Косинов Денис Константинович БПИ239.

Вариант 22.

АВС ИДЗ#2.

**Задача:**

Разработать программу вычисления числа π с точностью не хуже 0,1% посредством дзета-функции Римана.

**Метод решения задачи:**

Дзета-функция Римана от s (далее Z(s)) = 1/1^s + 1 / 2^s + 1 / 3^s+

+ …

Из знаний мат. анализа: Z(2) = 1 / 1^2 + 1/ 2^2 + … = pi^2 / 6

Тогда можно аппроксимировать pi таким образом:

1. Посчитать частичную сумму ряда (n первых слагаемых, где чем выше n, тем точнее будет аппроксимация).
2. Умножить ее на 6.
3. Посчитать квадратный корень.

**Ссылка на источник:**

<https://ru.wikipedia.org/wiki/%D0%94%D0%B7%D0%B5%D1%82%D0%B0-%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F_%D0%A0%D0%B8%D0%BC%D0%B0%D0%BD%D0%B0>

(Здесь описано, что Z(2) = pi^2 / 6). Далее алгоритм для нахождения pi было несложно придумать).

**Код программы:**

<https://github.com/o7Techno/CSA-HW/tree/main/%D0%90%D0%92%D0%A1%20%D0%98%D0%94%D0%97%232>

**Описание программы:**

Программа разбита на 3 файла: IDZ2.ams выступает main файлом, там запускается сама программа и оттуда идут вызовы дальнейших функций; functions.asm содержит в себе все подпрограммы вызываемые в программе; macros\_lib.inc - библиотека макро.

**Описание подпрограмм:**

print\_double - выводит на дисплей double значение находящееся в регистре fa0.

print\_string - выводит на дисплей строковое значение, адрес которого находится в регистре a0.

read\_int - считывает int значение с клавиатуры и возвращает его в регистр a0.

calculate\_pi - получает в регистр a0 количество слагаемых для аппроксимации pi, а затем выполняет алгоритм описанный в “Метод решения задачи” и возвращает полученное значение в регистр fa0.

**Описание макро:**

input\_macro (%register) - считывает int значение с клавиатуры при помощи подпрограммы read\_int и записывает его в регистр %register (Обертка над read\_int).

string\_output\_macro (%address) - выводит на дисплей string значение, находящееся по адресу %address при помощи подпрограммы print\_string (Обертка над print\_string).

double\_output\_macro (%register) - выводит на дисплей double значение, находящееся в регистре %register при помощи подпрограммы print\_double (Обертка над print\_double).

approx\_pi (%amount\_address) - аппроксимирует pi при помощи подпрограммы calculate\_pi, где %amount\_address - регистр со значением кол-ва слагаемых для подпрограммы calculate\_pi (Обертка над calculate\_pi).

testing\_macro - выполняет автоматическое тестирование, подробнее см. “Автоматическое тестирование”.

mode - макро, который в зависимости от введенного режима пользования программой выполняет соответствующий функционал. Т.е. является самой крупной по вложению оберткой исполнения функционала программы.

**Нюансы:**

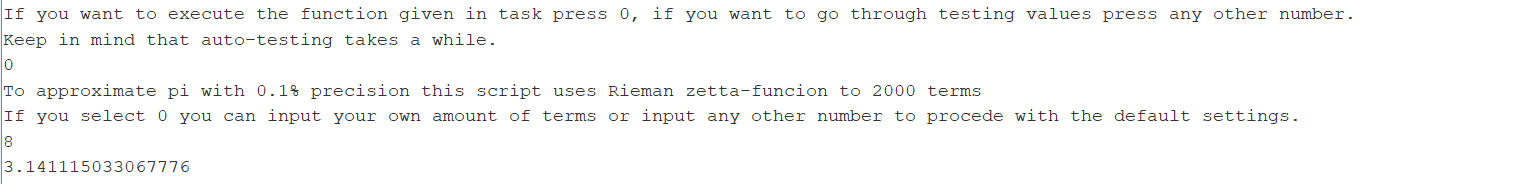
Так как в задаче варианта не был прописан ввод, было сделано так: программа, в зависимости от режима (подробнее см. “Как использовать программу”), может принимать на вход int значение, которое означает кол-во слагаемых которые будут просуммированы в подпрограмме calculate\_pi, для расчета Z(2).

