УТВЕРЖДАЮ

М.К. Семёнов

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

“\_\_” \_\_\_\_\_\_\_2022г.

Система арифметических операций в массиве

наименование вида АС

СибГУ им. М.Ф. Решетнёва

наименование объекта автоматизации

САО массива

сокращенное наименование АС

ТЕХНИЧЕСКАЯ ДОКУМЕНТАЦИЯ

На 23 листах

Действует с \_\_. \_\_ 2022г.

СОГЛАСОВАНО

М.К. Семёнов

“\_\_” \_\_\_\_\_\_\_ 2022г.

Оглавление

[1. Общие сведения 3](#_Toc122477248)

[1.1 Полное наименование системы и ее условное обозначение. 4](#_Toc122477249)

[1.2 Наименование разработчика системы и реквизиты заказчика. 4](#_Toc122477250)

[1.3. Основания для разработки ПО. 4](#_Toc122477251)

[1.4. Плановые сроки начала и окончания работы по созданию системы: 4](#_Toc122477252)

[1.5. Источник финансирования работ по созданию АС. 4](#_Toc122477253)

[2. Назначение и цели создания системы 3](#_Toc122477254)

[2.1 Назначение системы. 3](#_Toc122477255)

[2.2 Цели создания системы. 3](#_Toc122477256)

[3. Требования к системе 3](#_Toc122477257)

[3.1. Требования к системе в целом. 3](#_Toc122477258)

[3.1.1. Требования к структуре и функционированию системы 3](#_Toc122477259)

[3.1.2. Требования к средствам и способам связи для информационного обмена между компонентами системы. 3](#_Toc122477260)

[3.1.3. Требования к характеристикам взаимосвязи создаваемой системы со смежными системами, требования к ее совместимости. 4](#_Toc122477261)

[3.1.4. Требования по диагностированию системы. 4](#_Toc122477262)

[3.1.5. Перспективы ПО, модернизация ПО. 4](#_Toc122477263)

[3.1.6. Требования к надежности комплекса. 4](#_Toc122477264)

[3.1.7 Требования к квалификации пользователя 4](#_Toc122477265)

[3.1.8. Требования по стандартизации и унификации. 4](#_Toc122477266)

[3.2. Требования к задачам, выполняемым системой. 4](#_Toc122477267)

[3.2.1 Перечень функций, подлежащих автоматизации: 4](#_Toc122477268)

[3.3. Требования к видам обеспечения. 5](#_Toc122477269)

[3.3.1. Требования к программному обеспечению. 5](#_Toc122477270)

[3.3.2. Требования к техническому обеспечению. 5](#_Toc122477271)

[4. Тестирование программного обеспечения 6](#_Toc122477272)

[4.1. Тестирование пределов программы: 6](#_Toc122477273)

[4.2 Конфигурация рабочей машины1 8](#_Toc122477274)

[4.3 Конфигурация рабочей машины2 11](#_Toc122477275)

[5. Результаты тестов и выводы 14](#_Toc122477276)

[5.1 Различия в типах переменных 14](#_Toc122477277)

[5.2 Результаты исследования массивов с малыми числами 14](#_Toc122477278)

[5.3 Аппаратные ограничения 14](#_Toc122477279)

[5.4 Роль импортируемых типов 15](#_Toc122477280)

[6.Техническое руководство 15](#_Toc122477281)

[6.1 Создание массива размером L [x1,x2]: 16](#_Toc122477282)

[6.2 Выбор типа массива: 17](#_Toc122477283)

[6.4. Выполнение арифметических операций над элементами массива: 17](#_Toc122477284)

[6.5 Алгоритм работы функции “Автотест” 17](#_Toc122477285)

[6.6 Алгоритм работы подпрограммы “Монитор ресурсов”: 17](#_Toc122477286)

[6.7 UML диаграммы разработчика 17](#_Toc122477287)

# Общие сведения

## 1.1 Полное наименование системы и ее условное обозначение.

Система арифметических операций в массиве.

Условное обозначение: CAO массива

## 1.2 Наименование разработчика системы и реквизиты заказчика.

Заказчик – Семёнов Михаил Константинович

Разработчик – студент группы БПИ22-02 Трифонов Кирилл Вячеславович

## 1.3. Основания для разработки ПО.

Работа по созданию системы арифметических операций в массиве

## 1.4. Плановые сроки начала и окончания работы по созданию системы:

- начало работ по созданию системы – 10 ноября 2022

- окончание работ по созданию системы – 5 декабря 202

## 1.5. Источник финансирования работ по созданию АС.

Отсутствует.

# 2. Назначение и цели создания системы

## 2.1 Назначение системы.

САО массива предназначена для генерации массивов, размером заданных переменной N, с диапазоном генерации [x1; x2] и для проведения таких арифметических операций, как сложение, вычитание, умножение, деление. Программа должна выводить время выполнения операций.

## 2.2 Цели создания системы.

Целью создания системы является:  
Задание, данное преподавателем дисциплины “Основы программной инженерии” и получение новых навыков в дисциплине.

