УДК 004.3:51.37 DOI: 10.31799/2077-5687-2026-1-31-46

ПРОГРАММНАЯ РЕАЛИЗАЦИЯ МЕТОДА БЫСТРОЙ АППРОКСИМАЦИИ ЕВКЛИДОВА РАССТОЯНИЯ С ИСПОЛЬЗОВАНИЕМ МИКРОКОНТРОЛЛЕРОВ СЕМЕЙСТВА STM32

И. В. Корытцев, Г. Ф. Э. Док Цихон
Санкт-Петербургский государственный университет аэрокосмического приборостроения

В работе рассмотрена оптимизация процесса вычисления модуля комплексного числа методом быстрой аппроксимации Евклидова расстояния с применением аппаратной платформы STM32F3Discovery на базе микроконтроллера STM32F303VC. Была рассмотрена проблематика вычисления модуля комплексного числа в условиях ограниченности временных и вычислительных ресурсов, приведены основные аппаратные характеристики микроконтроллера, использующегося для реализации метода быстрой аппроксимации, а также проведено сравнение методов обработки вещественных коэффициентов в рамках применения рассматриваемого метода. В качестве результатов данного исследования были приведены временные интервалы и величины погрешностей вычисления модуля комплексного числа тремя способами: с использованием программно-эмулированных функций нахождения квадратного корня и возведения в степень, с использованием вещественных коэффициентов метода быстрой аппроксимации, с использованием целочисленных эквивалентов вещественных коэффициентов метода быстрой аппроксимации. Для сравнительного анализа результаты применения всех трех способов вычисления модуля комплексного числа были выведены в виде графиков, показывающих эффективность применения рассматриваемого метода быстрой аппроксимации Евклидова расстояния.

Ключевые слова: модуль комплексного числа, Евклидово расстояние, микроконтроллер, оптимизация, время выполнения.

Для цитирования:
Корытцев, И. В. Программная реализация метода быстрой аппроксимации Евклидова расстояния с использованием микроконтроллеров семейства STM32 / И. В. Корытцев, Г. Ф. Э. Док Цихон // Системный анализ и логистика. – 2026. – № 1(49). – с. 31-46. DOI: 10.31799/2077-5687-2026-1-31-46.

SOFTWARE IMPLEMENTATION OF THE EUCLIDEAN DISTANCE FAST APPROXIMATION METHOD USING STM32 FAMILY MICROCONTROLLERS

I. V. Koryttsev, G. Cichon
St. Petersburg State University of Aerospace Instrumentation

The paper considers the optimization of the process of calculating the modulus of a complex number by the method of fast approximation of the Euclidean distance using the STM32F3Discovery hardware platform based on the STM32F303VC microcontroller. The problems of calculating the modulus of a complex number in conditions of limited time and computing resources were considered, the main hardware characteristics of the microcontroller used to implement the fast approximation method were presented, and the methods of processing real coefficients in the framework of the application of the considered method were compared. As the results of this study, time intervals and error values for calculating the modulus of a complex number were given in three ways: using programmatically emulated functions for finding the square root and exponentiation, using real coefficients of the fast approximation method, and using integer equivalents of real coefficients of the fast approximation method. For comparative analysis, the results of applying all three methods of calculating the modulus of a complex number were displayed in the form of graphs showing the effectiveness of using the considered method of rapid approximation of the Euclidean distance.

Keywords: complex number module, Euclidean distance, microcontroller, optimization, execution time.

For citation:
Koryttsev, I. V. Software implementation of a method for fast approximation of the Euclidean distance using STM32 family microcontrollers / I. V. Koryttsev, Cichon Gordon // System analysis and logistics. – 2026. – № 1(49). – p. 31-46. DOI: 10.31799/2077-5687-2026-1-31-46.

Введение

Вычисление значения модуля комплексного числа z = x + jy, (где x и jy — действительная и мнимая части комплексного числа z соответственно), именуемого также Евклидовым расстоянием или L2-нормой комплексного числа, производится следующим образом:

$$z=\sqrt{(x^2+y^2)} \quad (1)$$

где z – модуль комплексного числа;
x – действительная часть комплексного числа;
y – мнимая часть комплексного числа.

Решение данного математического выражения является распространенной прикладной задачей для различных информационных систем, таких как: системы анализа электрических цепей, программы векторного анализа, программы анализа устойчивости систем и т. д. Особое значение данное понятие имеет в информационных системах обработки данных в реальном времени, например, расчет параметров радиосигнала и компьютерная графика. Одним из ключевых требований к функционированию подобных систем является быстродействие, обеспечивающее своевременный прием, обработку и передачу данных. Реализация выполнения программно-аппаратной платформой ряда вычислительных операций за относительно малый промежуток времени в условиях ограниченности ресурсов, таких как объем внутренней памяти, частота CPU/MCU, объем ОЗУ и других, является относительно трудоёмкой задачей для рассматриваемого аппаратного средства. Таким образом, в случае малой эффективности существующей конфигурации встроенной системы, а также для экономии средств на развертывание дополнительных вычислительных ресурсов, оптимальным подходом является оптимизация исполняемого программного кода путем модификации и упрощения соответствующих математических методов, лежащих в его основе. Один из таких подходов был предложен в [1], основная суть которого заключается в переходе от вычисления модуля комплексного числа по формуле (1) к линейной функции вида: w’ = ax + by + c, где w’ – модуль комплексного числа, x,y – действительная и мнимая части комплексного числа соответственно, a, b, c – подбираемые коэффициенты. Основной проблематикой выражения (1) с точки зрения ее реализации средствами вычислительной техники является наличие ресурсоемких вычислительных операций - определение квадратного корня и возведение в степень, которые в условиях ограниченности вычислительных ресурсов микроконтроллеров и некоторых процессоров выполняются средствами программной эмуляции, что представляет серьезную проблему для производительности систем в условиях реального времени. Метод, предложенный в [1], предлагает аппроксимацию искомого значения путем поиска коэффициентов и умножения их на переменные x и y. Недостатком подобного метода является присутствие погрешности в результате вычисления модуля комплексного числа таким способом, однако авторам [1] удалось минимизировать ее, снизив с 41.2% до 3.95%.

Целью настоящей работы является применение данного метода аппроксимации для вычисления модуля комплексного числа ресурсами микроконтроллера STM32F303VC и сравнение временных и статистических параметров разных методов вычисления.

Аппаратное обеспечение

Для применения математического метода аппроксимации вычисления модуля комплексного числа, рассмотренного в [1], была использована аппаратная платформа STM32F3Discovery MB1035D на базе микроконтроллера серии STM32F303VC от компании STMicroelectronics. Данный микроконтроллер обладает 256-Кбайтной Flash-памятью и 48-Кбайтной RAM-памятью [2]. Архитектура MCU – 32-битная, основанная на ядре ARM Cortex M4. Тактовая частота контроллера составляет 72 МГц, производительность оценивается в 90 DMIPS [2]. Также, немаловажным компонентом для данного исследования является наличие аппаратного блока чисел с плавающей точкой одинарной точности FPU. Для связи с ПК и прошивки используется порт USB и встроенный в плату программатор ST-Link.

Для конфигурации данного контроллера перед написанием программного кода использовалась среда конфигурирования STM32CubeMX. Для написания, отладки и прошивки программного кода использовалась среда разработки STM32CubeIDE. В качестве

компилятора исходного кода использовался GCC. Внешний вид используемой аппаратной платформы представлен на рисунке 1.

Рис. 1. Аппаратная платформа STM32F3Discovery

Реализация аппроксимации

Как было упомянуто ранее, ключевой особенностью рассматриваемого метода аппроксимации является переход от формы (1) к линейной форме вида w’ = ax + by + c, где значения коэффициентов a и b были определены как:

a = 0.96043, b = 0.39782. Коэффициент C при этом приравнивается нулю. Подробное описание получения этих значений представлено в [1]. Стоит отметить, что использование такого подхода к вычислению модуля комплексного числа позволит избежать использования функции SQRT() и возведение в степень, заменив их на одну операцию сложения и две операции умножения.

Для демонстрации эффективности данного метода аппроксимации предполагается провести вычисление модуля комплексного числа программно эмулируемой функцией вычисления квадратного корня SQRT() и с использованием линейной функции w’ = ax + by + c. Так как в микроконтроллере STM32F303VC присутствует модуль FPU, то представляется с технической точки зрения полезным сравнить скорость выполнения операций с вещественными коэффициентами типа float и аналогичных операций с приведением данных коэффициентов к виду числа с фиксированной точкой.

Для приведения вещественных коэффициентов a = 0.96043 и b = 0.39782 к виду числа с фиксированной точкой необходимо определить их целочисленный эквивалент. Для этого значение коэффициента следует умножить на 256. Соответствующие преобразования представлены ниже:

$$a = 0.96043 \Rightarrow 0.96043 = \frac{x}{256},\ x = 244.87 \approx 245$$
$$b = 0.39782 \Rightarrow 0.39782 = \frac{x}{256},\ x = 101.84 \approx 102$$

где a, b – подбираемые коэффициенты, x – искомый целочисленный эквивалент коэффициентам a и b.

