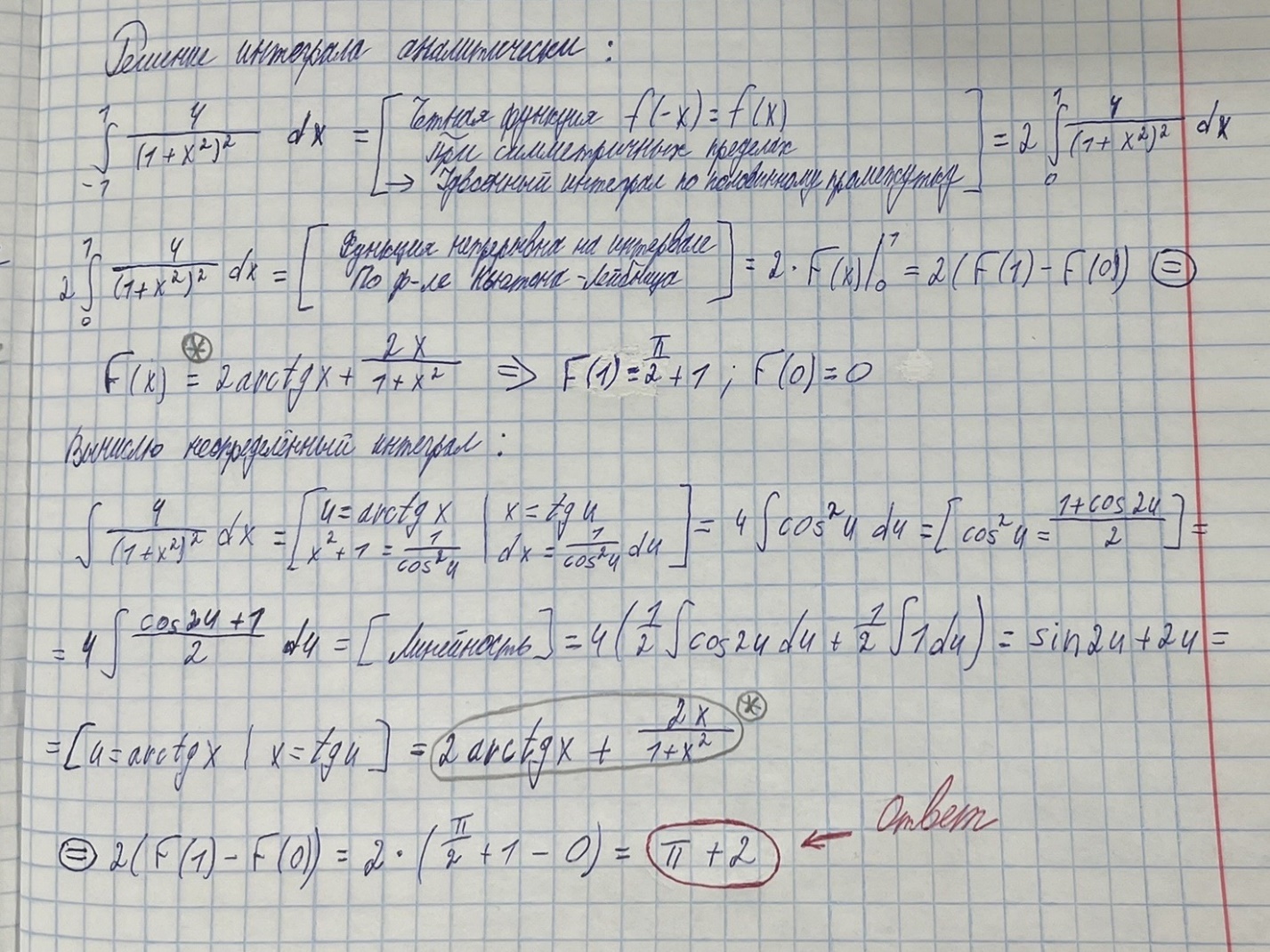
**Комплексное задание (домашняя часть работы)**

Вариант – 10



Привожу аналитическое решение интеграла:

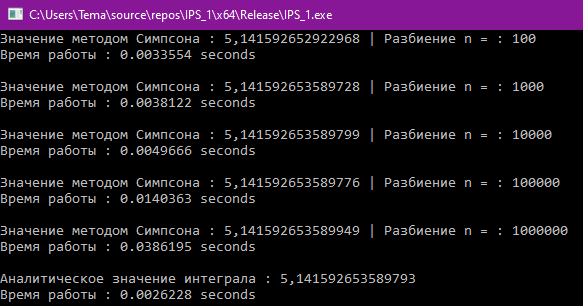


Ссылка на гит: https://github.com/T1Key/IPS

1. **Последовательная программа по расчету интеграла (1 занятие)**

После написания программы необходимо было оценить время и точность (относительно аналитического значения) расчета интеграла в зависимости от количества интервалов (равномерное разбиение, 100, 1000, 10000, 100000, 1000000).

