УДК 004

# Использование C++ и библиотеки OpenMp для создания многопоточных приложений

*Азаров Андрей Евгеньевич*
*Приамурский государственный университет имени Шолом-Алейхема*
*Студент*

*Глаголев Владимир Александрович*
*Приамурский государственный университет имени Шолом-Алейхема*
*к.г.н., доцент кафедры информационных систем, математики и методик обучения*

**Аннотация**
В данной статье описывается концепция многопоточных приложений, их преимущества, процесс создания на C ++, пример создания многопоточного приложения и результаты работы.
**Ключевые слова:** программирование, C++, многопоточность.

## Using C ++ and OpenMp library for creating multi-threaded applications.

*Azarov Andrey Evgenevich*
*Sholom-Aleichem Priamursky State University*
*Student*

*Glagolev Vladimir Alexandrovich*
*Sholom-Aleichem Priamursky State University*
*candidate of geographical Sciences, associate professor of the Department of information systems, mathematics and teaching methods*

**Abstract**
This article describes the concept of multi-threaded applications, their advantage, the process of creating them in C ++, an example of creating a multi-threaded application and work results.
**Keywords:** programming, C ++, multithreading.

Одним из способов ускорить работу программу является - многопоточное программирование, разбиение процессов приложения на различные потоки, которые выполняются параллельно и независимо друг от друга. К достоинствам данного подхода является следующее: упрощение программы за счёт использования общего адресного пространства, меньшие относительно процесса временные затраты на создание потока, повышение производительности процесса за счёт распараллеливания процессорных вычислений и операций ввода-вывода данных.

Целью работы является разработка приложения, позволяющего производить сложные математические вычисления, на примере расчета значений приближенных чисел.

Для выполнения цели работы необходимо выполнить: выбор числа для приближенного вычисления; определения языка программирования; проектирование авторского алгоритма; сравнения работы, реализованного алгоритма различными модулями среды объектно-ориентированного программирования.

Для демонстрации работы многопоточных программ существует множество бесконечных функций и рядов, возьмём число ПИ, точное число ПИ до пятнадцатого знака после запятой имеет значение 3,141592653589793, большее количество цифр не входило в диапазон исследования.

Современные информационные средства моделирования процессов основаны на языка объектно-ориентированного программирования, наибольшее распространения получил C++.

В языке программирования C++ существуют встроенные наборы функций в библиотеке thread.h, которая создана специально для работы с потоками[1]. Появилась библиотека thread.h в 11-ой версии языка C++, используя эту библиотеку можно создать отдельный поток или несколько потоков, потоки позволяют нескольким фрагментам кода работать асинхронно и одновременно. Либо библиотека Open Multi-Processing (OpenMP) – это интерфейс API, созданное для программирования многопоточных приложений. Разработку OpenMP ведут множество крупнейших компаний, таких как IBM, Oracle, Intel. Библиотека значительно упрощает разработчикам процесс создания приложений с потоками, а также даёт множество новых способов их реализации.

Каждая программа изначально имеет один главный поток – алгоритм, который выполняется последовательно. В этом случае задействовано только одно ядро процессора. Программист может, и должен, разгрузить и уменьшить нагрузку на один процессор, передав эту же задачу другим ядрам, другими словами распараллелить задачу. Данный способ не всегда актуален, но помогает ускорить программу, также скорость программы в данном случае зависит от количества ядер процессора. Оптимизация программного обеспечения тесно связана с умением работы программиста с оперативной памятью и другими вычислительными ресурсами компьютера, а не только написание структурированных алгоритмов. Рассмотрим более подробно алгоритм вычисления числа Пи, который приведен на рисунке 1 [2].