На данном этапе видно наличие погрешности в представлении вещественных чисел в эквивалентный целочисленный вид для формирования числа с фиксированной точкой. Значение 244.87 и 101.84 невозможно представить в целочисленном виде без потери десятой и сотой частей. Следовательно, данное преобразование заложит определённый процент ошибки в будущих вычислениях.

Алгоритм программной реализации метода аппроксимации

Программная реализация предложенного метода аппроксимации определяется

следующим алгоритмом:

1) Составление диапазонов переменных x и y для малых (десятки), средних (сотни), больших (тысячи), сверхбольших (десятки тысяч) значений;
2) Вычисление значения модуля комплексного числа с использованием функций SQRT() и возведения в степень POW() из библиотеки math.h языка Си и использование полученного значения как эталонного для вычисления величины погрешности метода аппроксимации. Вычисление времени выполнения;
3) Вычисление модуля комплексного числа по линейной формуле w’ = ax + by + c используя вещественные коэффициенты a и b типа float. Коэффициент C приравнивается к нулю. Вычисление времени выполнения и определение процента ошибки;
4) Вычисление модуля комплексного числа по линейной формуле w’ = ax + by + c используя коэффициенты a и b в виде чисел с фиксированной точкой. Вычисление времени выполнения и определение процента ошибки;
5) Анализ и сравнение полученных результатов вычисления и времени.

Для вычисления времени выполнения интересующего блока кода в рамках данной работы был использован встроенный 32-разрядный таймер DWT (Data Watchpoint and Trace unit). Значение данного таймера увеличивает на один каждый такт микроконтроллера [3]. Для микроконтроллера STM32F303VC максимальная тактовая частота составляет 72 МГц, соответственно период одного такта равен 13.88 нс [3]. Перед функционированием счетчика необходимо разрешить его работу следующими процедурами, представленными в листинге 1:

Листинг 1. Процедуры инициализации счетчика

```
SCB_DEMCR |= CoreDebug_DEMCR_TRCENA_Msk; // разрешение счётчика
DWT_CONTROL |= DWT_CTRL_CYCCNTENA_Msk; // запуск счётчика
```

Для реализации счета времени перед интересующим блоком кода необходимо инициализировать переменную-счетчик текущим значением, хранящимся в DWT на момент выполнения интересующего блока кода [3]. В целом, блок вычисления модуля комплексного числа с помощью функции SQRT(), фиксации времени выполнения кода и вывода информации в терминал с помощью протокола UART представлен в листинге 2:

Листинг 2. Программный код вычисления комплексного числа с помощью функции SQRT().

```
Mcounter = *DWT_CYCCNT | 1; // time counting start
SQRT_result = sqrt(pow(lowLevelNumber[0][i], 2) + pow(lowLevelNumber[1][i], 2));
count = *DWT_CYCCNT-Mcounter;// time counting stop
op_time = count / 72000000.0;
sprintf(str, "x: %d, y: %d, SQRT result: %f, Counts: %d, Time: %f
sec.\r\n",lowLevelNumber[0][i], lowLevelNumber[1][i], SQRT_result, count, op_time);
HAL_UART_Transmit(&huart1, str, strlen(str), time);
DWT->CYCCNT = 0U;
count = 0;
```

Как видно из кода, представленного в листинге 2, вычисление времени производится путем деления числа зафиксированных за время выполнения программы тактов в таймере DWT на тактовую частоту MCU – 72 МГц. Далее функцией sprintf() формируется текстовая строка, включающая в себя такие параметры, как: результат вычисления, количество тактов и время выполнения. Затем, с помощью функции HAL_UART_Transmit() сформированные данные отправляются в терминал ПК по протоколу UART [4].

Для вычисления модуля комплексного числа рассматриваемым методом аппроксимации

с использованием вещественных коэффициентов типа float используется код, представленный в листинге 3.

Листинг 3 Программный код вычисления комплексного числа с помощью метода аппроксимации.

```
Mcounter = *DWT_CYCCNT | 1; // time counting start
FPU_result = a * lowLevelNumber[0][i] + b * lowLevelNumber[1][i];
count = *DWT_CYCCNT-Mcounter;// time counting stop
FPU_error = ((SQRT_result - FPU_result) / SQRT_result) * 100;
if (FPU_error < 0) FPU_error *= (-1);
if (FPU_error > worstFPUErrorResult[2])
{
    worstFPUErrorResult[0] = lowLevelNumber[0][i];
    worstFPUErrorResult[1] = lowLevelNumber[1][i];
    worstFPUErrorResult[2] = FPU_error;

}
FPU_error_summ += FPU_error;
op_time = count / 72000000.0;
sprintf(str, "                  FPU result: %f, FPU error: %f perc., Counts: %d, Time: %f
sec.\r\n",FPU_result, FPU_error, count, op_time);
HAL_UART_Transmit(&huart1, str, strlen(str), time);
DWT->CYCCNT = 0U;
count = 0;
```

Процедура вычисления времени в данном (и в дальнейшем) случае аналогична. Для вычисления величины ошибки используется формула:

$$E = \frac{w - w'}{w}$$

где w – эталонный результат вычисления модуля с использованием функции SQRT(), принимаемой максимально точной, w’ – результат линейной функции аппроксимации.

Далее значение ошибки переводится в проценты и выводится в терминал. Кроме того, реализована процедура вычисления среднего значения ошибки по всей выборке и максимальной ошибки в выборке. Для вычисления модуля методом аппроксимации с использованием чисел с фиксированной точкой используется аналогичный код.

Результаты вычислений

Результатом исследования являются данные величины ошибки вычисления модуля комплексного числа и времени выполнения вычислительной процедуры. На рис. 2 представлена гистограмма распределения средней и максимальной ошибок вычисления (E, %) с помощью FPU и фиксированной точки для разных диапазонов входных значений x и y (десятки, сотни, тысячи, десятки тысяч).

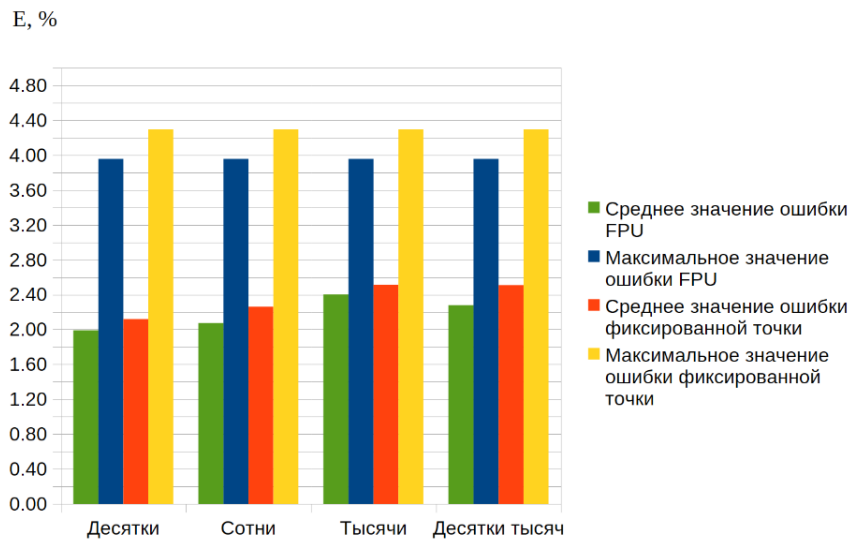

Рис. 2. Распределение средней и максимальной ошибок для различных диапазонов x и y

Как видно из гистограммы на рис. 2 среднее значение ошибки вычисления аппроксимированного значения модуля комплексного числа с использованием блока FPU не превышает 2.4%, а максимальное значение ошибки достигает значения 3.95 %. Для выборок «десятки», «сотни» и «тысячи» характерно постепенное увеличение среднего значения ошибки, как для значений, вычисленных с помощью FPU, так и для целочисленного вычисления модуля с помощью коэффициентов, представленных числом с фиксированной точкой. Можно заметить разницу между величиной ошибок FPU и операций с фиксированной точкой (2.40% максимальная средняя ошибка для FPU и 2.51 % - для вычисления с фиксированной точкой), заключающейся в большей ограниченности точности представления числа с фиксированной точкой, в отличие от аппаратно реализованного с помощью FPU числа с плавающей точкой.

В целом, максимальная ошибка вычисления для FPU не превышает 3.95%, что доказывает теоретическую погрешность, указанную в [1] (3.9566%). Максимальная ошибка вычисления с помощью чисел с фиксированной точкой не превышает 4.29%. Разница двух показателей объясняется ограниченной точностью представления вещественных чисел в целочисленном виде (с фиксированной точкой) по сравнению с вещественными числами с плавающей точкой одинарной точности (float), используемых аппаратным блоком FPU.