# 3. Требования к системе

## 3.1. Требования к системе в целом.

### 3.1.1. Требования к структуре и функционированию системы

САО массива должна включать в себя следующий функционал:

* создание массива с размером n
* выбор типа массива (целочисленный, дробный)
* генерация элементов массива с диапазоном [a;b]
* выполнение арифметических операций над элементами массива

1. **Создание массива с размером n:**

Переменная l задаётся пользователем в окно интерфейса;

1. **Выбор типа массива (целочисленный, дробный)**

Тип элементов задается выбором из списка (int, long, short, int8, int16, int32, int64, longlong, ulonglong, float, float16, float32, float64, bin);

В соответствии с выбранным типом, создаётся определённый массив на l элементов;

1. **Генерация элементов массива с диапазоном [x1; x2]:**

Переменные x1 и x2 задаются пользователем в соответствующие окна интерфейса “от” и “до”;

1. **Выполнение арифметических операций над элементами массива:**

Пользователь может выбрать 6 типа операций;

Арифметические операции должны быть выполнены над всеми элементами массива в соответствии с выбранной операцией

Программа подсчитывает время выполнения операции

### 3.1.2. Требования к средствам и способам связи для информационного обмена между компонентами системы.

Требования к средствам и способам связи отсутствуют.

### 3.1.3. Требования к характеристикам взаимосвязи создаваемой системы со смежными системами, требования к ее совместимости.

Требования к характеристикам взаимосвязи создаваемой системы отсутствуют.

### 3.1.4. Требования по диагностированию системы.

Проверка программного и аппаратного обеспечения проводится по мере необходимости.

### 3.1.5. Перспективы ПО, модернизация ПО.

Код САО может быть перенесён с языка python на более низкоуровневый c++, с целью повышения производительности. Также в перспективах параллельная работа подпрограмм САО, таких как “Заполнить”, “Вычислить”,”Удалить”,”Автотест”. Это позволит в любое время прервать процесс не закрывая программу, одновременно заполнять более одного массива, обновлять монитор ресурсов в реальном времени (сейчас он актуален только на момент запуска и окончания программы).

### 3.1.6. Требования к надежности комплекса.

Необходимо, чтобы система обладала устойчивостью к отказам оборудования и программных систем, а также электропитания. Для надежной работы комплекса необходимы высоконадежные аппаратные и программные системы. Требования надежности должны быть регламентированы для следующих аварийных ситуаций:

* выход из строя аппаратных средств системы;
* отсутствие электроэнергии;
* выход из строя программных средств системы;
* неверные действия персонала компании;
* пожар, взрыв и т.п.

Методы оценки и контроля показателей надежности на разных стадиях создания системы должны отвечать следующим особенностям:

* многофункциональность;
* разнообразные законы распределения среднего времени безотказной работы и восстановления.

### 3.1.7 Требования к квалификации пользователя

Квалификация пользователя программы:  
Пользователь программы должен владеть навыками работы с операционной системой Microsoft Windows 7/8/8.1/10/11.

### 3.1.8. Требования по стандартизации и унификации.

Массив и результаты вычислений содержатся исключительно в оперативной памяти, то есть не сохраняются.

## 3.2. Требования к задачам, выполняемым системой.

### Перечень функций, подлежащих автоматизации:

1. *Создание массива размером n:*

Массив создаётся путём выделения динамической памяти компьютера. Данный способ позволяет пользователю самому задать необходимый ему размера массива.

1. *Выбор типа массива:*

Пользователь может выбрать тип создаваемого массива и его элементов. Для каждого типа создаётся свой уникальный массив со своими уникальными функциями.

1. *Генерация элементов массива с диапазоном [x1; x2]*

Пользователь сам выбирает границы диапазона генерации элементов массива путём ввода переменных a и b. Стоит отметить, что числа a и b включены в диапазон генерации.

1. *Выполнение арифметических операций над элементами массива:*

Пользователю предлагается выбор из шести арифметических операций. Каждая операция выполняется при помощи циклов, которые затрагивают каждый элемент массива.

## 3.3. Требования к видам обеспечения.

### 3.3.1. Требования к программному обеспечению.

Для работы САО массива необходимо следующее ПО:

1. На рабочей машине должны быть установлены:

• Операционная система: Microsoft Windows 7/8/8.1/10/11.

### 3.3.2. Требования к техническому обеспечению.

1. процессор Intel Core i3-i5 или AMD Ryzen 3;
2. 4 ГБ и более оперативной памяти (для создания массивов весом 1 ГБ требуется примерно 3 ГБ свободной памяти);
3. 1 ГБ свободной постоянной памяти
4. Видеокарта с 512мБ и более видеопамяти;
5. Монитор (устройство вывода изображения);
6. Клавиатура;
7. Манипулятор типа «мышь».

