электронный журнал
МОЛОДЕЖНЫЙ НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51038.

УДК 16

Моделирование чисел с плавающей запятой и арифметических операций над ними в троичной симметричной системе счисления с использованием длинной арифметики

***Строганов Ю.В.**, магистр*
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Программное обеспечение ЭВМ и информационные технологии»

Научный руководитель: Ковтушенко А.П., к.т.н., доцент
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Программное обеспечение ЭВМ и информационные технологии»

Консультант: Горин С.В., к.т.н., доцент
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана
gorin@bmstu.ru

Троичная система счисления давно будоражила умы человечества, первые попытки сделать троичную машину были предприняты ещё в 18 веке Фаулером, но его деревянная машина оказалась неконкурентоспособной, более успешная попытка была осуществлена Бруснецовым в 1958 году с вычислительной машиной «Сетунь», но и после неё резкого развития и применения троичных систем не наблюдалось. Очередной приступ интереса к троичной системе счисления начался в 2010 году. Моделирование чисел с плавающей запятой в троичной системе счисления является перспективным направлением для определения преимуществ троичного представления чисел по сравнению с двоичным.

Счисления стандарт IEEE-754 определяет число с плавающей запятой в двоичной системе счисления. Этот стандарт выделяет в числах с плавающей запятой числа с основанием 2 и с основанием 10. Стандарт выделяет в числе с плавающей запятой бит знака, мантиссу, основание и порядок.

В операции перевода из порядка в мантиссу в числе с основанием 2 необходимо «сдвинуть» мантиссу в сторону старшего разряда, таким образом значение числа не изменится, а значение порядка уменьшится, что важно для операций сложения и вычитания. Также важна обратная операция — перевода разряда из мантиссы в порядок, что увеличивает значение порядка, но уменьшает мантиссу. В числах же с основанием 10 такая красота не наблюдается, вместо неё необходимо уменьшать/увеличивать порядок на 1 и умножать/делить мантиссу на 10.

Моделирование

Мантисса и порядок представляются целыми числами в троичной симметричной системе счисления. Также необходимо ограничить количество разрядов после запятой, которые требуется учесть (length_of). Бит (трит) знака для троичной симметричной системы счисления не нужен [1,2,3,4]. Наиболее интересен перевод в данный формат, поскольку есть 2 представления с основанием 3 и основанием 10. Классы для чисел с плавающей запятой с основанием 10 и основанием 3 различаются только названиями long3simFloat и long3_3simFloat соответственно.

```
class long3simFloat
{
public:
        long3simFloat(void);
        long3simFloat(long3simDigit x);
        long3simFloat(long3simDigit m,long3simDigit g, int len);
        long3simFloat(float x, int len=5);
        long3simFloat(int m, int g, int len=5);
        ~long3simFloat(void);

protected:
        int length_of;
        long3simDigit mantis, grade;

public:
        void setLength_of(int len);
        int getLength_of();
        void setMantis(int i);
        void setGrade(int i);

        void balance(long3simFloat &x, long3simFloat &y);

        char compare(long3simFloat &x, long3simFloat &y);

        long3simFloat sum(long3simFloat x, long3simFloat y);

        long3simFloat substruct(long3simFloat x, long3simFloat y);

        long3simFloat multi(long3simFloat x, long3simFloat y);

        long3simFloat division(long3simFloat x, long3simFloat y);
}
```

Перевод в число с основанием 10 осуществляется достаточно просто: число умножаем на 10 до тех пор, пока дробная часть не станет равной 0 (или будет достигнуто ограничение по размеру числа), а также увеличиваем на 1 значение порядка, после чего в

мантиссу заносим целое значение получившегося числа.

```
long3simFloat(float x, int len)
{
	int g=0, m=0;
	while(x - (int)x !=0)
	{
		g++;
		x *= 10;
	}
	m = (int)x;
	mantis = long3simDigit(m);
	grade = long3simDigit(g);
	length_of = len;
}
```

Перевод в число с основанием 3 осуществляется не менее интересно и захватывающе. На начальном этапе необходимо разделить целую и дробные части числа, после чего требуется занести в мантиссу целую часть, а при переводе дробной части действовать следующим образом: умножать дробную часть на основание (3), брать целую часть получившегося числа, добавить к ней добавочное значение (на первой итерации это 0), разбить её на то, что добавляется слева к мантиссе и на добавочное значение в следующую итерацию, после чего взять дробную часть от полученного числа и продолжать до тех пор, пока дробная часть не станет равна 0 или не будет достигнута заданная длина.

```
long3_3simFloat(float x, int len)
{
	length_of = len;
	int g=0, m=0;
	m = (int)x;
	mantis = long3simDigit(m);
	float xx = x - m;
	char here = 0, there =0;
	while(xx!=0 && len>0)
	{
		xx*=3;
		char m2 = (int)xx + there;
		here = m2;
		switch(m2)
		{
			case 2: here = 1; there = -1; break;
			case 3: here = 1; there = 1; break;
			case -2: here = -1; there = 1; break;
			case -3: here = 1; there = 1; break;
			default: here = m2; there = 0; break;
		}
		mantis.addToLeft(here);
```

```
            xx = xx - m2;
            len--;
            g++;
        }
        grade = long3simDigit(g);
}
```