В целом, использование чисел с фиксированной точкой для реализации вычислений на данной платформе не имеет смысла ввиду наличия аппаратного решения для обработки вещественных чисел. Однако, данный подход показывает достаточно высокую точность рассматриваемой аппроксимации (разница в точности между FPU и счетом с фиксированной точкой составляет 0.34%), что является хорошим решением не только для высокопроизводительных 32-битных контроллеров, но и для более ограниченных аппаратных платформ, например, 8-битных микроконтроллеров без аппаратной реализации вычислений квадратного корня и вещественных чисел.

На рисунке 3 представлена гистограмма времени выполнения процедуры вычисления (t, мкс) модуля комплексного числа функциями SQRT(), блока FPU и целочисленного счета.

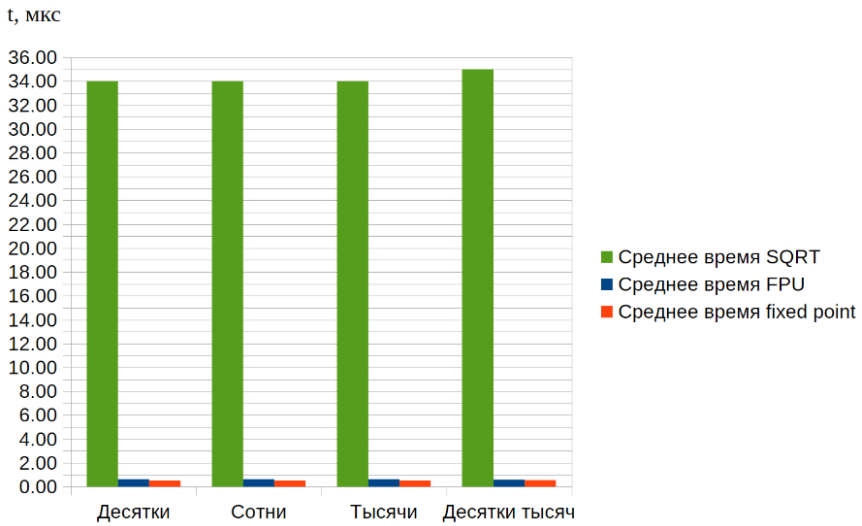

Рис. 3. Время выполнения вычислений функциями SQRT(), блока FPU и целочисленного вычисления

Как видно по гистограмме на рис. 3 программно-реализованная функция SQRT() и две операции возведения в степень используют относительно большое количество процессорного времени, достигая 35 мкс, тогда как вычисления аппроксимированного значения обоими рассмотренными способами не превышает 1 мкс. В частности, среднее время выполнения вычисления блоком FPU составляет 625 нс, с использованием чисел с фиксированной точкой — 513 нс. Разница в 112 нс объясняется отсутствием необходимости обращения к отдельному аппаратному блоку для выполнения операций с целочисленными величинами с фиксированной точкой.

Заключение

Как итог, на основании представленных выше гистограмм (рис. 2, рис. 3) отчетливо видна эффективность предложенного метода аппроксимации вычисления L2-нормы комплексного числа, заключающаяся в многократном ускорении данной операции (0.625 мкс вместо 34 мкс) путем исключения операций возведения в степень и извлечения квадратного корня. Также, благодаря использованию аппаратного блока FPU для обработки вещественных чисел с плавающей точкой была подтверждена величина ошибки данного метода (3.95%), заявленная в [1]. Помимо использования аппаратных особенностей рассматриваемой платформы для вычисления L2-нормы, была рассмотрена реализация данной аппроксимации для преобразованных вещественных коэффициентов a = 0.96043 и b = 0.39782 в числа с фиксированной точкой для ускорения процесса вычисления. Данные показали увеличение скорости вычисления L2-нормы на 112 нс при сравнении с аналогичным вычислением используя FPU (625 нс для FPU, 513 нс для чисел с фиксированной точкой). Однако, это сопровождается увеличением максимальной ошибки вычисления на 0.34% по сравнению с FPU (3.95% максимальная ошибка для FPU и 4.29% - для вычисления с фиксированной точкой).

Как итог, рассматриваемый метод аппроксимации представляет пользу для реализации вычисления модуля комплексного числа в вычислительных системах реального времени с ограниченными ресурсами, обладая при этом относительно малым временем выполнения и приемлемым процентом ошибки, возникающим при вычислении. Полный код функции main() данного исследования представлен в Приложении 1. Консольный вывод с параметрами вычислений представлен в Приложении 2.

СПИСОК ЛИТЕРАТУРЫ

1. *Док Цихон, Г. Ф. Э.* Быстрая аппроксимация Евклидова расстояния / Г. Ф. Э. Док Цихон, А. В. Шахомиров // Радиотехника. – 2024. – № 7. – С. 5-13.
2. ST. STM32F303xB STM32F303xC Datasheet [электронный ресурс] URL: https://www.st.com/en/microcontrollers-microprocessors/stm32f303vc.html (дата обращения: 04.11.2025).
3. StD. Счетчик DWT. Микросекундные паузы [электронный ресурс] URL: https://istarik.ru/blog/stm32/131.html (дата обращения: 04.11.2025).
4. ST wiki. Getting started with UART [электронный ресурс] URL: https://wiki.st.com/stm32mcu/wiki/Getting_started_with_UART (дата обращения: 06.11.2025).