# 4. Тестирование программного обеспечения

## 4.1. Тестирование пределов программы:

*Таблица представленных в программе типов:*

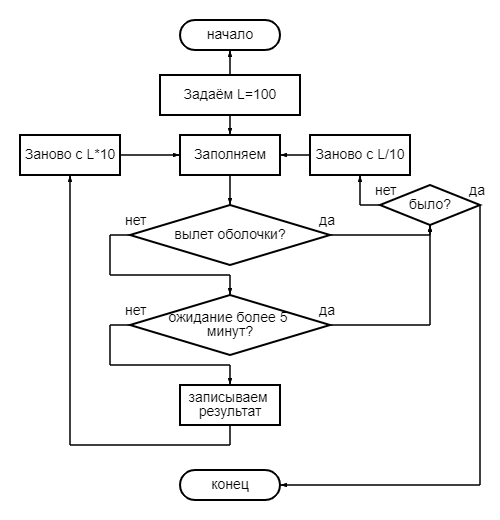
|  |  |  |
| --- | --- | --- |
| **Инфо о типах данных** | | |
| **Тип данных** | **Вес переменной, бит** | **Способ создания** |
|
| int | *Динамический* | Стандартный for |
| long | 32 | numpy |
| short | 16 | numpy |
| int8 | 8 | numpy |
| int16 | 16 | numpy |
| int32 | 32 | numpy |
| int64 | 64 | numpy |
| longlong | 64 | numpy |
| ulonglong | 64 | numpy |
| float | *Динамический* | Стандартный for |
| float16 | 16 | Стандартный for |
| float32 | 32 | Стандартный for |
| float64 | 64 | numpy |
| bin | 2 | numpy |

В таблице ниже представлен вес каждого из типов при различной длинне массива:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Тип данных** | **Занимаемый объём памяти массивом при разной длинне** | | | | | |
| 10тыс, KB | 100тыс, KB | 1млн, MB | 100млн, MB | 1млрд, MB | 2млрд, MB |
| int | 83,18 | 782,2 | 8,057 | 796.4 | 7964 | 15928 |
| long | 39,17 | 390,7 | 3,815 | 381.4 | 3814 | 7628 |
| short | 19,64 | 195,4 | 1,907 | 190.7 | 1907 | 3814 |
| int8 | 9,875 | 97,76 | 0,976 | 95.36 | 983 | 1907 |
| int16 | 19,64 | 195,4 | 1,907 | 190.7 | 1907 | 3814 |
| int32 | 39,17 | 390,7 | 3,815 | 381.4 | 3814 | 7628 |
| int64 | 78,23 | 781,3 | 7,63 | 762.9 | 7628 | 15256 |
| loglong | 78,23 | 781,3 | 7,63 | 762.9 | 7628 | 15256 |
| ulonglong | 78,23 | 781,3 | 7,63 | 762.9 | 7628 | 15256 |
| float | 78,18 | 781,3 | 7,63 | 762.9 | 7629 | 15256 |
| float16 | 19,64 | 195,4 | 1.907 | 190.7 | Долго(<ОЗУ) | Долго(<ОЗУ) |
| float32 | 39,17 | 390,7 | 3.815 | 381.5 | Долго(<ОЗУ) | Долго(<ОЗУ) |
| float64 | 78,23 | 781,3 | 7,63 | 762.9 | Долго(<ОЗУ) | Долго(<ОЗУ) |
| bin | 9,875 | 97,76 | 0,976 | 95.36 | Долго | Долго |

В таблице ниже представлено время умножения разных типов при разной длинне массива:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Тип данных | Время умножения при L=… | | | | |  |
| 20 | 40 | 80 | 120 | 250 | 1000 |
| int | <1 | <1 | <1 | <1 | <1 | <1 |
| long | Переполнение | Переполнение | Переполнение | Переполнение | Переполнение | Переполнение |
| short | Переполнение | Переполнение | Переполнение | Переполнение | Переполнение | Переполнение |
| int8 | Переполнение | Переполнение | Переполнение | Переполнение | Переполнение | Переполнение |
| int16 | <1 | Переполнение | Переполнение | Переполнение | Переполнение | Переполнение |
| int32 | <1 | Переполнение | Переполнение | Переполнение | Переполнение | Переполнение |
| int64 | <1 | <1 | Переполнение | Переполнение | Переполнение | Переполнение |
| longlong | <1 | <1 | Переполнение | Переполнение | Переполнение | Переполнение |
| ulonglong | <1 | <1 | <1 | <1 | <1 | Переполнение |
| float | <1 | <1 | <1 | <1 | <1 | Переполнение |
| float16 | <1 | <1 | <1 | <1 | <1 | Переполнение |
| float32 | 1 | <1 | <1 | <1 | <1 | Переполнение |
| float64 | <1 | <1 | <1 | <1 | <1 | Переполнение |
| bin |  |  |  |  |  |  |



*Алгоритм тестирования*

## 4.2 Конфигурация рабочей машины1

|  |  |
| --- | --- |
| Процессор | Quad-core AMD Ryzen 5 3500u @ 3700 mhz |
| Оперативная память | 8 ГБ ddr4 2400 mhz (Из них доступно 6 ГБ, из-за использования iGPU) |
| Накопитель | SSD nvme 240 GB |
| Тип системы | 64-разрядная операционная система, процессор x64 |
| Видеоадаптер | AMD radeon vega 8 graphics |

