# Лабораторная 8

Функции в качестве параметра падаётся число ( количество потоков исполнения ). Необходимо преобразовать полученную строку в argv в число.

# Используемые функции:

## Функция strtol

Сигнатура функции:

Функция strtol() преобразует начальную часть строки( до невалидного символа) в nptr к длинному целочисленному значению в соответствии с заданной базой системы счиления аргументы функции:

- 1. указатель на строку, которую необходимо преобразовать
- 2. указатель на область памяти, куда функция поместит (если указатель ≠ NULL ) первый невалидный символ
- 3. Основание системы счисления

#### Возвращаемое значение:

- успешное выполнение результат преобразования
- underflow LONG MIN
- overflow LONG MAX
- преобразование невозможно ( нет цифр ) 0

#### Возвможные ошибки:

**EINVAL** — указанное основание системы счисления невалидно

**ERANGE** — для записи результата преобразований недостаточно памяти ( underfow или overflow )

**EINVAL** — преобразование невозможно ( нет цифр )

Лабораторная 8

### Функция malloc

#### Сигнатура функции:

```
#include <stdlib.h>
void *malloc(size t size);
```

Функция malloc() выделяет переданное количество байтов и возвращает указатель на выделенную память. Память не инициализируется. Если размер равно 0, тогда malloc() возвращает либо NULL, либо уникальный указатель значение, которое позже может быть успешно передано free().

#### Возвращаемое значение:

- упешное выполение указатель на выделенную память, которая соответствующим образом выровнена для любого встроенного типа.
- ошибка эти функции возвращают значение NULL. ( NULL также может быть возвращен успешным вызовом malloc() с размером равным нулю.)

#### Возможные ошибки:

**ENOMEM** — Недостаточно памяти, чтобы выделить такое количество байтов.

А также функции для работы с потоками, которые мы использовали в задачах 1-2

# Общее описание работы программы

В начале мы проверяем валидность переданных на вход данных с помощью функции is\_valid\_input ( проверяет правильное ли количество аргументов, а также является аргумент числом и входит ли в диапозон допустимых значение )

Затем, вызываем функцию, которая поместит нам по указанному адресу вычисленное значение числа Пи — calculate\_pi

Она создаёт заданное количество потоков, расвномерно распределяем между ними задачи и после завершения их работы и их присоединения считает общую сумму из частичных

Отметим, что в функции, которую выполняют потоки не возвращается указатель на перменную созданную в ней на стеке, тк после присоединения не гарантируется валидность этих данных.

Лабораторная 8

# Анализ полученного результата

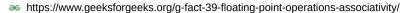
Мы получаем различные результаты с одинаковым числом итераций но разным числом потоков.

Так получается из-за представления чисел с плавающей точкой, в данном случае, из-за представления дабла. Во-первых отметим, что не все действительные числа могут быть представлены в системе счисления с основанием 2. То есть, например, числа 0,1 и 0,2 на самом деле будут представлены числами очень близкими к ним, так как они не являются дробями с степенью двойки в знаменателе. Таким образом, например 0,1 + 0,2 ≠ 0,3. Вовторых, операции с числами с плавающей точкой не являются ассоциативными

Например, выражение (x+y) + z имеет совершенно иной ответ, чем x + (y+z), когда x = 1e30, y = -1e30 и z = 1 (в первом случае это 1, во втором - 0).

#### Floating Point Operations & Associativity in C, C++ and Java - GeeksforGeeks

Do Floating point operations follow property of associativity? In other words, do we always get the same results for expressions "(A + B) + C" and "A + (B + C)" One may expect that floating numbers to follow the rule of associativity in programming languages





#### course/float.md at master · cme212/course

Floating point data types are finite precision approximations for real numbers. C++ standard recognizes three fundamental floating point data types: float - single precision floating point type double - double precision floating point type

https://github.com/cme212/course/blob/master/notes/lecture-11/float.md



Лабораторная 8