Приложение 1. Программный код функции main() данного исследования

```
int main(void)
{
  *DEMCR = *DEMCR |= 0x01000000;
  *DWT_CYCCNT = 0;
  *DWT_CONTROL = *DWT_CONTROL | 1;


  HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();
  MX_I2C1_Init();
  MX_SPI1_Init();
  MX_USART1_UART_Init();
  MX_USB_PCD_Init();

  char str[200];
  int lowLevelNumber[2][16];
  int midLevelNumber[2][16];
  int highLevelNumber[2][16];
  int unroundHighNumbers[2][16];

        // x                        // y
  //lowLevelNumber[0][0] = 0; lowLevelNumber[1][0] = 0;
  lowLevelNumber[0][1] = 1; lowLevelNumber[1][1] = 0;
  lowLevelNumber[0][2] = 2; lowLevelNumber[1][2] = 1;
  lowLevelNumber[0][3] = 3; lowLevelNumber[1][3] = 2;
  lowLevelNumber[0][4] = 4; lowLevelNumber[1][4] = 3;
  lowLevelNumber[0][5] = 5; lowLevelNumber[1][5] = 4;
  lowLevelNumber[0][6] = 6; lowLevelNumber[1][6] = 5;
  lowLevelNumber[0][7] = 7; lowLevelNumber[1][7] = 6;
  lowLevelNumber[0][8] = 8; lowLevelNumber[1][8] = 7;
  lowLevelNumber[0][9] = 9; lowLevelNumber[1][9] = 8;
  lowLevelNumber[0][10] = 10; lowLevelNumber[1][10] = 9;
  lowLevelNumber[0][11] = 11; lowLevelNumber[1][11] = 10;
  lowLevelNumber[0][12] = 12; lowLevelNumber[1][12] = 11;
  lowLevelNumber[0][13] = 13; lowLevelNumber[1][13] = 12;
  lowLevelNumber[0][14] = 14; lowLevelNumber[1][14] = 13;
  lowLevelNumber[0][15] = 15; lowLevelNumber[1][15] = 14;

  midLevelNumber[0][1] = 112; midLevelNumber[1][1] = 0;
  midLevelNumber[0][2] = 265; midLevelNumber[1][2] = 132;
  midLevelNumber[0][3] = 354; midLevelNumber[1][3] = 288;
  midLevelNumber[0][4] = 415; midLevelNumber[1][4] = 354;
  midLevelNumber[0][5] = 501; midLevelNumber[1][5] = 422;
  midLevelNumber[0][6] = 610; midLevelNumber[1][6] = 506;
  midLevelNumber[0][7] = 771; midLevelNumber[1][7] = 643;
  midLevelNumber[0][8] = 810; midLevelNumber[1][8] = 754;
  midLevelNumber[0][9] = 941; midLevelNumber[1][9] = 865;
  midLevelNumber[0][10] = 1022; midLevelNumber[1][10] = 900;
  midLevelNumber[0][11] = 1103; midLevelNumber[1][11] = 1090;
  midLevelNumber[0][12] = 1220; midLevelNumber[1][12] = 1103;
  midLevelNumber[0][13] = 1360; midLevelNumber[1][13] = 1207;
  midLevelNumber[0][14] = 1400; midLevelNumber[1][14] = 1301;
  midLevelNumber[0][15] = 1503; midLevelNumber[1][15] = 1420;

  highLevelNumber[0][1] = 1010; highLevelNumber[1][1] = 0;
  highLevelNumber[0][2] = 2120; highLevelNumber[1][2] = 1060;
  highLevelNumber[0][3] = 3165; highLevelNumber[1][3] = 2998;
  highLevelNumber[0][4] = 4015; highLevelNumber[1][4] = 3658;
```

```
	highLevelNumber[0][5] = 5986; highLevelNumber[1][5] = 4031;
	highLevelNumber[0][6] = 6887; highLevelNumber[1][6] = 5132;
	highLevelNumber[0][7] = 7650; highLevelNumber[1][7] = 6010;
	highLevelNumber[0][8] = 8117; highLevelNumber[1][8] = 7991;
	highLevelNumber[0][9] = 9010; highLevelNumber[1][9] = 8540;
	highLevelNumber[0][10] = 10100; highLevelNumber[1][10] = 9020;
	highLevelNumber[0][11] = 11620; highLevelNumber[1][11] = 10010;
	highLevelNumber[0][12] = 12001; highLevelNumber[1][12] = 11022;
	highLevelNumber[0][13] = 13041; highLevelNumber[1][13] = 12001;
	highLevelNumber[0][14] = 14099; highLevelNumber[1][14] = 13700;
	highLevelNumber[0][15] = 15001; highLevelNumber[1][15] = 14000;

	unroundHighNumbers[0][1] = 15251; unroundHighNumbers[1][1] = 0;
	unroundHighNumbers[0][2] = 26152; unroundHighNumbers[1][2] = 17311;
	unroundHighNumbers[0][3] = 39575; unroundHighNumbers[1][3] = 2334;
	unroundHighNumbers[0][4] = 46111; unroundHighNumbers[1][4] = 3864;
	unroundHighNumbers[0][5] = 59587; unroundHighNumbers[1][5] = 472;
	unroundHighNumbers[0][6] = 67764; unroundHighNumbers[1][6] = 58722;
	unroundHighNumbers[0][7] = 77181; unroundHighNumbers[1][7] = 6152;
	unroundHighNumbers[0][8] = 81522; unroundHighNumbers[1][8] = 79901;
	unroundHighNumbers[0][9] = 91554; unroundHighNumbers[1][9] = 81315;
	unroundHighNumbers[0][10] = 105613; unroundHighNumbers[1][10] = 98001;
	unroundHighNumbers[0][11] = 114536; unroundHighNumbers[1][11] = 105492;
	unroundHighNumbers[0][12] = 128743; unroundHighNumbers[1][12] = 11365;
	unroundHighNumbers[0][13] = 136778; unroundHighNumbers[1][13] = 122689;
	unroundHighNumbers[0][14] = 145413; unroundHighNumbers[1][14] = 139223;
	unroundHighNumbers[0][15] = 154449; unroundHighNumbers[1][15] = 148333;

	long double SQRT_result;
	float FPU_result;
	int integer_result;

	float a = 0.96043;
	float b = 0.39782;

	int a_int = 245;
	int b_int = 102;

	float FPU_error, Int_error;
	float FPU_error_summ = 0, Int_error_summ = 0;
				//x  y  max
	float worstFPUErrorResult[3];
	worstFPUErrorResult[0] = 0; worstFPUErrorResult[1] = 0; worstFPUErrorResult[2] = 0;
	float worstIntErrorResult[3];
	worstIntErrorResult[0] = 0; worstIntErrorResult[1] = 0; worstIntErrorResult[2] = 0;

	float op_time = 0;

	//============================ LOW RANGE VALUES ==================================

	for(int i = 1; i <= 15; i++)
		{
		 Mcounter = *DWT_CYCCNT | 1; // time counting start
			SQRT_result = sqrt(pow(lowLevelNumber[0][i], 2) + pow(lowLevelNumber[1][i], 2));
		 count = *DWT_CYCCNT-Mcounter;// time counting stop
		 op_time = count / 72000000.0;
		 sprintf(str, "x: %d, y: %d, SQRT result: %f, Counts: %d, Time: %f sec.\r\n",lowLevelNumber[0][i],
lowLevelNumber[1][i], SQRT_result, count, op_time);
		 HAL_UART_Transmit(&huart1, str, strlen(str), time);
		 DWT->CYCCNT = 0U;
		 count = 0;

		 Mcounter = *DWT_CYCCNT | 1; // time counting start
			FPU_result = a * lowLevelNumber[0][i] + b * lowLevelNumber[1][i];
		 count = *DWT_CYCCNT-Mcounter;// time counting stop
		 FPU_error = ((SQRT_result - FPU_result) / SQRT_result) * 100;
		 if (FPU_error < 0) FPU_error *= (-1);
		 if (FPU_error > worstFPUErrorResult[2])
		 {
			worstFPUErrorResult[0] = lowLevelNumber[0][i];
			worstFPUErrorResult[1] = lowLevelNumber[1][i];
			worstFPUErrorResult[2] = FPU_error;

		 }
		 FPU_error_summ += FPU_error;
		 op_time = count / 72000000.0;
		 sprintf(str, "            FPU result: %f, FPU error: %f perc., Counts: %d, Time: %f
sec.\r\n",FPU_result, FPU_error, count, op_time);
		 HAL_UART_Transmit(&huart1, str, strlen(str), time);
		 DWT->CYCCNT = 0U;
		 count = 0;


		 Mcounter = *DWT_CYCCNT | 1; // time counting start
			integer_result = (a_int * lowLevelNumber[0][i] + b_int * lowLevelNumber[1][i]);
		 count = *DWT_CYCCNT-Mcounter;// time counting stop
		 Int_error = ((SQRT_result - (((float)integer_result) / 256)) / SQRT_result) * 100;
```

```
                    if (Int_error < 0) Int_error *= (-1);
                    if (Int_error > worstIntErrorResult[2])
                    {
                            worstIntErrorResult[0] = lowLevelNumber[0][i];
                            worstIntErrorResult[1] = lowLevelNumber[1][i];
                            worstIntErrorResult[2] = Int_error;
                    }
                    Int_error_summ += Int_error;
                    op_time = count / 72000000.0;
                    sprintf(str, "             INT result(fixed point): %d, (floatRepresent): %f, INT error: %f perc.,
Counts: %d, Time: %f sec.\r\n" ,integer_result, (((float)integer_result) / 256), Int_error, count, op_time);
                    HAL_UART_Transmit(&huart1, str, strlen(str), time);
                    DWT->CYCCNT = 0U;
                    count = 0;
                   }

                   sprintf(str, "Average FPU error: %f, max FPU error: %f, worst result when x: %f, y: %f\n\r",
(FPU_error_summ / 15), worstFPUErrorResult[2], worstFPUErrorResult[0], worstFPUErrorResult[1]);
                    HAL_UART_Transmit(&huart1, str, strlen(str), time);
                    sprintf(str, "Average INT error: %f, max INT error: %f, worst result when x: %f, y: %f\n\r",
(Int_error_summ / 15), worstIntErrorResult[2], worstIntErrorResult[0], worstIntErrorResult[1]);
                    HAL_UART_Transmit(&huart1, str, strlen(str), time);

          //============================ MID RANGE VALUES ====================================


          for(int i = 1; i <= 15; i++)
                    {
                     Mcounter = *DWT_CYCCNT | 1; // time counting start
                              SQRT_result = sqrt(pow(midLevelNumber[0][i], 2) + pow(midLevelNumber[1][i], 2));
                     count = *DWT_CYCCNT-Mcounter;// time counting stop
                     op_time = count / 72000000.0;
                     sprintf(str, "x: %d, y: %d, SQRT result: %lf, Counts: %d, Time: %f sec.\r\n",midLevelNumber[0][i],
midLevelNumber[1][i], SQRT_result, count, op_time);
                     HAL_UART_Transmit(&huart1, str, strlen(str), time);
                     DWT->CYCCNT = 0U;
                     count = 0;

                     Mcounter = *DWT_CYCCNT | 1; // time counting start
                              FPU_result = a * midLevelNumber[0][i] + b * midLevelNumber[1][i];
                     count = *DWT_CYCCNT-Mcounter;// time counting stop
                     FPU_error = ((SQRT_result - FPU_result) / SQRT_result) * 100;
                     if (FPU_error < 0) FPU_error *= (-1);
                     if (FPU_error > worstFPUErrorResult[2])
                     {
                              worstFPUErrorResult[0] = midLevelNumber[0][i];
                              worstFPUErrorResult[1] = midLevelNumber[1][i];
                              worstFPUErrorResult[2] = FPU_error;

                     }
                     FPU_error_summ += FPU_error;