Результаты тестирования при длине массива равной 1тыс:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Тестирование** | | | |
| **Тип данных** | **Диапазон** | **Время выполнения, ms** | | |
| **Сложение** | **Умножение** | **Умножить на 5** |
| int | 1 до 2 | <1 | 30 | <1 |
| long | 1 до 2 | 1 | 30 | 31 |
| short | 1 до 2 | 2 | 29 | 67 |
| int8 | 1 до 2 | 1 | 31 | 53 |
| int16 | 1 до 2 | 1 | 30 | 53 |
| int32 | 1 до 2 | 1 | 30 | 32 |
| int64 | 1 до 2 | 1 | 30 | 30 |
| longlong | 1 до 2 | 2 | 30 | 49 |
| ulonglong | 1 до 2 | 2 | 30 | 49 |
| float | 1 до 2 | <1 | 30 | 27 |
| Decimal (str) | 1 до 2 | 30 | 79 | 353 |
| float16 | 1 до 2 | 1 | 30 | 30 |
| float32 | 1 до 2 | 1 | 30 | 49 |
| float64 | 1 до 2 | 2 | 30 | 49 |
| bin | 1 до 2 | 1 |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Тип данных** | **Вес массива, MB** | **Время заполнения, ms** | **Диапазон** | **Время выполнения, ms** | | |
| **Сложение/Вычитание** | **Умножение/деление** | **Умножить/возвести на 5** |
| int | 8,057 | 381 | 1 до 5 | 11 | Сбой | 5622 |
| long | 3,815 | 51 | 1 до 5 | 133 | Переполнение | 6952 |
| short | 1,907 | 41 | 1 до 5 | 230 | Переполнение | 11103 |
| int8 | 0,953 | 36 | 1 до 5 | 135 | Переполнение | 11037 |
| int16 | 1,907 | 43 | 1 до 5 | 225 | Переполнение | 11430 |
| int32 | 3,815 | 47 | 1 до 5 | 139 | Переполнение | 8390 |
| int64 | 7,63 | 45 | 1 до 5 | 134 | Переполнение | 10624 |
| longlong | 7,63 | 49 | 1 до 5 | 147 | Переполнение | 6367 |
| Ulonglong | 7,63 | 52 | 1 до 5 | 251 | Переполнение | 10976 |
| float | 7,63 | 378 | 0.999 -1.001 | 16 | 1575 | 9748 |
| Decimal (str) | 7,63 | 320544 | 0.999 -1.001 | Сбой | Сбой | Сбой |
| float16 | 1,907 | 420 | 0.999 -1.001 | 252 | 2193 | 10829 |
| float32 | 3,815 | 420 | 0.999 -1.001 | 240 | 2131 | 10882 |
| float64 | 7,63 | 38 | 0.999 -1.001 | 151 | 1950 | 6715 |
| bin | 0,953 | 60 | 1 до 5 | 170 | Переполнение |  |

Результаты тестирования при длине массива равной 1млн:

Результаты тестирования заполнения массива имеющимися типами при разной длинне:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Время заполнения** | | | | | |
| **Тип данных** | 10тыс, мс | 100тыс, мс | 1млн, мс | 100млн, мс | 1млрд, мс | 2млрд, мс |
| int | 16 | 62 | 8461 | 61827 | Долго | Долго |
| long | 8 | 20 | 32 | 1738 | 36042 | Долго |
| short | 5 | 16 | 47 | 1206 | 29300 | 79320 |
| int8 | 1 | 9 | 31 | 1056 | 28047 | 83168 |
| int16 | 3 | 9 | 32 | 1298 | 29300 | 79100 |
| int32 | 3 | 5 | 32 | 1591 | 35039 | Долго |
| int64 | 5 | 6 | 33 | 1584 | 47066 | Долго |
| longlong | 4 | 5 | 47 | 1607 | 45468 | Долго |
| Ulonglong | 6 | 5 | 47 | 1607 | 44879 | Долго |
| float | 25 | 195 | 1820 | Долго | Долго | Долго |
| Decimal | 13568 | Долго | Долго | Долго | Долго | Долго |
| float16 | 23 | 191 | 1921 | Долго | Долго | Долго |
| float32 | 23 | 204 | 1926 | Долго | Долго | Долго |
| float64 | 53 | 4 | 17 | Долго | Долго | Долго |
| bin | 26 | 8 | 36 | Долго | Долго | Долго |

Результаты более подробно отражены в диаграмме:

Примерное место нехватки ОЗУ

## 4.3 Конфигурация рабочей машины2

|  |  |
| --- | --- |
| Процессор | Six-core AMD Ryzen 5 2600 @ 3200 mhz |
| Оперативная память | 16 ГБ ddr4 3600 mhz |
| Накопитель | SSD sata 480 GB |
| Тип системы | 64-разрядная операционная система, процессор x64 |
| Видеоадаптер | Nvidia GeForce RTX 2060 |

Результаты тестирования при длине массива равной 1тыс:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Тестирование** | | | |
| **Тип данных** | **Диапазон** | **Время выполнения, ms** | | |
| **Сложение** | **Умножение** | **Умножить на 5** |
| int | 1 до 2 | <1 | 20 | <1 |
| long | 1 до 2 | 1 | 20 | 31 |
| short | 1 до 2 | 2 | 22 | 67 |
| int8 | 1 до 2 | 1 | 23 | 53 |
| int16 | 1 до 2 | 1 | 20 | 53 |
| int32 | 1 до 2 | 1 | 20 | 32 |
| int64 | 1 до 2 | 1 | 20 | 30 |
| longlong | 1 до 2 | 2 | 20 | 49 |
| ulonglong | 1 до 2 | 2 | 20 | 49 |
| float | 1 до 2 | <1 | 20 | 27 |
| Decimal (str) | 1 до 2 | 30 | 64 | 353 |
| float16 | 1 до 2 | 1 | 23 | 30 |
| float32 | 1 до 2 | 1 | 23 | 49 |
| float64 | 1 до 2 | 2 | 23 | 49 |
| bin | 1 до 2 | 1 |  |  |