Т.к. их количество напрямую связано с точностью аппроксимации pi (чем больше, тем точнее), то в автоматическом тестировании именно это количество будет подаваться в качестве входных данных для теста.

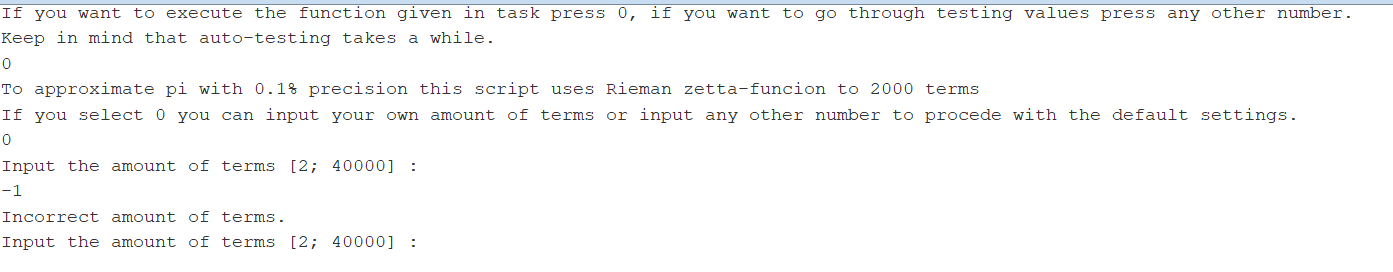
**Как использовать программу:**

Первый ввод пользователя - режим, в котором он хочет использовать данную программу. (0 - ввод данных самому, любое другое число - автотестирование). Далее, если пользователь ввел не 0, то программа выводит, корректно ли было завершено автотестирование (все ли ответы программы, совпали с действительными ответами) и на этом программа завершается, если же пользователь ввел 0 на выборе режима, то следующий его ввод будет означать режим, в котором он хочет запустить программу вручную. (0 - ввод n самому, где n кол-во слагаемых для аппроксимации pi, любое другое число - будет подставлено n = 2000, т.к. именно этого значения будет достаточно, чтобы аппроксимировать pi с точностью 0.1%, что и требуется в задании). Затем если пользователь ввел не 0, то программа выведет рассчитанную аппроксимацию pi с точностью 0.1% и на этом программа завершается, иначе если пользователь ввел 0, то следующий его ввод будет кол-во слагаемых [2; 40000] (Есть проверка на корректность введенного значения, сделано для того, чтобы можно было посчитать сумму первых n слагаемых Z(2) без переполнения) для calculate\_pi, после чего программа посчитает и выведет полученное значение на дисплей и на этом программа завершится.

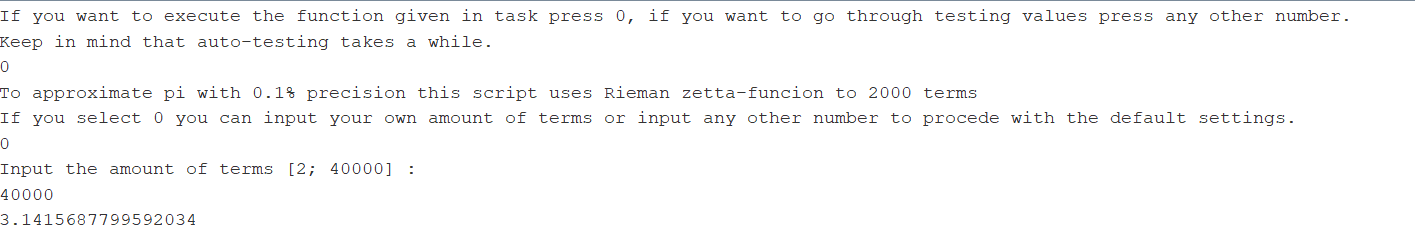
**Тест (функционал программы в задании):**