                     op_time = count / 72000000.0;
                     sprintf(str, "             FPU result: %f, FPU error: %f perc., Counts: %d, Time: %f
sec.\r\n",FPU_result, FPU_error, count, op_time);
                     HAL_UART_Transmit(&huart1, str, strlen(str), time);
                     DWT->CYCCNT = 0U;
                     count = 0;


                     Mcounter = *DWT_CYCCNT | 1; // time counting start
                              integer_result = (a_int * midLevelNumber[0][i] + b_int * midLevelNumber[1][i]);
                     count = *DWT_CYCCNT-Mcounter;// time counting stop
                     Int_error = ((SQRT_result - (((float)integer_result) / 256)) / SQRT_result) * 100;
                     if (Int_error < 0) Int_error *= (-1);
                     if (Int_error > worstIntErrorResult[2])
                     {
                              worstIntErrorResult[0] = midLevelNumber[0][i];
                              worstIntErrorResult[1] = midLevelNumber[1][i];
                              worstIntErrorResult[2] = Int_error;
                     }
                     Int_error_summ += Int_error;
                     op_time = count / 72000000.0;
                     sprintf(str, "             INT result(fixed point): %d, (floatRepresent): %f, INT error: %f perc.,
Counts: %d, Time: %f sec.\r\n" ,integer_result, (((float)integer_result) / 256), Int_error, count, op_time);
                     HAL_UART_Transmit(&huart1, str, strlen(str), time);
                     DWT->CYCCNT = 0U;
                     count = 0;
                    }

                    sprintf(str, "Average FPU error: %f, max FPU error: %f, worst result when x: %f, y: %f\n\r",
(FPU_error_summ / 15), worstFPUErrorResult[2], worstFPUErrorResult[0], worstFPUErrorResult[1]);
                    HAL_UART_Transmit(&huart1, str, strlen(str), time);
                    sprintf(str, "Average INT error: %f, max INT error: %f, worst result when x: %f, y: %f\n\r",
(Int_error_summ / 15), worstIntErrorResult[2], worstIntErrorResult[0], worstIntErrorResult[1]);
                    HAL_UART_Transmit(&huart1, str, strlen(str), time);

            //============================ HIGH RANGE VALUES ====================================
```

```
        for(int i = 1; i <= 15; i++)
                {
                 Mcounter = *DWT_CYCCNT | 1; // time counting start
                          SQRT_result = sqrt(pow(highLevelNumber[0][i], 2) + pow(highLevelNumber[1][i], 2));
                 count = *DWT_CYCCNT-Mcounter;// time counting stop
                 op_time = count / 72000000.0;
                 sprintf(str, "x: %d, y: %d, SQRT result: %lf, Counts: %d, Time: %f sec.\r\n",highLevelNumber[0][i],
highLevelNumber[1][i], SQRT_result, count, op_time);
                 HAL_UART_Transmit(&huart1, str, strlen(str), time);
                 DWT->CYCCNT = 0U;
                 count = 0;

                 Mcounter = *DWT_CYCCNT | 1; // time counting start
                          FPU_result = a * highLevelNumber[0][i] + b * highLevelNumber[1][i];
                 count = *DWT_CYCCNT-Mcounter;// time counting stop
                 FPU_error = ((SQRT_result - FPU_result) / SQRT_result) * 100;
                 if (FPU_error < 0) FPU_error *= (-1);
                 if (FPU_error > worstFPUErrorResult[2])
                 {
                          worstFPUErrorResult[0] = highLevelNumber[0][i];
                          worstFPUErrorResult[1] = highLevelNumber[1][i];
                          worstFPUErrorResult[2] = FPU_error;

                 }
                 FPU_error_summ += FPU_error;
                 op_time = count / 72000000.0;
                 sprintf(str, "              FPU result: %f, FPU error: %f perc., Counts: %d, Time: %f
sec.\r\n",FPU_result, FPU_error, count, op_time);
                 HAL_UART_Transmit(&huart1, str, strlen(str), time);
                 DWT->CYCCNT = 0U;
                 count = 0;


                 Mcounter = *DWT_CYCCNT | 1; // time counting start
                          integer_result = (a_int * highLevelNumber[0][i] + b_int * highLevelNumber[1][i]);
                 count = *DWT_CYCCNT-Mcounter;// time counting stop
                 Int_error = ((SQRT_result - (((float)integer_result) / 256)) / SQRT_result) * 100;
                 if (Int_error < 0) Int_error *= (-1);
                 if (Int_error > worstIntErrorResult[2])
                 {
                          worstIntErrorResult[0] = highLevelNumber[0][i];
                          worstIntErrorResult[1] = highLevelNumber[1][i];
                          worstIntErrorResult[2] = Int_error;
                 }
                 Int_error_summ += Int_error;
                 op_time = count / 72000000.0;
                 sprintf(str, "              INT result(fixed point): %d, (floatRepresent): %f, INT error: %f perc.,
Counts: %d, Time: %f sec.\r\n" ,integer_result, (((float)integer_result) / 256), Int_error, count, op_time);
                 HAL_UART_Transmit(&huart1, str, strlen(str), time);
                 DWT->CYCCNT = 0U;
                 count = 0;
                }

                sprintf(str, "Average FPU error: %f, max FPU error: %f, worst result when x: %f, y: %f\n\r",
(FPU_error_summ / 15), worstFPUErrorResult[2], worstFPUErrorResult[0], worstFPUErrorResult[1]);
                HAL_UART_Transmit(&huart1, str, strlen(str), time);
                sprintf(str, "Average INT error: %f, max INT error: %f, worst result when x: %f, y: %f\n\r",
(Int_error_summ / 15), worstIntErrorResult[2], worstIntErrorResult[0], worstIntErrorResult[1]);
                HAL_UART_Transmit(&huart1, str, strlen(str), time);


        //============================ UNROUND RANGE VALUES ===================================

            for(int i = 1; i <= 15; i++)
                {
                 Mcounter = *DWT_CYCCNT | 1; // time counting start
                          SQRT_result = sqrt(pow(unroundHighNumbers[0][i], 2) + pow(unroundHighNumbers[1][i], 2));
                 count = *DWT_CYCCNT-Mcounter;// time counting stop
                 op_time = count / 72000000.0;
                 sprintf(str, "x: %d, y: %d, SQRT result: %lf, Counts: %d, Time: %f sec.\r\n",unroundHighNumbers[0][i],
unroundHighNumbers[1][i], SQRT_result, count, op_time);
                 HAL_UART_Transmit(&huart1, str, strlen(str), time);
                 DWT->CYCCNT = 0U;
                 count = 0;

                 Mcounter = *DWT_CYCCNT | 1; // time counting start
                          FPU_result = a * unroundHighNumbers[0][i] + b * unroundHighNumbers[1][i];
                 count = *DWT_CYCCNT-Mcounter;// time counting stop
                 FPU_error = ((SQRT_result - FPU_result) / SQRT_result) * 100;
                 if (FPU_error < 0) FPU_error *= (-1);
                 if (FPU_error > worstFPUErrorResult[2])
                 {
                          worstFPUErrorResult[0] = unroundHighNumbers[0][i];
                          worstFPUErrorResult[1] = unroundHighNumbers[1][i];
                          worstFPUErrorResult[2] = FPU_error;
```

```
                    }
                    FPU_error_summ += FPU_error;
                    op_time = count / 72000000.0;
                    sprintf(str, "              FPU result: %f, FPU error: %f perc., Counts: %d, Time: %f
sec.\r\n",FPU_result, FPU_error, count, op_time);
                    HAL_UART_Transmit(&huart1, str, strlen(str), time);
                    DWT->CYCCNT = 0U;
                    count = 0;


                    Mcounter = *DWT_CYCCNT | 1; // time counting start
                             integer_result = (a_int * unroundHighNumbers[0][i] + b_int * unroundHighNumbers[1][i]);
                    count = *DWT_CYCCNT-Mcounter;// time counting stop
                    Int_error = ((SQRT_result - (((float)integer_result) / 256)) / SQRT_result) * 100;
                    if (Int_error < 0) Int_error *= (-1);
                    if (Int_error > worstIntErrorResult[2])
                    {
                             worstIntErrorResult[0] = unroundHighNumbers[0][i];
                             worstIntErrorResult[1] = unroundHighNumbers[1][i];
                             worstIntErrorResult[2] = Int_error;
                    }
                    Int_error_summ += Int_error;
                    op_time = count / 72000000.0;
                    sprintf(str, "              INT result(fixed point): %d, (floatRepresent): %f, INT error: %f perc.,
Counts: %d, Time: %f sec.\r\n" ,integer_result, (((float)integer_result) / 256), Int_error, count, op_time);
                    HAL_UART_Transmit(&huart1, str, strlen(str), time);
                    DWT->CYCCNT = 0U;
                    count = 0;
                }

                sprintf(str, "Average FPU error: %f, max FPU error: %f, worst result when x: %f, y: %f\n\r",
(FPU_error_summ / 15), worstFPUErrorResult[2], worstFPUErrorResult[0], worstFPUErrorResult[1]);
                HAL_UART_Transmit(&huart1, str, strlen(str), time);
                sprintf(str, "Average INT error: %f, max INT error: %f, worst result when x: %f, y: %f\n\r",
(Int_error_summ / 15), worstIntErrorResult[2], worstIntErrorResult[0], worstIntErrorResult[1]);
                HAL_UART_Transmit(&huart1, str, strlen(str), time);

        while (1)
        {

        }
        /* USER CODE END 3 */
      }
```