Время перемножения элементов int при разных длинах массива (поиск пределов):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Тип данных** | **Время умножения при L=…** | | | | |
| 10тыс. | 100тыс. | 300тыс. | 600тыс. | 700тыс. |
| int | 6 | 280 | 2266 | 8731 | Переполнение |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Тестирование | | | | | |
| **Тип данных** | **Вес массива, MB** | **Время заполнения, ms** | **Диапазон** | **Время выполнения, ms** | | |
| **Сложение/вычитание** | **Умножение/Деление** | **Умножить/возвести на 5** |
| int | 8,057 | 197 | 1 до 5 | 4 | Сбой | 2941 |
| long | 3,815 | 17 | 1 до 5 | 77 | Переполнение | 4141 |
| short | 1,907 | 14 | 1 до 5 | 140 | Переполнение | 6967 |
| int8 | 0,953 | 12 | 1 до 5 | 135 | Переполнение | 5957 |
| int16 | 1,907 | 15 | 1 до 5 | 139 | Переполнение | 5957 |
| int32 | 3,815 | 17 | 1 до 5 | 80 | Переполнение | 3433 |
| int64 | 7,63 | 17 | 1 до 5 | 74 | Переполнение | 4770 |
| longlong | 7,63 | 17 | 1 до 5 | 73 | Переполнение | 5004 |
| Ulonglong | 7,63 | 17 | 1 до 5 | 150 | Переполнение | 6618 |
| float | 7,63 | 192 | 1 до 5 | 6 | 1240 | 2782 |
| Decimal (str) | 7,63 | 163932 | 0.999-1.001 | Сбой | Сбой | Сбой |
| float16 | 1,907 | 277 | 0.999-1.001 | 152 | 1503 | 8112 |
| float32 | 3,815 | 271 | 0.999-1.001 | 155 | 1500 | 5914 |
| float64 | 7,63 | 13 | 0.999-1.001 | 90 | 1394 | 3154 |
| bin | 0,953 | 24 | 1 до 5 |  | Переполнение | |

Результаты тестирования при длине массива равной 1млн:

Результаты тестирования заполнения массива имеющимися типами при разной длинне:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Тип данных** | **Время заполнения** | | | | | |
| 10тыс, мс | 100тыс, мс | 1млн, мс | 100млн, мс | 1млрд, мс | 2млрд, мс |
| int | 17 | 60 | 577 | 59711 | Долго | Долго(<ОЗУ) |
| long | 4 | 12 | 20 | 1566 | 16567 | Долго(<ОЗУ) |
| short | 3 | 5 | 17 | 1285 | 13323 | 43097 |
| int8 | 3 | 4 | 15 | 1056 | 11157 | 50464 |
| int16 | 3 | 9 | 17 | 1295 | 13033 | 43097 |
| int32 | 3 | 5 | 20 | 1591 | 15824 | Долго(<ОЗУ) |
| int64 | 5 | 6 | 19 | 1584 | 24215 | Долго(<ОЗУ) |
| longlong | 4 | 5 | 19 | 1607 | 24200 | Долго(<ОЗУ) |
| ulonglong | 6 | 5 | 19 | 1607 | 24320 | Долго(<ОЗУ) |
| float | 25 | 195 | 1723 | Долго | Долго(<ОЗУ) | Долго(<ОЗУ) |
| float16 | 23 | 191 | 1921 | Долго | Долго(<ОЗУ) | Долго(<ОЗУ) |
| float32 | 23 | 204 | 1926 | Долго | Долго(<ОЗУ) | Долго(<ОЗУ) |
| float64 | 53 | 4 | 14 | Долго | Долго(<ОЗУ) | Долго(<ОЗУ) |
| bin | 26 | 8 | 36 |  | Долго | Долго |

Результаты более подробно отражены в диграмме:

Примерное место нехватки ОЗУ

# Результаты тестов и выводы

## 5.1 Различия в типах переменных

В ходе тестов было выяснено, что у разных типов данных разные пределы выводимого ответа (максимальное число в их количестве бит). Особенно это критично при умножении и делении, т.к. рост (уменьшение) числа в ответе геометрический(-ое). При переполнении типа он уходит в “минус” и далее в нуль, либо в inf в случае умножения и 0 в случае деления (т.к. интерпретатор вычисляет 1/inf).

Типы различаются по времени заполнения и выполнения арифметических операций, т.к. имеют разный вес 1 яейки массива. Например используя тип int8 в памяти 1 ячейка будет занимать 8 бит, а тип int64 будет занимать в 8 раз больше места, следовательно для транспортировки из подсистемы памяти в кеш процессора и операций над ячейкой потребуется больше времени, из-за этого происходит различие результатов.

## 5.2 Результаты исследования массивов с малыми числами

Я попытался предотварить переплнение в умножении используя малый диапазон генерации (от 0.999 до 1.001), благодаря этому я выяснил, что в моём случае тип данных с плавающей запятой практически не влияет на время умножения.

В случае со сложением и вычитанием проблемы наблюдаются заметно позже, т.к. ответ растет арифметической прогрессией.