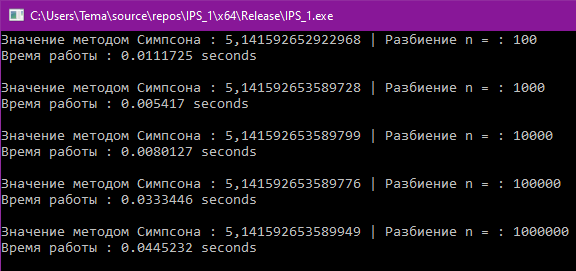
Привожу результаты выполнения последовательной программы по расчету интеграла:



Обоснование полученных результатов: стоит отметить, что точность метода Симпсона выше точности метода прямоугольников и трапеций для заданного n (в этом можно удостовериться из оценки абсолютной погрешности). По полученным результатам действительно заметна высокая точность полученных значений интегралов (относительно вычисленного аналитического значения интеграла) при равномерном разбиении n. Но следует помнить о влиянии вычислительной погрешности на результат при больших n, что может уже отдалить приближенное значение от точного. Что касается времени расчета, то при увеличении разбиения n заметно увеличение времени работы программы.

1. **Программа по расчету интеграла с отключением векторизации**

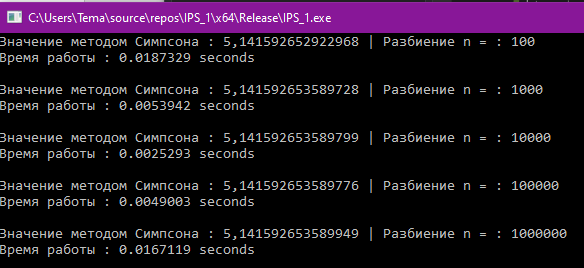
Привожу результаты выполнения программы по расчету интеграла с отключенным векторизатором:



Обоснование полученных результатов: необходимо отметить, что по умолчанию в компиляторе векторизатор включен. Поэтому для того, чтобы сравнить производительность кода при использовании векторной обработки и без неё, нужно в программе прописать #pragma loop(no\_vector) для отключения векторизации. Итак, по полученным результатам видно, что при отключении векторизации увеличивается время работы программы (относительно результатов первого пункта), сравнивая время при одном и том же разбиении первого и второго пункта.

1. **Программа по расчету интеграла с автоматическим параллелизатором**

Привожу результаты выполнения программы по расчету интеграла с автоматической параллелизацией(/Qpar):

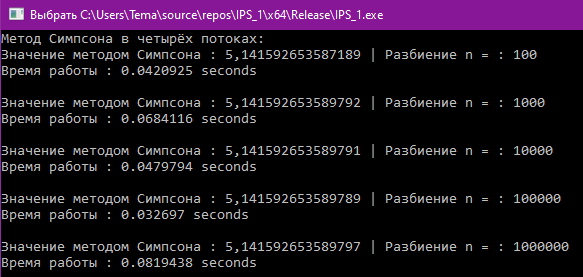


Обоснование полученных результатов: необходимо сказать, что использование автоматической параллелизации #pragma loop(hint\_parallel(0)) подразумевает работу программы на максимальном количестве потоков. Также следует отметить, что в данном методе мы как бы «подсказываем», но не указываем компилятору, что должен быть распараллелен цикл. Поэтому нельзя дать стопроцентной гарантии, что будет распараллелен данный цикл. Тем не менее, по результатам работы программы в большинстве случаев получаем выигрыш по времени выполнения по сравнению с последовательной программой. Следовательно, делаю вывод, что во многих расчетах была применена автоматическая параллелизация цикла расчёта интеграла.

1. **Программа по расчету интеграла в разных потоках**

Необходимо было добавить метод по расчёту интеграла в разных потоках с использованием библиотеки <thread>.

Привожу результаты выполнения программы по расчету интеграла в четырёх потоках:



Обоснование полученных результатов: было использовано вычисление интеграла в четырёх потоках, следовательно, необходимо было создать четыре области, далее передать границы интегрирования в функцию (стоит отметить, что в каждом потоке функция модифицирует свою переменную). Последним этапом являлось сложение этих переменных и получение всей площади. Что касается результатов, то заметно увеличение точности значения при расчёте в нескольких потоках, сравнивая с предыдущими пунктами, но заметен проигрыш во времени работы программы.