Приложение 2. Консольный вывод с параметрами вычислений

```
==========Low values range ====================

x: 1, y: 0, SQRT result: 1.000000, Counts: 1958, Time: 0.000027 sec.
              FPU result: 0.960430, FPU error: 3.956997 perc., Counts: 45, Time: 0.000001 sec.
              INT result(fixed point): 245, (floatRepresent): 0.957031, INT error: 4.296875 perc., Counts: 37, Time:
0.000001 sec.
x: 2, y: 1, SQRT result: 2.236068, Counts: 2529, Time: 0.000035 sec.
              FPU result: 2.318680, FPU error: 3.694524 perc., Counts: 45, Time: 0.000001 sec.
              INT result(fixed point): 592, (floatRepresent): 2.312500, INT error: 3.418144 perc., Counts: 37, Time:
0.000001 sec.
x: 3, y: 2, SQRT result: 3.605551, Counts: 2489, Time: 0.000035 sec.
              FPU result: 3.676930, FPU error: 1.979688 perc., Counts: 45, Time: 0.000001 sec.
              INT result(fixed point): 939, (floatRepresent): 3.667969, INT error: 1.731149 perc., Counts: 37, Time:
0.000001 sec.
x: 4, y: 3, SQRT result: 5.000000, Counts: 2034, Time: 0.000028 sec.
              FPU result: 5.035180, FPU error: 0.703602 perc., Counts: 45, Time: 0.000001 sec.
              INT result(fixed point): 1286, (floatRepresent): 5.023438, INT error: 0.468750 perc., Counts: 37, Time:
0.000001 sec.
x: 5, y: 4, SQRT result: 6.403124, Counts: 2529, Time: 0.000035 sec.
              FPU result: 6.393430, FPU error: 0.151395 perc., Counts: 45, Time: 0.000001 sec.
              INT result(fixed point): 1633, (floatRepresent): 6.378906, INT error: 0.378221 perc., Counts: 37, Time:
0.000001 sec.
x: 6, y: 5, SQRT result: 7.810250, Counts: 2519, Time: 0.000035 sec.
              FPU result: 7.751680, FPU error: 0.749909 perc., Counts: 45, Time: 0.000001 sec.
              INT result(fixed point): 1980, (floatRepresent): 7.734375, INT error: 0.971476 perc., Counts: 37, Time:
0.000001 sec.
x: 7, y: 6, SQRT result: 9.219544, Counts: 2529, Time: 0.000035 sec.
              FPU result: 9.109930, FPU error: 1.188935 perc., Counts: 45, Time: 0.000001 sec.
              INT result(fixed point): 2327, (floatRepresent): 9.089844, INT error: 1.406802 perc., Counts: 37, Time:
0.000001 sec.
x: 8, y: 7, SQRT result: 10.630146, Counts: 2613, Time: 0.000036 sec.
              FPU result: 10.468180, FPU error: 1.523649 perc., Counts: 45, Time: 0.000001 sec.
              INT result(fixed point): 2674, (floatRepresent): 10.445312, INT error: 1.738766 perc., Counts: 37,
Time: 0.000001 sec.
x: 9, y: 8, SQRT result: 12.041595, Counts: 2644, Time: 0.000037 sec.
```

```
                    FPU result: 11.826430, FPU error: 1.786842 perc., Counts: 45, Time: 0.000001 sec.
                    INT result(fixed point): 3021, (floatRepresent): 11.800781, INT error: 1.999846 perc., Counts: 37,
Time: 0.000001 sec.
       x: 10, y: 9, SQRT result: 13.453624, Counts: 2617, Time: 0.000036 sec.
                    FPU result: 13.184681, FPU error: 1.999038 perc., Counts: 45, Time: 0.000001 sec.
                    INT result(fixed point): 3368, (floatRepresent): 13.156250, INT error: 2.210364 perc., Counts: 37,
Time: 0.000001 sec.
       x: 11, y: 10, SQRT result: 14.866069, Counts: 2556, Time: 0.000036 sec.
                    FPU result: 14.542931, FPU error: 2.173662 perc., Counts: 45, Time: 0.000001 sec.
                    INT result(fixed point): 3715, (floatRepresent): 14.511719, INT error: 2.383616 perc., Counts: 37,
Time: 0.000001 sec.
       x: 12, y: 11, SQRT result: 16.278821, Counts: 2514, Time: 0.000035 sec.
                    FPU result: 15.901180, FPU error: 2.319826 perc., Counts: 45, Time: 0.000001 sec.
                    INT result(fixed point): 4062, (floatRepresent): 15.867188, INT error: 2.528642 perc., Counts: 37,
Time: 0.000001 sec.
       x: 13, y: 12, SQRT result: 17.691806, Counts: 2618, Time: 0.000036 sec.
                    FPU result: 17.259430, FPU error: 2.443934 perc., Counts: 45, Time: 0.000001 sec.
                    INT result(fixed point): 4409, (floatRepresent): 17.222656, INT error: 2.651791 perc., Counts: 37,
Time: 0.000001 sec.
       x: 14, y: 13, SQRT result: 19.104973, Counts: 2563, Time: 0.000036 sec.
                    FPU result: 18.617680, FPU error: 2.550611 perc., Counts: 45, Time: 0.000001 sec.
                    INT result(fixed point): 4756, (floatRepresent): 18.578125, INT error: 2.757649 perc., Counts: 37,
Time: 0.000001 sec.
       x: 15, y: 14, SQRT result: 20.518285, Counts: 2574, Time: 0.000036 sec.
                    FPU result: 19.975929, FPU error: 2.643278 perc., Counts: 45, Time: 0.000001 sec.
                    INT result(fixed point): 5103, (floatRepresent): 19.933594, INT error: 2.849608 perc., Counts: 37,
Time: 0.000001 sec.
       Average FPU error: 1.991060, max FPU error: 3.956997, worst result when x: 1.000000, y: 0.000000
       Average INT error: 2.119447, max INT error: 4.296875, worst result when x: 1.000000, y: 0.000000
       Average SQRT time: 2485 clk = 0.000034 sec = 34 мкс              БУДЕТ ИЗМЕРЯТЬСЯ ДОЛЯМИ МИКРОСЕКУНД
       Average FPU time: 45 clk = 625 нс
       Average INT time: 37 clk = 513 нс


       ============================ Mid values range ==============================

       x: 112, y: 0, SQRT result: 112.000000, Counts: 2037, Time: 0.000028 sec.
                    FPU result: 107.568161, FPU error: 3.956999 perc., Counts: 42, Time: 0.000001 sec.
                    INT result(fixed point): 27440, (floatRepresent): 107.187500, INT error: 4.296875 perc., Counts: 38,
Time: 0.000001 sec.
       x: 265, y: 132, SQRT result: 296.055738, Counts: 2590, Time: 0.000036 sec.
                    FPU result: 307.026184, FPU error: 3.705534 perc., Counts: 42, Time: 0.000001 sec.
                    INT result(fixed point): 78389, (floatRepresent): 306.207031, INT error: 3.428845 perc., Counts: 38,
Time: 0.000001 sec.
       x: 354, y: 288, SQRT result: 456.355125, Counts: 2614, Time: 0.000036 sec.
                    FPU result: 454.564392, FPU error: 0.392399 perc., Counts: 42, Time: 0.000001 sec.
                    INT result(fixed point): 116106, (floatRepresent): 453.539062, INT error: 0.617077 perc., Counts: 38,
Time: 0.000001 sec.
       x: 415, y: 354, SQRT result: 545.473189, Counts: 2597, Time: 0.000036 sec.
                    FPU result: 539.406738, FPU error: 1.112145 perc., Counts: 42, Time: 0.000001 sec.
                    INT result(fixed point): 137783, (floatRepresent): 538.214844, INT error: 1.330651 perc., Counts: 38,
Time: 0.000001 sec.
       x: 501, y: 422, SQRT result: 655.045800, Counts: 2604, Time: 0.000036 sec.
                    FPU result: 649.055481, FPU error: 0.914489 perc., Counts: 42, Time: 0.000001 sec.
                    INT result(fixed point): 165789, (floatRepresent): 647.613281, INT error: 1.134656 perc., Counts: 38,
Time: 0.000001 sec.
       x: 610, y: 506, SQRT result: 792.550314, Counts: 2544, Time: 0.000035 sec.
                    FPU result: 787.159241, FPU error: 0.680218 perc., Counts: 42, Time: 0.000001 sec.
                    INT result(fixed point): 201062, (floatRepresent): 785.398438, INT error: 0.902388 perc., Counts: 38,
Time: 0.000001 sec.
       x: 771, y: 643, SQRT result: 1003.937249, Counts: 2585, Time: 0.000036 sec.
                    FPU result: 996.289856, FPU error: 0.761740 perc., Counts: 42, Time: 0.000001 sec.
                    INT result(fixed point): 254481, (floatRepresent): 994.066406, INT error: 0.983213 perc., Counts: 38,
Time: 0.000001 sec.
       x: 810, y: 754, SQRT result: 1106.623694, Counts: 2523, Time: 0.000035 sec.
                    FPU result: 1077.904541, FPU error: 2.595205 perc., Counts: 42, Time: 0.000001 sec.
                    INT result(fixed point): 275358, (floatRepresent): 1075.617188, INT error: 2.801902 perc., Counts: 38,
Time: 0.000001 sec.
       x: 941, y: 865, SQRT result: 1278.165091, Counts: 2540, Time: 0.000035 sec.
                    FPU result: 1247.878906, FPU error: 2.369505 perc., Counts: 42, Time: 0.000001 sec.
                    INT result(fixed point): 318775, (floatRepresent): 1245.214844, INT error: 2.577934 perc., Counts: 38,
Time: 0.000001 sec.
       x: 1022, y: 900, SQRT result: 1361.794404, Counts: 2576, Time: 0.000036 sec.
                    FPU result: 1339.597534, FPU error: 1.629972 perc., Counts: 42, Time: 0.000001 sec.
                    INT result(fixed point): 342190, (floatRepresent): 1336.679688, INT error: 1.844237 perc., Counts: 38,
Time: 0.000001 sec.
       x: 1103, y: 1090, SQRT result: 1550.712417, Counts: 2575, Time: 0.000036 sec.
                    FPU result: 1492.978149, FPU error: 3.723080 perc., Counts: 42, Time: 0.000001 sec.
                    INT result(fixed point): 381415, (floatRepresent): 1489.902344, INT error: 3.921428 perc., Counts: 38,
Time: 0.000001 sec.
       x: 1220, y: 1103, SQRT result: 1644.691156, Counts: 2531, Time: 0.000035 sec.
                    FPU result: 1610.520020, FPU error: 2.077663 perc., Counts: 42, Time: 0.000001 sec.
                    INT result(fixed point): 411406, (floatRepresent): 1607.054688, INT error: 2.288361 perc., Counts: 38,
Time: 0.000001 sec.
       x: 1360, y: 1207, SQRT result: 1818.364375, Counts: 2588, Time: 0.000036 sec.
                    FPU result: 1786.353516, FPU error: 1.760421 perc., Counts: 42, Time: 0.000001 sec.
```

```
                    INT result(fixed point): 456314, (floatRepresent): 1782.476562, INT error: 1.973632 perc., Counts: 38,
Time: 0.000001 sec.
        x: 1400, y: 1301, SQRT result: 1911.177909, Counts: 2616, Time: 0.000036 sec.
                    FPU result: 1862.165894, FPU error: 2.564492 perc., Counts: 42, Time: 0.000001 sec.
                    INT result(fixed point): 475702, (floatRepresent): 1858.210938, INT error: 2.771431 perc., Counts: 38,
Time: 0.000001 sec.
        x: 1503, y: 1420, SQRT result: 2067.706217, Counts: 2583, Time: 0.000036 sec.
                    FPU result: 2008.430786, FPU error: 2.866724 perc., Counts: 42, Time: 0.000001 sec.
                    INT result(fixed point): 513075, (floatRepresent): 2004.199219, INT error: 3.071374 perc., Counts: 38,
Time: 0.000001 sec.
        Average FPU error: 2.074039, max FPU error: 3.956999, worst result when x: 112.000000, y: 0.000000
        Average INT error: 2.262933, max INT error: 4.296875, worst result when x: 112.000000, y: 0.000000



        ============================ High values range ==============================
        x: 1010, y: 0, SQRT result: 1010.000000, Counts: 2037, Time: 0.000028 sec.
                    FPU result: 970.034302, FPU error: 3.957000 perc., Counts: 42, Time: 0.000001 sec.