## 5.3 Аппаратные ограничения

Путём тестов было выяснено, что эффективно вычисляемое максимальное значение l для 6GB ОЗУ– равно около 200млн с типом int32 (значение может превышать максимум в зависимости типа данных). В пределах этого значения программа способна эффективно заполнять массив без использования файла подкачки (из-за переполнения ОЗУ). Если необходимо заполнить массив, превышающий это значение (а именно ОЗУ для создания потребуется больше, чем доступно), то время выполнения увеличится в соответствии с тем насколько накопитель пользователя медленнее ОЗУ. Для популярных ddr4 2400mhz и ssd значения скорости примерно равны соответственно 30GB/s и 0.5GB/s, следовательно это приведет к замедлению работы программы.

Можно задавать массивы и длинной более 1млрд, что уже не будет так эффективно, однако в таком случае максимальная длинна массива ограничивается только размером файла подкачки ОС windows (например 20ГБ), т.к. если он установлен в режиме авто не получится вычислить и даже 800млн элементов без использования большого объема установленной ОЗУ.

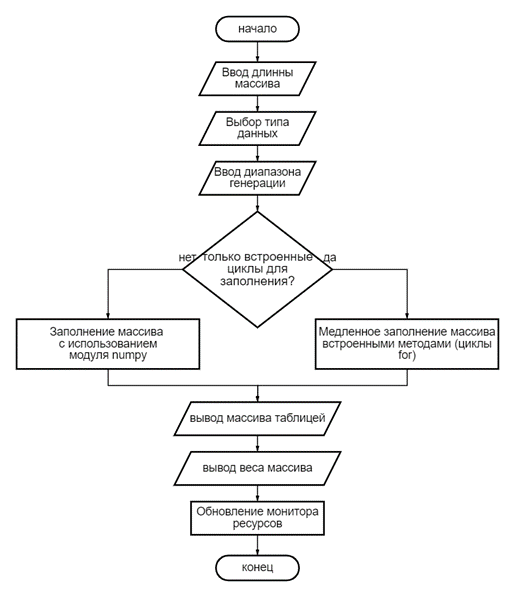
Как ранее было отмечено, время заполнения (и остальные операции, требующие память) существенно увеличивается при исчерпывании доступной памяти, связано это с обращением к файлу подкачки, расположенному на диске. При заполнении больших массивов (весом более объёма доступной ОЗУ), могут возникать проблемы с файлом подкачки (вылетит компилятор), если он установлен в значение по-умолчанию (при тестировании на всех машинах он был установлен размером 20GB).

На машине2 резкое увеличение времени от размера массива происходит позже из-за больше объема ОЗУ. До этого момента время вычисления примерно равно, скорее всего из-за схожести архитектур (и тот и тот zen+) и производительности двух процессоров ryzen (у второй машины 6 ядер, однако меньшая их частота, что приводит к подобному результату).

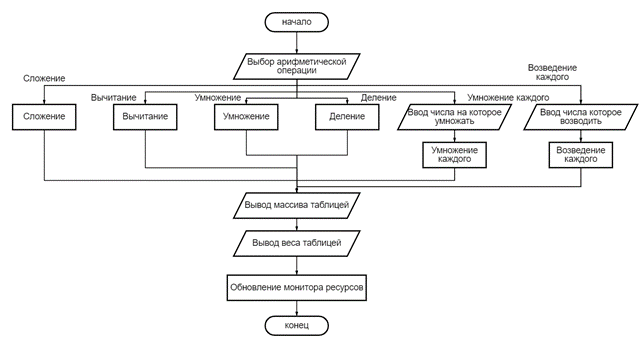
## 5.4 Роль импортируемых типов

Важно подметить, что заполнение через циклы for работает куда медленнее специализированных функций numpy, также импортируемые типы данных за счёт своего более оптимизированного веса позволяют проводить более быстрые операции заполнения, однако проигрывают встроенным типам в операциях вычисления. Таким образом, используя эти типы по назначению можно добиться улучшения результата.

# 6.Техническое руководство



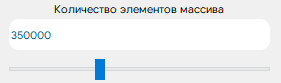
*UML диаграмма подпрограммы «Заполнить»*



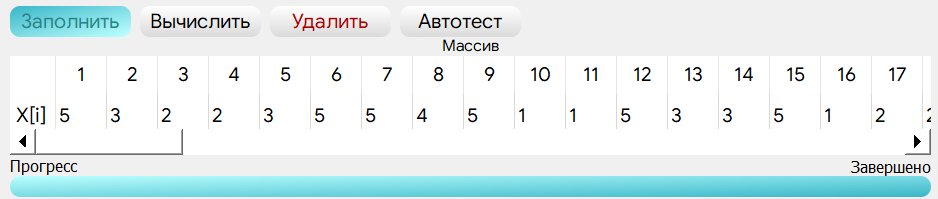
*UML диаграмма подпрограммы «Вычислить»*

## 6.1 Создание массива размером L [x1,x2]:

В соответствующее окно программы вводится число – длинна массива L, также указываются границы диапазона [x1,x2] (целые для целых типов и дробные для типов с плавающей запятой), для удобства длинну массива можно указать слайдером:



О прогрессе операции сообщит прогресс бар голубого цвета, по завершению массив отобразится в соответствующей таблице, а в поле “Вес массива” отобразится его вес, конвертированный для удобства в байты, KB и MB:





Переключатель “*Только циклы for*” отключает заполнение с помощью функций numpy и работает только при помощи встроенных методов python 3. Это замедляет скорость работы программы при заполнении массивов с типами переменных numpy - *long, short, int8, int16, int32, int64, longlong, ulonglong, float64, bin.*