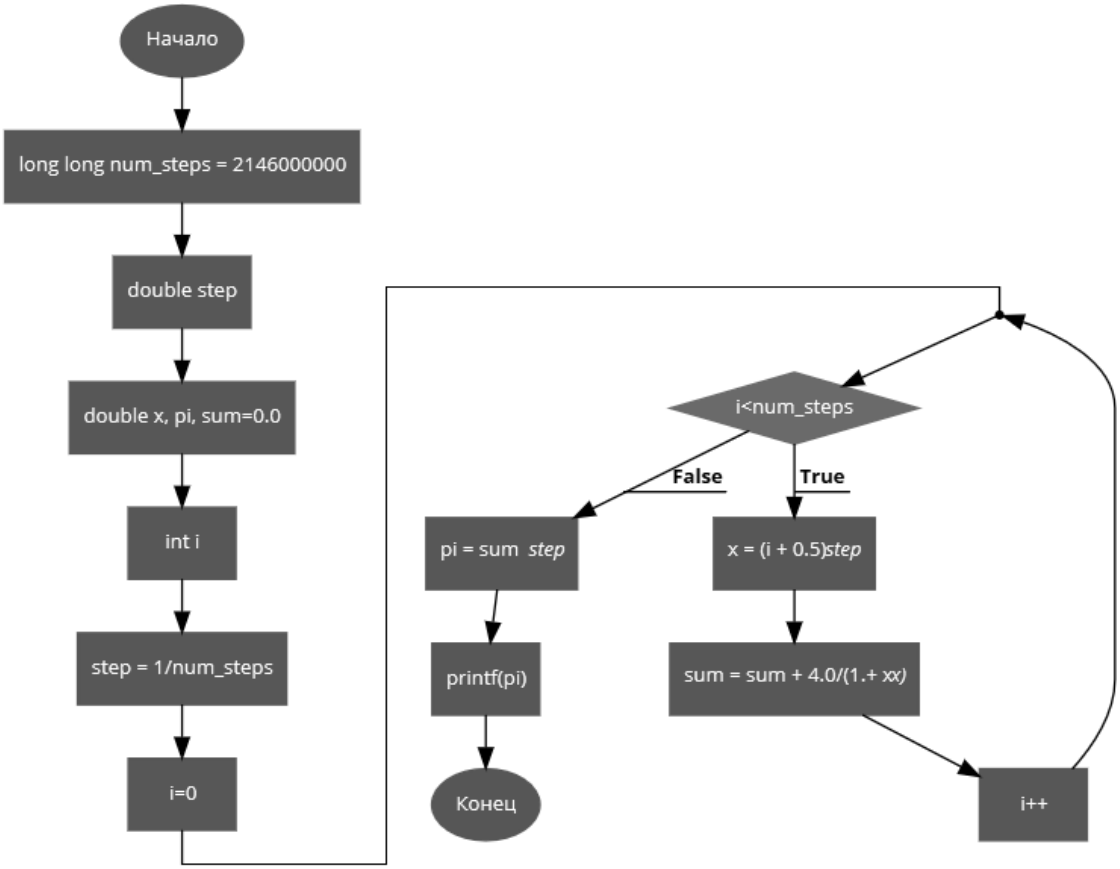


Рис. 1. Блок-схема программы по нахождению числа Пи методом интегрирования

Данный алгоритм реализован двумя способами.

Во-первых, использование библиотеки OpenMp, можно скачать на официальном сайте одной из компаний поддерживающих разработку данной библиотеки и подключать с помощью класса omp.h. В библиотеке не нужно создавать потоки в программном коде, вместо этого необходимо объявить перед исполняемой функцией директиву #pragma, данная директива укажет компилятору, что часть кода может был распараллелена. С помощью библиотеки OpenMP компилятор среды программирования самостоятельно генерирует приложение из главного потока и остальных побочных потоков, которые выполняются параллельно [3]. Эти потоки синхронизируются в конце параллельного блока программного кода, и снова возвращаются к одному главному потоку. Для создания параллельного блока кода нужно указать #pragma omp parallel. Каждый из созданных потоков выполнит одинаковый код содержащийся в блоке, но не одинаковый набор команд. В разных потоках могут выполняться различные ветви или обрабатываться различные данные, что зависит от таких условных операторов, как if-else или использования директив распределения работы. Чтобы распараллелить цикл с параметром (например, for) используют специальный набор директив, #pragma omp for, иначе будет выполнено множество избыточных операций[4]. В случае последнего примера, каждый создаваемый поток будет обрабатывать только, отданную ему часть массива данных.

Во-вторых, применение стандартной библиотеки среды программирования thread.h. В этой библиотеке для создания потока используется функция CreateThread, для синхронизации потоков CreateMutex, ExitThread для завершения текущего потока. Данные функции являются базовыми операциями и могут сильно отличаться от конкретного компилятора и среды разработки на языке C++. Работа с потоками содержит множество сложностей, связанных с синхронизацией и не всегда даёт качественный результат. Разработчику начального уровня легко запутаться в многопоточности, и потоки будут только замедлять работу программу.

Проведенный эксперимент вычисление числа «Пи» (каноничный результат 3,141592653589) дал следующие результаты:

1) без использования OpenMP 3,141592653608 за 7,193 секунд;

2) с использованием OpenMP 3,141592653590 за 1,880 секунд.

Как видно из результатов, вычисление с использованием библиотеки OpenMP не только ускорило время выполнение программы, но и увеличило точность до ещё одного знака.

Проведенное тестирование вычисления числа «ПИ» с точность до десяти знаков после запятой, на компьютере, с частотой процессора 3,4Ггц и 16 ГБ оперативной памяти показало, что при выполнение библиотеки OpenMP значительно упрощается время расчета программы и вычисление значений в многопоточных приложениях. Вычисления проходят в 3,82 раза быстрее, и при этом, приложение работает без исключительных операций, чем если бы потоки создавались через стандартный класс thread. Таким образом, альтернативная библиотека OpenMP наиболее полно оптимизирована под работу с потоками данных, в настоящее время активно используется многими профессиональными программистами во всех странах для создания кроссплатформенных приложений.

**Библиографический список**

1. Пример простейшей многопоточной программы на WinAPI // Записки программиста URL: http://eax.me/winapi-threads/ (дата обращения: 29.10.2017).
2. Вычисление числа π // Механико-математический факультет МГУ имени М.В. Ломоносова URL: http://mech.math.msu.su/~shvetz/54/inf/perl-problems/chPi.xhtml (дата обращения: 29.10.2017).
3. Введение в OpenMP: параллельное программирование на C++ // Intel Software URL: https://software.intel.com/ru-ru/blogs/2011/11/21/openmp-c (дата обращения: 29.10.2017).
4. Параллельные заметки №1 – технология OpenMP // Habrahabr URL: https://habrahabr.ru/company/intel/blog/82486/ (дата обращения: 29.10.2017).