УДК 004.021

АЛГОРИТМ БЫСТРОГО ВЫЧИСЛЕНИЯ ОБРАТНОГО КВАДРАТНОГО КОРНЯ

Майоров Александр Олегович
«Пензенский государственный университет», Пенза
Пенза, Россия (440026, г. Пенза, ул. Красная, 40)

ALGORITHM FOR FAST CALCULATION OF INVERSE SQUARE ROOT

Mayorov Alexander Olegovich
Penza state university, Penza
Penza, Russia (440026, Penza, Krasnaya str, 40)

Аннотация

В статье рассматривается алгоритм быстрого вычисления обратного квадратного корня числа. Алгоритм использует неординарные битовые манипуляции для приближенного вычисления обратного квадратного корня 32-битного числа с плавающей точкой.

Ключевые слова: числа с плавающей точкой, побитовые манипуляции, приближенные вычисления.

Annotation

The article discusses an algorithm for fast calculation of inverse square root of a number. The algorithm uses unorthodox bit manipulations to approximate the inverse square root of a 32-bit floating-point number.

Keywords: floating-point number, bit manipulations, approximation.

При решении задач визуализации двухмерной и трёхмерной компьютерной графики часто используются элементы аналитической геометрии, в том числе векторы. Многие алгоритмы требуют нормализации векторов для корректной работы. Так, чтобы нормализовать вектор $\vec{v} = (x, y, z)$ каждую координату необходимо разделить на $|\vec{v}| = \sqrt{x^2 + y^2 + z^2}$ или же умножить на величину обратную $|\vec{v}|$, то есть $\frac{1}{|\vec{v}|} = \frac{1}{\sqrt{x^2+y^2+z^2}}$. Поскольку для визуализации компьютерной графики необходимо большое количество вычислений (иногда вычислений в реальном времени), необходимо быстро вычислять значение величины $\frac{1}{\sqrt{\mathrm{n}}}$, где n - произвольное число с плавающей точкой.

Принцип работы алгоритма.

```
float Q_rsqrt(float number) {
    long i;
    float x2, y;
    const float threehalfs = 1.5F;
    x2 = number * 0.5F;
    y = number;
    i = *(long *) &y;
    i = 0x5f3759df - (i >> 1);
```

```
    y = *(float *) &i;
    y = y * (threehalfs - (x2 * y * y));
    // y = y * (threehalfs - (x2 * y * y));
    return y;
}
```

Реализация алгоритма из исходного кода Quake III: Arena (директивы препроцессора и комментарии опущены).

Описание работы функции

Стандарт языка C определяет лишь граничные размеры типов [1], но для упрощения анализа будем считать long 32-битным знаковым целым числом, а float 32-битным числом с плавающей точкой.

```
float Q_rsqrt(float number) {
```

Функция принимает и возвращает числа типа float.

```
    long i;
    float x2, y;
    const float threehalfs = 1.5F;
    x2 = number * 0.5F;
    y = number;
```

В начале функции определяются переменные необходимые для работы алгоритма.

```
    i = *(long *) &y;
```

В этой строчке кода адрес y (переменной типа float) явно преобразуется к типу указатель на long, после полученный указатель разыменуется, и полученное значение сохраняется в i (переменной типа long). Эта операция позволяет побитово скопировать число типа float в число типа long.

Данная функция основывается на стандарте IEEE для чисел с плавающей точкой.

IEEE 754 – технический стандарт представления чисел с плавающей точкой, выпущенный в 1985 году Институтом инженеров электротехники и электроники (Institute of Electrical and Electronics Engineers).

В соответствии с IEEE 754, 32-битное число с плавающей точкой φ имеет следующее представление w в памяти [2]:

s	E								M																						
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

где s – знак числа (1 – если отрицательное число, 0 – если положительное), E – экспонента (показатель степени) (в виде целого числа со сдвигом 127 вправо), M – мантисса в нормализованном виде.

Таким образом, число равно

$$\varphi(w) = (-1)^s \left(1 + \frac{M}{2^{23}}\right) 2^{E-127}, \qquad (1)$$

где φ – число, w – представление числа φ в соответствии с IEEE 754.

Представление в памяти y:

s	E	M

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Представление в памяти i:

s																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Переменные y и i состоят из одинаковых битов, несмотря на тип данных. Следовательно, в переменной i хранится значение $M + 2^{23}E$ (при $s = 0$). Так, $i = M + 2^{23}E$.

```
i = 0x5f3759df - (i >> 1);
```

Как было показано выше,

$$y = (-1)^s \left(1 + \frac{M}{2^{23}}\right) 2^{E-127}, \qquad (2)$$

где y – значение одноименной переменной, s – знак числа (1 – если отрицательное число, 0 – если положительное), E – экспонента (показатель степени) (в виде целого числа со сдвигом 127 вправо), M – мантисса в нормализованном виде.