                    INT result(fixed point): 247450, (floatRepresent): 966.601562, INT error: 4.296875 perc., Counts: 38,
Time: 0.000001 sec.
        x: 2120, y: 1060, SQRT result: 2370.232056, Counts: 2499, Time: 0.000035 sec.
                    FPU result: 2457.800781, FPU error: 3.694521 perc., Counts: 42, Time: 0.000001 sec.
                    INT result(fixed point): 627520, (floatRepresent): 2451.250000, INT error: 3.418144 perc., Counts: 38,
Time: 0.000001 sec.
        x: 3165, y: 2998, SQRT result: 4359.498710, Counts: 2579, Time: 0.000036 sec.
                    FPU result: 4232.425293, FPU error: 2.914863 perc., Counts: 42, Time: 0.000001 sec.
                    INT result(fixed point): 1081221, (floatRepresent): 4223.519531, INT error: 3.119147 perc., Counts: 38,
Time: 0.000001 sec.
        x: 4015, y: 3658, SQRT result: 5431.499701, Counts: 2605, Time: 0.000036 sec.
                    FPU result: 5311.352051, FPU error: 2.212053 perc., Counts: 42, Time: 0.000001 sec.
                    INT result(fixed point): 1356791, (floatRepresent): 5299.964844, INT error: 2.421704 perc., Counts: 38,
Time: 0.000001 sec.
        x: 5986, y: 4031, SQRT result: 7216.727582, Counts: 2594, Time: 0.000036 sec.
                    FPU result: 7352.746582, FPU error: 1.884774 perc., Counts: 42, Time: 0.000001 sec.
                    INT result(fixed point): 1877732, (floatRepresent): 7334.890625, INT error: 1.637349 perc., Counts: 38,
Time: 0.000001 sec.
        x: 6887, y: 5132, SQRT result: 8588.841191, Counts: 2668, Time: 0.000037 sec.
                    FPU result: 8656.093750, FPU error: 0.783023 perc., Counts: 42, Time: 0.000001 sec.
                    INT result(fixed point): 2210779, (floatRepresent): 8635.855469, INT error: 0.547388 perc., Counts: 38,
Time: 0.000001 sec.
        x: 7650, y: 6010, SQRT result: 9728.442835, Counts: 2635, Time: 0.000037 sec.
                    FPU result: 9738.187500, FPU error: 0.100167 perc., Counts: 42, Time: 0.000001 sec.
                    INT result(fixed point): 2487270, (floatRepresent): 9715.898438, INT error: 0.128946 perc., Counts: 38,
Time: 0.000001 sec.
        x: 8117, y: 7991, SQRT result: 11390.424487, Counts: 2606, Time: 0.000036 sec.
                    FPU result: 10974.790039, FPU error: 3.648981 perc., Counts: 42, Time: 0.000001 sec.
                    INT result(fixed point): 2803747, (floatRepresent): 10952.136719, INT error: 3.847862 perc., Counts:
38, Time: 0.000001 sec.
        x: 9010, y: 8540, SQRT result: 12414.173351, Counts: 2573, Time: 0.000036 sec.
                    FPU result: 12050.857422, FPU error: 2.926622 perc., Counts: 42, Time: 0.000001 sec.
                    INT result(fixed point): 3078530, (floatRepresent): 12025.507812, INT error: 3.130821 perc., Counts:
38, Time: 0.000001 sec.
        x: 10100, y: 9020, SQRT result: 13541.432716, Counts: 2613, Time: 0.000036 sec.
                    FPU result: 13288.679688, FPU error: 1.866516 perc., Counts: 42, Time: 0.000001 sec.
                    INT result(fixed point): 3394540, (floatRepresent): 13259.921875, INT error: 2.078885 perc., Counts:
38, Time: 0.000001 sec.
        x: 11620, y: 10010, SQRT result: 15337.030351, Counts: 2586, Time: 0.000036 sec.
                    FPU result: 15142.375000, FPU error: 1.269185 perc., Counts: 42, Time: 0.000001 sec.
                    INT result(fixed point): 3867920, (floatRepresent): 15109.062500, INT error: 1.486388 perc., Counts:
38, Time: 0.000001 sec.
        x: 12001, y: 11022, SQRT result: 16294.431104, Counts: 2602, Time: 0.000036 sec.
                    FPU result: 15910.892578, FPU error: 2.353801 perc., Counts: 42, Time: 0.000001 sec.
                    INT result(fixed point): 4064489, (floatRepresent): 15876.910156, INT error: 2.562354 perc., Counts:
38, Time: 0.000001 sec.
        x: 13041, y: 12001, SQRT result: 17722.631915, Counts: 2520, Time: 0.000035 sec.
                    FPU result: 17299.205078, FPU error: 2.389187 perc., Counts: 42, Time: 0.000001 sec.
                    INT result(fixed point): 4419147, (floatRepresent): 17262.292969, INT error: 2.597464 perc., Counts:
38, Time: 0.000001 sec.
        x: 14099, y: 13700, SQRT result: 19658.886057, Counts: 2552, Time: 0.000035 sec.
                    FPU result: 18991.236328, FPU error: 3.396173 perc., Counts: 42, Time: 0.000001 sec.
                    INT result(fixed point): 4851655, (floatRepresent): 18951.777344, INT error: 3.596891 perc., Counts:
38, Time: 0.000001 sec.
        x: 15001, y: 14000, SQRT result: 20519.015595, Counts: 2661, Time: 0.000037 sec.
                    FPU result: 19976.890625, FPU error: 2.642061 perc., Counts: 42, Time: 0.000001 sec.
                    INT result(fixed point): 5103245, (floatRepresent): 19934.550781, INT error: 2.848406 perc., Counts:
38, Time: 0.000001 sec.
        Average FPU error: 2.402595, max FPU error: 3.957000, worst result when x: 1010.000000, y: 0.000000
        Average INT error: 2.514575, max INT error: 4.296875, worst result when x: 1010.000000, y: 0.000000



        ============================ Unround values range ============================

        x: 15251, y: 0, SQRT result: 15251.000000, Counts: 2036, Time: 0.000028 sec.
                    FPU result: 14647.518555, FPU error: 3.956996 perc., Counts: 43, Time: 0.000001 sec.
                    INT result(fixed point): 3736495, (floatRepresent): 14595.683594, INT error: 4.296875 perc., Counts:
39, Time: 0.000001 sec.
```

```
        x: 26152, y: 17311, SQRT result: 31362.363192, Counts: 2560, Time: 0.000036 sec.
                        FPU result: 32003.828125, FPU error: 2.045334 perc., Counts: 43, Time: 0.000001 sec.
                        INT result(fixed point): 8172962, (floatRepresent): 31925.632812, INT error: 1.796005 perc., Counts:
39, Time: 0.000001 sec.
        x: 39575, y: 2334, SQRT result: 39643.765979, Counts: 2578, Time: 0.000036 sec.
                        FPU result: 38937.531250, FPU error: 1.781452 perc., Counts: 43, Time: 0.000001 sec.
                        INT result(fixed point): 9933943, (floatRepresent): 38804.464844, INT error: 2.117107 perc., Counts:
39, Time: 0.000001 sec.
        x: 46111, y: 3864, SQRT result: 46272.614115, Counts: 2573, Time: 0.000036 sec.
                        FPU result: 45823.566406, FPU error: 0.970439 perc., Counts: 43, Time: 0.000001 sec.
                        INT result(fixed point): 11691323, (floatRepresent): 45669.230469, INT error: 1.303976 perc., Counts:
39, Time: 0.000001 sec.
        x: 59587, y: 472, SQRT result: 59588.869372, Counts: 2568, Time: 0.000036 sec.
                        FPU result: 57416.914062, FPU error: 3.644901 perc., Counts: 43, Time: 0.000001 sec.
                        INT result(fixed point): 14646959, (floatRepresent): 57214.683594, INT error: 3.984277 perc., Counts:
39, Time: 0.000001 sec.
        x: 67764, y: 58722, SQRT result: 89667.346230, Counts: 2638, Time: 0.000037 sec.
                        FPU result: 88443.367188, FPU error: 1.365022 perc., Counts: 43, Time: 0.000001 sec.
                        INT result(fixed point): 22591824, (floatRepresent): 88249.312500, INT error: 1.581438 perc., Counts:
39, Time: 0.000001 sec.
        x: 77181, y: 6152, SQRT result: 77425.795863, Counts: 2658, Time: 0.000037 sec.
                        FPU result: 76574.343750, FPU error: 1.099701 perc., Counts: 43, Time: 0.000001 sec.
                        INT result(fixed point): 19536849, (floatRepresent): 76315.812500, INT error: 1.433609 perc., Counts:
39, Time: 0.000001 sec.
        x: 81522, y: 79901, SQRT result: 114149.052931, Counts: 2624, Time: 0.000036 sec.
                        FPU result: 110082.390625, FPU error: 3.562590 perc., Counts: 43, Time: 0.000001 sec.
                        INT result(fixed point): 28122792, (floatRepresent): 109854.656250, INT error: 3.762096 perc., Counts:
39, Time: 0.000001 sec.
        x: 91554, y: 81315, SQRT result: 122451.068354, Counts: 2639, Time: 0.000037 sec.
                        FPU result: 120279.945312, FPU error: 1.773054 perc., Counts: 43, Time: 0.000001 sec.
                        INT result(fixed point): 30724860, (floatRepresent): 120018.984375, INT error: 1.986168 perc., Counts:
39, Time: 0.000001 sec.
        x: 105613, y: 98001, SQRT result: 144077.415892, Counts: 2642, Time: 0.000037 sec.
                        FPU result: 140420.656250, FPU error: 2.538052 perc., Counts: 43, Time: 0.000001 sec.
                        INT result(fixed point): 35871287, (floatRepresent): 140122.218750, INT error: 2.745189 perc., Counts:
39, Time: 0.000001 sec.
        x: 114536, y: 105492, SQRT result: 155714.666490, Counts: 2658, Time: 0.000037 sec.
                        FPU result: 151970.640625, FPU error: 2.404414 perc., Counts: 43, Time: 0.000001 sec.
                        INT result(fixed point): 38821504, (floatRepresent): 151646.500000, INT error: 2.612578 perc., Counts:
39, Time: 0.000001 sec.
        x: 128743, y: 11365, SQRT result: 129243.658545, Counts: 2561, Time: 0.000036 sec.
                        FPU result: 128169.867188, FPU error: 0.830827 perc., Counts: 43, Time: 0.000001 sec.
                        INT result(fixed point): 32701265, (floatRepresent): 127739.312500, INT error: 1.163961 perc., Counts:
39, Time: 0.000001 sec.
        x: 136778, y: 122689, SQRT result: 183741.154903, Counts: 2658, Time: 0.000037 sec.
                        FPU result: 180173.843750, FPU error: 1.941487 perc., Counts: 43, Time: 0.000001 sec.
                        INT result(fixed point): 46024888, (floatRepresent): 179784.718750, INT error: 2.153266 perc., Counts:
39, Time: 0.000001 sec.
        x: 145413, y: 139223, SQRT result: 201315.633516, Counts: 2599, Time: 0.000036 sec.
                        FPU result: 195044.703125, FPU error: 3.114974 perc., Counts: 43, Time: 0.000001 sec.
                        INT result(fixed point): 49826931, (floatRepresent): 194636.453125, INT error: 3.317765 perc., Counts:
39, Time: 0.000001 sec.
        x: 154449, y: 148333, SQRT result: 214142.878682, Counts: 2587, Time: 0.000036 sec.
                        FPU result: 207347.281250, FPU error: 3.173394 perc., Counts: 43, Time: 0.000001 sec.
                        INT result(fixed point): 52969971, (floatRepresent): 206913.953125, INT error: 3.375749 perc., Counts:
39, Time: 0.000001 sec.
        Average FPU error: 2.280176, max FPU error: 3.956996, worst result when x: 15251.000000, y: 0.000000
        Average INT error: 2.508671, max INT error: 4.296875, worst result when x: 15251.000000, y: 0.000000
        Average SQRT time: clk 2571= 35 мкс
        Average FPU time: 43 clk = 597 нс
        Average INT time: 39 clk = 541 нс
```