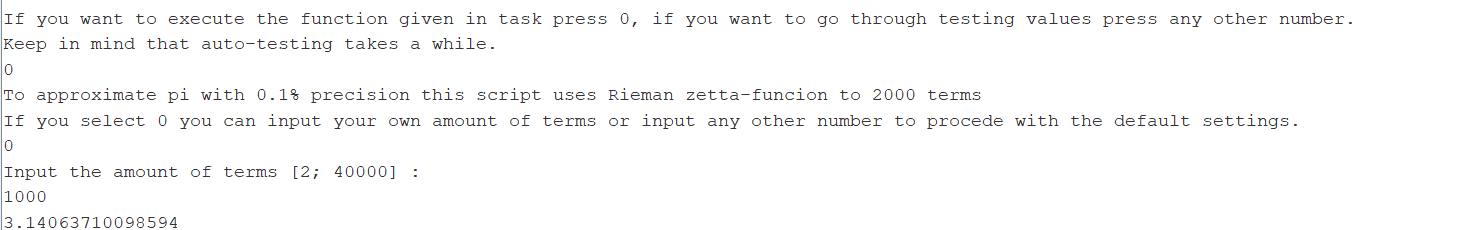
****

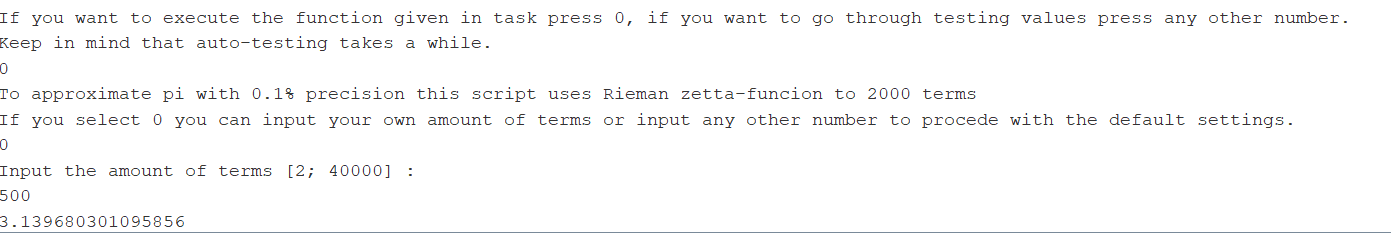
**Тесты с различными n (руками):**

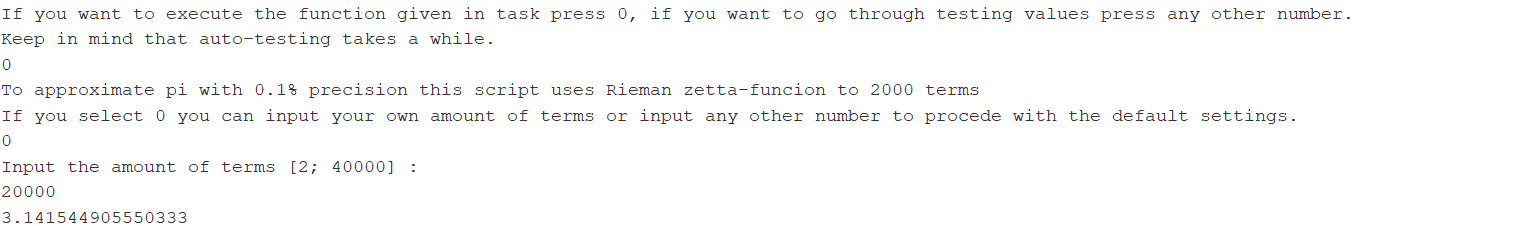
****

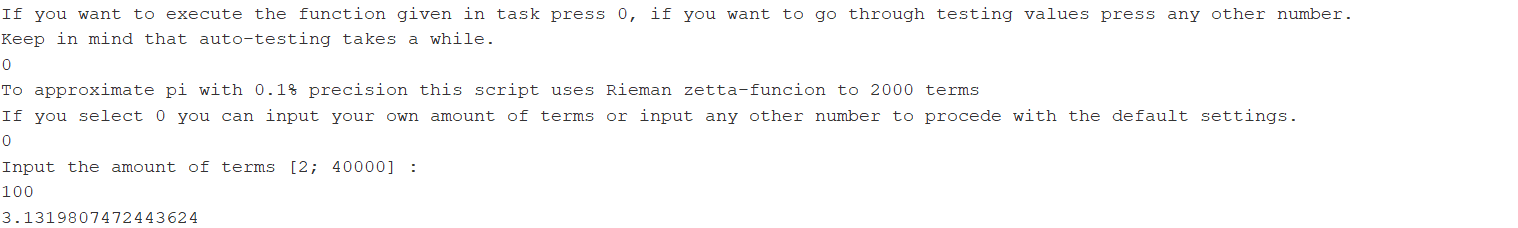
(Было введено некорректное значение n)