## 6.2 Выбор типа массива:

Представленные в программе типы выбираются из соответствующего списка и обрабатываются как ключ при помощи операторов условия и заполняется соответствующими командами:

***Int:***

Стандартный тип данных python 3

X = np.random.randint(x1, x2+1, l,dtype=np.int64)

for i in range(len(X)):

X.append(int(X[i]))

***Long:***

Импортируемый тип из numpy

X=np.random.randint(x1, x2, l,dtype=np.int\_)

***Short:***

Импортируемый тип из numpy

X=np.random.randint(x1, x2, l,dtype=np.short)

***Int8-64:***

Импортируемый тип из numpy

X=np.random.randint(x1, x2, l,dtype=np.int8-64)

***(u)longlong:***

Импортируемый тип из numpy

X=np.random.randint(x1, x2, l,dtype=(u)np.longlong)

***Float:***

Стандартный тип данных python 3

X[i]=random.uniform(x1, x2)

***Decimal (str):***

Для хранения в массиве импользуется строковый тип и каждый символ после запятой генерируется отдельно через for (очень медленная работа на больших длиннах)

strl=str(random.randint(x1,x2))+"."

for j in range(dec):

n=str(random.randint(0,9))

strl=strl+n

X[i]=strl

***Float16-64:***

Импортируемый тип из numpy

X=np.zeros(l,dtype="float16-64")

for i in range(l):

X[i]=random.uniform(x1, x2)

***Boolean:***

Стандартный тип данных python 3

X=np.random.choice([False, True], size=(l,), p=[1./3, 2./3])

**Для работы программы необходимы следующие директивы:**

**gc** (garbage collector) – для работы функции удаления массива из памяти:

import gc

gc.collect()

Для успешной работы интерфейса:

from **PyQt5** import QtWidgets, QtGui, QtCore

from **PyQt5.QtCore** import pyqtSignal, QObject, pyqtSlot

from **PyQt5.QtGui** import QIntValidator

from **PyQt5.QtWidgets** import QTableWidgetItem,QProgressBar

**psutil** – для работы монитора ресурсов (показатели ОЗУ и частот процессора)

**time, datetime** – для работы таймера

**functools.reduce** – для оптимизированного перемножения массивов

**sys** – для измерения объема данных, занимаемых массивом

**numpy** – для использования типов данных из языка C (*np.long, np.short, np.int8, np.int16, np.in32, np.int64, np.longlong, np.ulonglong, np.float16, np.float32, np.float64*)

**random** – для задания рандомизированных значений

**subprocess** – для запуска файла с документацией, при нажатии на кнопку интерфейса “О программе”

Для динамической точности операций с float:

from **decimal** import **Decimal**

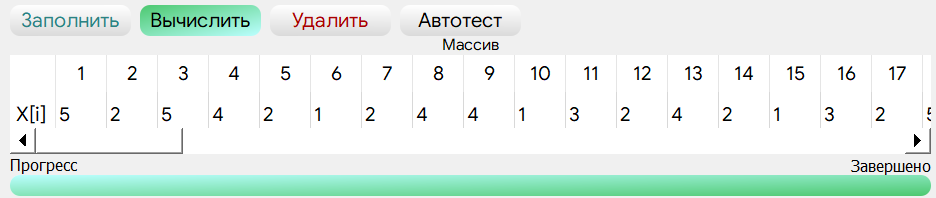
from **decimal** import **getcontext**

## 6.4. Выполнение арифметических операций над элементами массива:

Если массив до этого не был создан или был удалён программа об этом сообщит в логе:



О прогрессе операции сообщает прогресс бар под таблицей окрашенный в зеленый цвет:



При выборе операции *“****Сложение****”* все элементы массива будут суммированы в перменную ***s*** методом ***sum()***. С помощью команд ***int(round(time.time() \* 1000))*** будет вычислено время выполнения оперции посредством нахождения дельты между конечным и начальным временем (t2-t1).

Операция ***“Вычитание”*** представляет из себя обратную сложению, поэтому выполняется тем же методом, но со знаком отрицания ***(-sum()).*** Поэтому время выполнения двух этих операций приблизительно будет равным.

По-скольку операции умножения и деления с типом данных float могут потребовать дополнительной точности (при которой 8 знаков после запятой будет недостаточно) была добавлена динамическая точность, реализованная с помощью модуля decimal и устанавливаемая перед началом вычислений в соответствующем окне:



*Указание слишком большой точности может увеличить время операций.*

Повышенную точность (устанавливаемую параметром Dec L) можно также установить и для каждой ячейки массива, данные будут храниться в массиве построчно (рандомизироваться будут также и все символы после запятой), однако скорость заполнения при использовании этого метода падает в разы, поскольку к первому циклу for с ячейками добавляется второй цикл for, который посимвольно заполняет в строку рандомными числами. Задается галочкой в соответствующем окне:



Выбор операции *“****Умножение****”* приводит к умножению всех элементов последовательно на стартовое s=1 при помощи цикла for:

for i in range(l):

s\*=Decimal(X[i])

При выборе операции “**Деление**” происходит циклическое деление s=1 на элементы массива, т.е. 1 деленная на операцию умножения.

