реализаций: при увеличении длины входной строки модифицированный алгоритм постепенно выигрывает у стандартного алгоритма по времени в разы.

**Список использованной литературы**

[1] [Электронный ресурс] – Режим доступа: https://zlib.net, свободный – (1.12.2020)

[2] [Электронный ресурс] – Режим доступа: https://hashtab.ru/algorithms/adler-32.html, свободный – (1.12.2020)

[3] [Электронный ресурс] – Режим доступа: https://amp.ww.google-info.org/870602/1/adler-32.html, свободный – (1.12.2020)

[4] [Электронный ресурс] – Режим доступа: https://en.cppreference.com/w/, свободный – (1.12.2020)