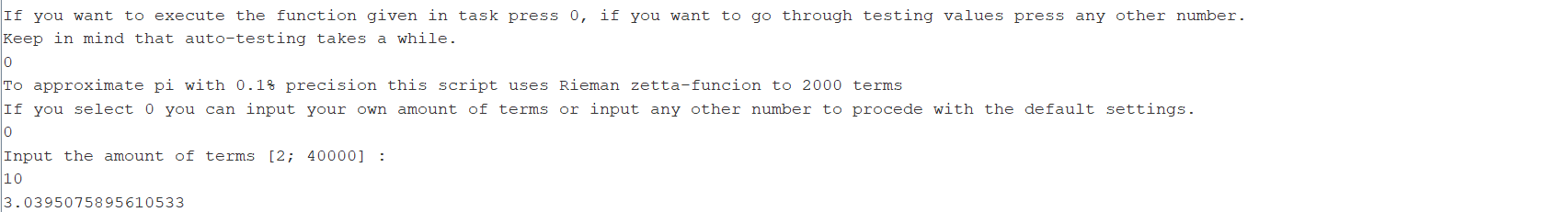












**На 4-5 баллов:**

Есть решение задачи, комментарии присутствуют, тесты с различными n можно увидеть выше.

**На 6-7 баллов:**

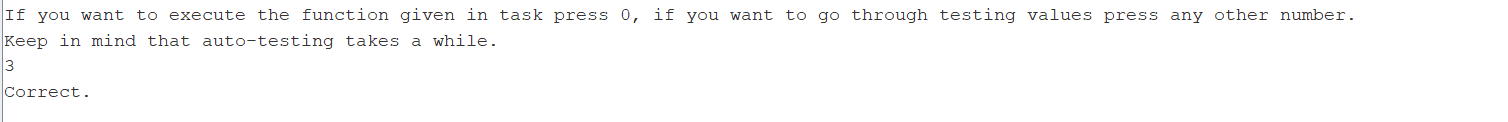
Все функции получают все необходимые данные (если они необходимы) через регистры a0 - a7 или их f версии, используют только регистры t/ft (стек не используется), а также внутри функций не используются данные не переданные им (т.е. они атомарны). Что делает их соответствующими конвенции. Все комментарии присутствуют в коде.

**Автоматическое тестирование:**

В .data с метки test\_begin до метки ans\_begin располагаются в разных word различные входные данные для calculate\_pi (16 шт.).

С метки ans\_begin располагаются 16 double значений - верные ответы для аппроксимации pi по соответствующему int значению с метки test\_begin. Когда пользователь вводит не 0 макро testing\_macro прогоняет макро approx\_macro с i-тым значением с отметки test\_begin и сравнивает, совпадает ли полученный ответ с корректным i-тым ответом с отметки ans\_begin. Если хотя бы 1 значение не совпадает, на дисплей выводится Incorrect, если все совпадают выводится Correct, после чего программа завершается.