Мы полагаем $s = 0$, так как квадратный корень может быть взят только от положительного числа. Следовательно,

$$y = \left(1 + \frac{M}{2^{23}}\right) 2^{E-127}, \qquad (3)$$

$$\log_2 \mathrm{y} = \log_2 \left(\left(1 + \frac{M}{2^{23}}\right) 2^{E-127}\right), \qquad (4)$$

$$\log_2 \mathrm{y} = \log_2 \left(1 + \frac{M}{2^{23}}\right) + \log_2(2^{E-127}), \qquad (5)$$

$$\log_2 \mathrm{y} = \log_2 \left(1 + \frac{M}{2^{23}}\right) + E - 127, \qquad (6)$$

Известно, что для $\mathrm{x} \in [0; 1]$ $\log_2(1 + \mathrm{x}) \cong x$. Подобное приближение абсолютно верно в точках $x = 0$ и $x = 1$. Однако, рассмотрим $\log_2(1 + \mathrm{x}) \cong x + \mu$, где μ - поправка, которая может быть выбрана произвольно.

$$\log_2 \mathrm{y} = \frac{M}{2^{23}} + \mu + E - 127, \qquad (7)$$

$$\log_2 \mathrm{y} = \frac{M}{2^{23}} + \frac{2^{23}E}{2^{23}} + \mu - 127, \qquad (8)$$

$$\log_2 \mathrm{y} = \frac{M + 2^{23}E}{2^{23}} + \mu - 127, \qquad (9)$$

Пусть $\Gamma = \frac{1}{\sqrt{\mathrm{y}}}$. Следовательно,

$$\log_2 \Gamma = \log_2 \frac{1}{\sqrt{\mathrm{y}}} = \log_2 y^{-\frac{1}{2}} = -\frac{1}{2} \log_2 \mathrm{y}, \qquad (10)$$

Чтобы сохранить величину $\frac{1}{\sqrt{y}}$ с битами, соответствующими типу float, в переменную типа long, будем рассматривать $\Gamma = (-1)^{s'}\left(1+\frac{M'}{2^{23}}\right)2^{E'-127}$. Следовательно, $\log_2 \Gamma = \frac{M'+2^{23}E'}{2^{23}} + \mu - 127$ (аналогично с $\log_2 y$).

$$\log_2 \Gamma = -\frac{1}{2}\log_2 y, \qquad (11)$$

$$\frac{M'+2^{23}E'}{2^{23}} + \mu - 127 = -\frac{1}{2}\left(\frac{M+2^{23}E}{2^{23}} + \mu - 127\right), \qquad (12)$$

$$\frac{M'+2^{23}E'}{2^{23}} = -\frac{1}{2}\left(\frac{M+2^{23}E}{2^{23}} + \mu - 127\right) - (\mu - 127), \qquad (13)$$

$$\frac{M'+2^{23}E'}{2^{23}} = -\frac{1}{2}\left(\frac{M+2^{23}E}{2^{23}}\right) - \frac{3}{2}(\mu - 127), \qquad (14)$$

$$M' + 2^{23}E' = 2^{23}\frac{3}{2}(127-\mu) - \frac{1}{2}(M + 2^{23}E), \qquad (15)$$

$$M' + 2^{23}E' = 2^{23}\frac{3}{2}(127-\mu) - \frac{i}{2}, \qquad (16)$$

При $\mu = 0.0450465679168701171875$ [3], $2^{23}\frac{3}{2}(127-\mu) = 5.f3759df_{16} \times 16^7 = 5f3759df_{16}$.

$$\frac{i}{2} = i \gg 1, \qquad (17)$$

где $\gg$ – операция побитового сдвига вправо.

$$M' + 2^{23}E' = 5f3759df_{16} - (i \gg 1), \qquad (18)$$

Следовательно, в строчке `i = 0x5f3759df - (i >> 1);` в переменную i (типа long) сохраняется величина приблизительно равная $\frac{1}{\sqrt{y}}$ с битами, соответствующими типу float.

```
    y = *(float *) &i;
```

Принцип работы данной строчки аналогичен принципу работы строчки `i = *(long *) &y;`, описанной выше. Следовательно, переменной y хранится значение $\frac{1}{\sqrt{number}}$.

```
    y = y * (threehalfs - (x2 * y * y));
```

Метод Ньютона – итерационный численный метод для нахождения корня заданной функции.

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \qquad (19)$$

где x_n – приблизительный корень уравнения $f(x) = 0$, x_{n+1} – уточненный корень уравнения $f(x) = 0$, $f'(x)$ – производная функции $f(x)$ по переменной x.

Для $f(y) = \frac{1}{y^2} - x$, $f'(y) = -\frac{2}{y^3}$

$$y_{n+1} = y_n - \frac{f(y_n)}{f'(y_n)} \qquad (20)$$

$$y_{n+1} = y - \frac{\frac{1}{y^2} - x}{-\frac{2}{y^3}} = y + \frac{\frac{1}{y^2} - x}{\frac{2}{y^3}} = y + \left(\frac{1}{y^2} - x\right)\frac{y^3}{2} = \frac{3y}{2} - \frac{xy^3}{2}$$

$$= y\left(\frac{3}{2} - \frac{x}{2}y^2\right) \qquad (21)$$

```
    // y = y * (threehalfs - (x2 * y * y));
```

Количество итераций метода Ньютона может быть увеличено для повышения точности результата. Однако даже при одной итерации относительная погрешность не превышает 0.175228% [4].

```
    return y;
```

Функция возвращает найденное значение. Алгоритм завершается.

Учитывая вышеизложенное, приведенный алгоритм позволяет получить приближенное значение обратного квадратного корня с высокой точностью за сравнительно небольшое время. Данный алгоритм может быть применен при решении задач компьютерной графики, а также при работе с процессорами, имеющими ограниченный набор инструкций.

Список литературы

1. International Organization for Standardization, International Electrotechnical Commission. ISO/IEC 9899:1999 Committee Draft. // International Organization for Standardization, International Electrotechnical Commission, 2007, с. 21.
2. Institute of Electrical and Electronics Engineers Computer Society. IEEE Standard for Floating-Point Arithmetic // Institute of Electrical and Electronics Engineers, 2012, с. 9.
3. Charles McEniry. The Mathematics Behind The Fast Inverse Square Root Function Code, 2007, с. 15.
4. Chris Lomont. Fast Inverse Square Root // Indiana: Purdue University, 2003, с. 1