Операция ***“Умножить на …”*** домножает каждый элемент массива на число, заданное в окно интерфейса, при этом массив полностью перезаписывается и обновляется графическая таблица в интерфейсе. Реализовано с помощью стандартного цикла for с одновременным обновлением таблиц:

for i in range(len(X)):

X[i]=X[i]\*mn

cellinfo = QTableWidgetItem(str(X[i]))

self.ui.tableWidget.setItem(0, i, cellinfo)

При выборе операции *“****Возвести в … степень****”* программа возведёт каждый элемент массива в степень указанную в окно интерфейса, при этом массив полностью перезаписывается и обновляется графическая таблица в интерфейсе. Реализовано с помощью стандартного цикла for с одновременным обновлением таблиц:

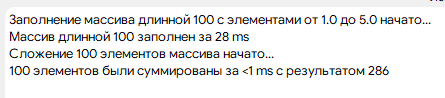
for i in range(len(X)):

X[i]=X[i]\*\*mn

cellinfo = QTableWidgetItem(str(X[i]))

self.ui.tableWidget.setItem(0, i, cellinfo)

Об окончании операции пользователя уведомит соответсвующая надпись в логах, в ней содежится информация о количестве элементов вычисленного массива, времени вычисления и о результате вычисления:



T=t\_2-t\_1

if T<1: log('Автотест закончен за <1ms',self)

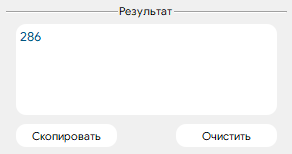
else: log('Автотест закончен за '+ str(T) +' ms',self)

*- тут log() – функция для сокращения кода:*

def log(a,sel):

sel.ui.textBrowser.append(str(a))

Для более удобного просмотра результат также продублирован в соответствующем окне интерфейса:



## 6.5 Алгоритм работы функции “Автотест”

При нажатии на соответствующую кнопку интерфейса программа выполняет проверку на существование массива, если массив до этого не был создан или был удалён программа об этом сообщит в логе:



Далее программа выполняет Тест №1: проверка на количество элементов, используя стандартные функции len(X) сравнивает полученную длинну с той, что была задана при создании массива:

if len(X)==l:

t2=int(round(time.time() \* 1000))

T=t2-t1

if T<1: log(' проверка выполнена успешно за <1 ms (массив сгенерирован полностью и имеет '+ str(l)+ ' элементов)',self)

else: log(' проверка выполнена успешно за ' + str(T) +' ms',self)

else:

t2=int(round(time.time() \* 1000))

log(' проверка закончилась неудачей',self)

Исходя из результата в логи выводится соответствующее сообщение:

*проверка выполнена успешно за <1 ms (массив сгенерирован полностью и имеет 10000 элементов)*

Далее идёт выполнение Тест №2: проверка на сумму арифметической прогрессии, используется формула суммы элементов n членов прогрессии:

*-где a1 – x1 (нижняя граница диапазона), a2 – x2 (верхняя граница диапазона), n – L (длинна массива).*

Затем происходит сравнение полученного *S* по формуле с тем, что получилось в реальной сумме массива и выводится отличие от формулы в процентах:

t1=int(round(time.time() \* 1000))

s=sum(X)

S=((x1+x2)/2)\*l

t2=int(round(time.time() \* 1000))

T=t2-t1

if T<1: log(' проверка выполнена успешно за <1 ms (s отличается от Sn на '+ str(round(abs(s/S-1)\*100,1))+' %)',self)

else: log(' проверка выполнена успешно за ' + str(T) +' ms (s отличается от Sn на '+ str(round(abs(s/S-1)\*100,1))+' %)',self)

По окончанию следует выполнение Тест №3: проверка на диапазон, каждое число с помощью цикла for прогоняется через условие вхождения в [x1,x2], если находится не входящее число, то переменная flag станет равна 1 и выведется ошибка:

t1=int(round(time.time() \* 1000))

fl=0

for i in range(l):

if not(x1<=X[i] and X[i]<=x2+1):

fl=1

break

t2=int(round(time.time() \* 1000))

if not(fl):

T=t2-t1

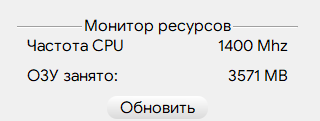
if T<1: log(' проверка выполнена успешно за <1 ms (каждый член массива входит в диапазон [' + str(x1)+','+str(x2)+'])',self)

else: log(' проверка выполнена успешно за ' + str(T) +' ms (каждый член массива входит в диапазон [' + str(x1)+','+str(x2)+'])',self)

else: log(' проверка закончилась неудачей',self)

## 6.6 Алгоритм работы подпрограммы “Монитор ресурсов”:

Для более наглядного мониторинга ресурсов машины, за которой пользователь выполняет вычисления было добавлено компактное поле Монитора ресурсов, которое включает в себя показатели частоты CPU и занятого объёма ОЗУ:



Т.к. в этой версии программы отсутствуют параллельные вычисления, значения в нём обновляются только по началу и окончанию какого-либо действия связанного с обработкой данных в программе или по нажатию кнопки обновить. Для реализации была использована библиотека *psutil* и следующие методы:

self.ui.label\_15.setText(str(round(psutil.virtual\_memory()[3]/1024\*\*2))+' MB')

self.ui.label\_17.setText(str(int(psutil.cpu\_freq().current))+' Mhz')

## 6.7 UML диаграммы разработчика