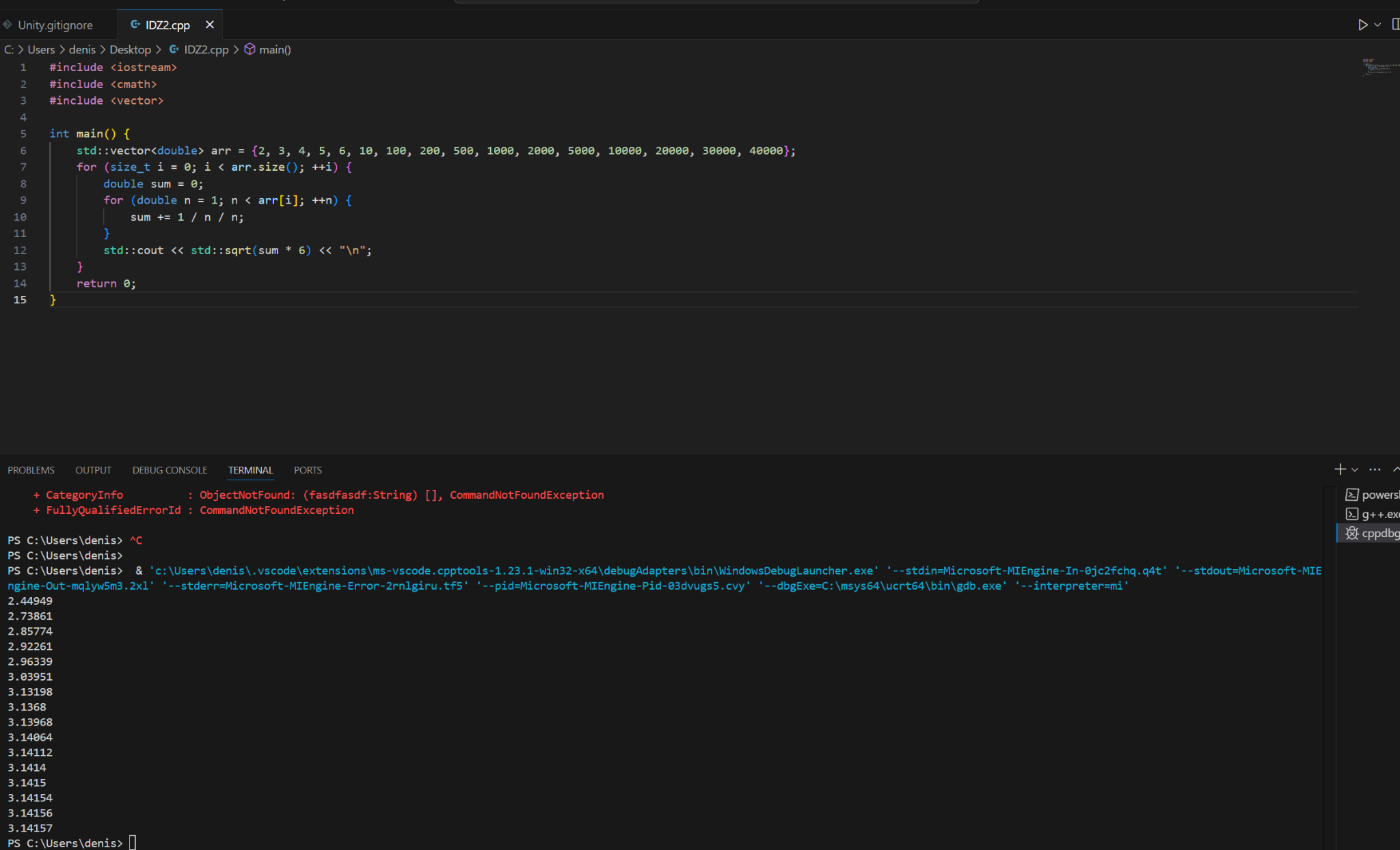
**Результат работы автоматического тестирования:**



**Код на C++ рассчитывающий значение pi по входным данным из автоматического тестирования:**

<https://github.com/o7Techno/CSA-HW/blob/main/%D0%90%D0%92%D0%A1%20%D0%98%D0%94%D0%97%232/IDZ2.cpp>

**Скриншот работы кода на C++:**

****

**На 8 баллов:**

Т.к. функции поддерживают работу с формальными данными, они поддерживают многократное использование с различными входными данными. Реализовано автотестирование. Представлена программа на C++.

**На 9 баллов:**

Выполнены требования по использованию макросов - оберток (подробнее см. описание макро).

**На 10 баллов:**

Программа разбита на 3 различных ассемблерных файла (подробнее см. описание программы). Подпрограммы print\_string, print\_double, read\_int - унифицированные модули, которые вызываются повторно в обоих режимах работы программы.

Макро лежат в библиотеке макро в отдельном файле, подпрограммы лежат в отдельном ассемблерном файле.