Архитектура вычислительных систем.
ИДЗ 2. Отчёт
Вариант 19
Работа на 10 баллов

Фролов-Буканов Виктор Дмитриевич БПИ-228

4 ноября 2023

1 Условие задачи

Разработать программу вычисления корня кубического из заданного числа **согласно быстро сходящемуся итерационному алгоритму** определения корня n-ной степени с точностью не хуже 0,05%.

2 Описание метода решения задачи

Для поиска требуемого алгоритма я воспользовался статьёй на хабре. Поняв суть алгоритма, я его несколько переделал, чтобы удовлетворить требование на точность вычисления. Для этого я использую вместо цикла *while* цикл $do - while$, в каждой итерации которого я запоминаю предыдущее значение корня ($prev\_root$) и тут же вычисляю новое значение ($root$). Если разница между новым и старым значением в отношении к новому значению (всё по модулю) больше требуемой точности, то я продолжаю вычисления, иначе - их останавливаю и вывожу полученный ответ. По сути, я проверяю, какой процент от нового значения составляет прирост точности. Если он превышает требуемую точность ($0.05\% = 0.0005$), то мы продолжаем вычисления, иначе - их останавливаем. Единственная проблема такой реализации - деление на 0 при $num = 0$, поэтому вручную добавим проверку на это значение перед входом в цикл. Ниже прикрепляю код решения на C++, для того чтобы лучше ориентироваться в написанном тексте:

```cpp
#include <iostream>

int main() {
  double num;
  const double eps = 0.0005;
  std::cin >> num;
  if (num == 0) {
    std::cout << 0.0;
    return 0;
  }
  double root = num / 3;
  double prev_root;
  double rn;
  do {
    prev_root = root;
    rn = num;
    rn /= root;
    rn /= root;
    root = 0.5 * (root + rn);
  } while (std::abs((root - prev_root) / root) > eps);
  std::cout << root;
  return 0;
}
```

3 Описание тестовых прогонов с представлением информации о результатах тестирования

Все тесты лежат в файле *tests.txt* (содержимое файла: 0 1 -1 0.0001 -0.0001 8 -8 213456 -213456 -7194.168 58125912.76123). Дополнительная тестовая программа (*testChecker.cpp*) читает тестовый файл в вектор, выводит его содержимое на экран, и записывает каждое прочитанное значение **в отдельный файл** (в файлы **test1.txt . . . test11.txt**), а после для каждого элемента вектора запускает сначала ассемблерную программу (*main.asm*), а после - запускает встроенный метод *std* :: *cbrt* из библиотеки *cmath* в C++ для проверки корректности вычислений. Результат работы тестовой программы:

```
File content: 0 1 -1 0.0001 -0.0001 8 -8 213456 -213456 -7194.17 5.81259e+07

1. RARS 1.6  Copyright 2003-2019 Pete Sanderson and Kenneth Vollmar

Enter the number: The cube root of the input number is 0.0 (Library method: 0)

2. RARS 1.6  Copyright 2003-2019 Pete Sanderson and Kenneth Vollmar

Enter the number: The cube root of the input number is 1.000118124949815 (Library method: 1)

3. RARS 1.6  Copyright 2003-2019 Pete Sanderson and Kenneth Vollmar

Enter the number: The cube root of the input number is -1.000118124949815 (Library method: -1)

4. RARS 1.6  Copyright 2003-2019 Pete Sanderson and Kenneth Vollmar

Enter the number: The cube root of the input number is 0.04640956580097921 (Library method: 0.0464159)

5. RARS 1.6  Copyright 2003-2019 Pete Sanderson and Kenneth Vollmar

Enter the number: The cube root of the input number is -0.04640956580097921 (Library method: -0.0464159)

6. RARS 1.6  Copyright 2003-2019 Pete Sanderson and Kenneth Vollmar

Enter the number: The cube root of the input number is 2.000222712416407 (Library method: 2)

7. RARS 1.6  Copyright 2003-2019 Pete Sanderson and Kenneth Vollmar

Enter the number: The cube root of the input number is -2.000222712416407 (Library method: -2)
```

Figure 1: 1 скрин

Данный тестовый файл покрывает всевозможное тестовое покрытие и примерное совпадение результатов работы ассемблерной программы и встроенного метода (расхождение возникает лишь из-за точности вычислений) говорит о корректной реализации ассемблерной программы.

4 Дополнительная информация, подтверждающая выполнение задания в соответствие требованиям на предполагаемую оценку

1. Все комментарии, а также текстовые сообщения, выводимые пользователю в качестве подсказок, написаны на английском языке во избежание конфликтов с кодировками, а также для соответствия предоставленным требованиям

2. В каждом макросе, где уместно, используется сохранение локальных переменных (целочисленных) на стек с использованием макросов **push** и **pop** из макробиблиотеки с семинаров или с использованием аналогичных макросов $\mathbf{push}_d\mathbf{pop}_d(double)()$, , , .., , (**cube_**

3. Учитывая замечание от самих авторов RARS относительно их макроассемблера ("Макросредства ассемблера не входят ни в какой стандарт и остаются на усмотрение авторов ассемблера"), я не придерживался каких-то строгих конвенций при реализации макросов. Единственное, чтобы не запутаться в ходе реализации макроса, передаю я регистр, непосредственно значение, метку или название другого макроса, я использовал следующий подход:

   - если передаётся регистр, то я сопровождал его постфиксом _reg
   - если передаётся непосредственное значение, то я сопровождал его постфиксом _imm

   Такой подход упрощает чтение сигнатуры макроса и защищает от ошибок в реализации

4. Реализованные макросы поддерживают многократное использование с различными наборами исходных данных, включая возможность подключения различных исходных данных (имеется в виду подключение из тестовых файлов). Каждый макрос не привязан к конкретному регистру или метке, что обеспечивает унификацию, то есть возможность многократного использования одних и тех же конструкций. Так, макрос с выводом строки используется дважды, а все остальные макросы хоть и используется единожды, но они являются унифицированными модулями в силу своей реализации, что и подтверждает выполнение одного из критерия на 10 баллов (разбиение программы на **унифицированные модули**)

5. Автоматизированное тестирование программы реализовано через код на C++ (файл **testChecker.cpp**), а тестовые данные поступают из файла **tests.txt**, что позволяет осуществить прогон программы, осуществляющей вычисление корня кубического для различных тестовых данных (вместо их ввода). Данные тесты составляют полный набор тестового покрытия, и программа выдаёт на них корректный ответ, что позволяет утвердить о корректности реализации ассемблерной программы

6. Расхождение результата работы ассемблерной программы и встроенного метода возникает из-за точности вычислений

7. В тестовой программе **testChecker.cpp** при запуске на локальной машине нужно поменять пути к файлам в соответствии с тем, как они расположены на вашем компьютере