Операция сравнения

Сравнение осуществимо для чисел с одинаковым основанием, т. е. сравнение чисел с плавающей запятой с основанием 3 и основанием 10 некорректно. Для чисел с плавающей запятой необходимо предварительно выровнять мантиссы, чтобы были одинаковые длины. После чего необходимо сравнить порядки чисел, если порядок первого числа больше порядка второго, то первое число больше второго по модулю, далее необходимо сравнить знаки чисел (как и в случае с целыми числами знак числа определяется знаком старшего разряда). Если порядки равны, необходимо сравнить мантиссы [1].

```
char compare(long3simFloat &x, long3simFloat &y)
{
        balance(x,y);
        char res =0;
        char res2 = x.grade.compare(y.grade);
        if(res2==0)
        {
            res = x.mantis.compare(x.mantis,y.mantis);
        }
        else
        {
            res = res2;
            if(x.mantis.compare0() <0 ) res = 1;
            if(y.mantis.compare0() <0 ) res = -1;
        }
        return res;
}
```

Арифметические операции

Для реализации операций умножения и деления нет необходимости каким-либо способом предварительно менять значения порядка и мантиссы. Необходимо осуществить вычитание/сложение порядков и соответствующую операцию деления/умножения для мантисс.

```
long3simFloat multi(long3simFloat x, long3simFloat y)
{
        long3simFloat res = long3simFloat();
```

```
        long3simDigit rr = long3simDigit();
        res.mantis = rr.multiplication(x.mantis,y.mantis);
        res.grade = rr.sum(x.grade,y.grade);
        rr.~long3simDigit();
        return res;
}
```

и деления

```
long3simFloat division(long3simFloat x, long3simFloat y)
{
        long3simFloat res = long3simFloat();
        long3simDigit rr = long3simDigit();
        res.grade = rr.substruct(x.grade,y.grade);

        int len = length_of;
        res.mantis = rr.division(x.mantis,y.mantis,len);
        int lenRes = length_of - len;
        long3simDigit r2 = long3simDigit(lenRes);
        res.grade.printIt();
        res.grade.substruct(r2);

        rr.~long3simDigit();
        r2.~long3simDigit();
        return res;
}
```

Для реализации операций сложения и вычитания необходимо предварительно уравнять значения порядков, что можно сделать уменьшив больший порядок до значения меньшего, после чего в порядок результата заносится значение порядка, а в мантиссу результата заносится результат операции между мантиссами чисел.

Сложение:

```
long3simFloat sum(long3simFloat x, long3simFloat y)
{
        long3simFloat res = long3simFloat();
        long3simDigit rr = long3simDigit();
        balance(x,y);
        res.mantis = rr.sum(x.mantis,y.mantis);
        res.grade = y.grade;
        rr.~long3simDigit();
        return res;
}
```

Вычитание:

```
long3simFloat substruct(long3simFloat x, long3simFloat y)
{
        long3simFloat res = long3simFloat();
        long3simDigit rr = long3simDigit();
        balance(x,y);
        res.mantis = rr.substruct(x.mantis,y.mantis);
        res.mantis.delExcessZeros();
```

```
	res.grade = y.grade;
	rr.~long3simDigit();
	return res;
}
```

Алгоритмы этих операции для чисел с плавающей запятой с основаниями 10 и 3 не отличаются. Как видно из описаний различия в алгоритмах сложения и вычитания для чисел с плавающей запятой с основанием 3 и основанием 10 заключается в способе «балансировки», т. е. каким образом переводят из порядка в мантиссу и наоборот. Для основания 3 это сдвиг вправо или влево, а для основания 10 это умножение или деление мантиссы на 10.

Для основания 10

```
void long3simFloat::balance(long3simFloat &x, long3simFloat &y)
{
	long3simDigit rr = long3simDigit();
	char xx = rr.compare(x.grade,y.grade);
	long3simDigit req = long3simDigit(10);
	if(xx!=0)
	{
		if(xx==-1)
		{
			while(xx!=0)
			{
				x.grade.dec();
				x.mantis.multiplication(req);
				xx = rr.compare(x.grade,y.grade);
			}
		}
		else
		{
			while(xx!=0)
			{
				y.grade.dec();
				y.mantis.multiplication(req);
				xx = rr.compare(x.grade,y.grade);
			}
		}
	}
	req.~long3simDigit();
	rr.~long3simDigit();
}
```

И для основания 3:

```
void long3_3simFloat::balance(long3_3simFloat &x, long3_3simFloat &y)
{
	long3simDigit rr = long3simDigit();
	char xx = rr.compare(x.grade,y.grade);
	if(xx!=0)
```

```
	{
		if(xx==-1)
		{
			while(xx!=0)
			{
				x.grade.dec();
				x.mantis.moveRight(1);
				xx = rr.compare(x.grade,y.grade);
			}
		}
		else
		{
			while(xx!=0)
			{
				y.grade.dec();
				y.mantis.moveRight(1);
				xx = rr.compare(x.grade,y.grade);
			}
		}
	}
	rr.~long3simDigit();
}
```

Заключение

Предложенный метод моделирования чисел с плавающей запятой позволяет достигать произвольной точности вычислений, что даёт возможность опытным путём определять необходимый размер для чисел с плавающей запятой в троичной симметричной системе счисления для различный областей применения.

Список литературы

1. Рамиль Альварес Х. Алгоритмы троичной арифметики. М.: Фонд «Новое тысячелетие», 2012. 20 с.
2. Рамиль Альварес Х. Троичный алгоритм извлечения квадратного корня // Программные системы и инструменты. Тематический сборник. 2010. № 11. С. 98-106.
3. Рамиль Альварес Х. Деление целых чисел в троичной симметричной системе // Программные системы и инструменты. Тематический сборник. 2011. № 12. С. 228-234.
4. Программные системы и инструменты. Тематический сборник № 12. 2011. С. 228-234.