ИНФОРМАЦИЯ ОБ АВТОРАХ

Корытцев Иван Витальевич
Студент
Санкт-Петербургский государственный университет аэрокосмического приборостроения
Россия, 190000, Санкт-Петербург, ул. Большая Морская, д. 67, лит. А
E-mail: johnmacray@internet.ru

Док Цихон Гордон Франц Эмануэль
Доцент кафедры аэрокосмических компьютерных и программных систем
Санкт-Петербургский государственный университет аэрокосмического приборостроения
Россия, 190000, Санкт-Петербург, ул. Большая Морская, д. 67, лит. А
E-mail: gordon.cichon@guap.ru

INFORMATION ABOUT THE AUTHORS

Koryttsev Ivan Vitalievich
Student
Saint-Petersburg State University of Aerospace Instrumentation
67, Bolshaya Morskaia str., Saint-Petersburg, 190000, Russia
E-mail: johnmacray@internet.ru

Gordon Cichon
PhD, Associate Professor, Department “Aerospace Computer and Software Systems”
Saint-Petersburg State University of Aerospace Instrumentation
67, Bolshaya Morskaia str., Saint-Petersburg, 190000, Russia
E-mail: gordon.cichon@guap.ru

Дата поступления: 09.03.2026
Дата принятия: 17.03.2026