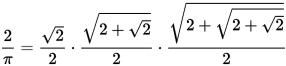
1. Начнем с разминки. Напишите программу, которая выводит на экран последовательность единиц и нулей – переменную типа unsigned int в том виде, в котором она хранится в памяти. Отдельные байты удобно отделять пробелами.
2. Напишите программу, которая выводит на экран последовательность единиц и нулей – переменную типа int в том виде, в котором она хранится в памяти.
3. Напишите программу, которая выводит на экран последовательность единиц и нулей – переменную типа float в том виде, в котором она хранится в памяти. Советую воспользоваться явным приведением типов данных, но с одной особенностью. Если вы напрямую приведете float к unsigned int, то число просто округлится до целого. Это не то, что нам нужно. Нам нужно посмотреть на тот участок памяти, в котором лежит переменная типа float, и прочитать его, ту же последовательность единиц и нулей, как unsigned int. А потом воспользоваться кодом из пункта 1.
4. Чтобы трактовать данные, которые вам выдает программа из предыдущего пункта (и убедиться в правильности ее работы), посмотрите на то, как хранятся float’ы в стандарте IEEE 754: <https://en.wikipedia.org/wiki/Single-precision_floating-point_format>
5. Напишите код на С, демонстрирующий эффект переполнения мантиссы – потери точности при вычислениях. Попробуйте придумать, как продемонстрировать этот эффект одним присвоением.
6. Напишите код на С, демонстрирующий неассоциативность арифметических операций.
7. Напишите код на С, демонстрирующий эффект антипереполнения. Определите минимальное по модулю число, которым может оперировать компьютер для типов float и double. Проверьте, поддерживает ли ваш сопроцессор денормализованные числа. Сравните производительность работы сопроцессора на нормализованных и денормализованных числах.
8. Найти среднее арифметическое массива из вещественных чисел и засечь время работы вашего кода. Советую усреднять по большому количеству измерений и использовать достаточно большие массивы. Сравнить время работы при оптимизациях О0, О1, О2 и О3.
9. Оценить, как влияет исполнение различных операций с числами с плавающей точкой (сложение, вычитание, умножение, деление и т.д.) на время работы.
10. Ускорить программу из предыдущего пункта за счет конвейера операций. Напоминаю, что операции с вещественными числами выполняет математический сопроцессор, который, вообще-то, работает достаточно независимо от центрального процессора (который оперирует только целыми числами), и способен выполнять операции параллельно с ним. Вам нужно добиться того, чтобы за один шаг цикла количество операций с вещественными числами соответствовало количеству операций с целыми числами.
11. Напишите код, вычисляющий число пи по различным формулам. Несколько тут приложены, можете откопать еще дополнительных. Да, их очень много. Для каждой из формул постройте график – зависимость вычисленного значения от количества итераций. Лучше строить все кривые на одном графике, чтобы разница между формулами была более наглядной. Найдите хотя бы одну формулу, которая сходится к теоретическому значению, и хотя бы одну, которая разваливается и уходит в бесконечность. Проверьте эффект на float и double.











1. Разберитесь с floating point paranoia.

R. Karpinski. 1985. Paranoia: A floating-point benchmark. Byte Magazine 10, 2 (Feb.), 223-235.

<http://www.netlib.org/paranoia/paranoia.c>