Учебные материалы для 2 курса

Особенности чисел с плавающей точкой

Множество значений, представимых в типах `float`, `double` и `long double` кроме обычных конечных значений содержит три специальных значения `NaN`, `-Inf`, `Inf`.

Значение `NaN`

Значение `NaN` (Not-a-Number) используется для того, чтобы сигнализировать о том, что результат некоторой операции не может быть вычислен из-за неопределенностей различного рода.

Например, операция `0.0/0.0` даст результатом `NaN`.

На самом деле в вещественных типах представимо целое семейство значений `NaN`, дополнительная информация о конкретном `NaN`-значении может быть использована как код ошибки.

Все функции перевода из строкового представления в вещественное значение, такие как `*scanf`, `strtod` и т. п. распознают строку `NAN` (независимо от регистра букв) и возвращают значение `NaN`.

Все функции перевода из вещественного значения в строковое представление, такие как `*printf` распознают значение `NaN` и выводят его в виде строки `nan`.

Результат операции с числами с плавающей точкой, такой как сложение, умножение и т. д. равен `NaN`, если один из аргументов операции равен `NaN`.

Если один из операндов операции сравнения равен `NaN`, операции сравнения дают следующий результат:

==	0
!=	1
<	0
<=	0
>	0
>=	0

Чтобы проверить вещественное значение на принадлежность к классу `NaN` можно использовать функции `fpclassify` или

isnan.

```
#include <math.h>

//...
    if (fpclassify(value) == FP_NAN) {
        // значение value является NaN
    }

// или другой вариант
    if (isnan(value)) {
        // значение value является NaN
    }
```

В силу свойств значения `NaN` мы будем полагать, что последовательность чисел с плавающей точкой не может быть упорядочена, если в ней содержится элемент `NaN`

Значения `-Inf`, `Inf`

Эти значения представляют результат "бесконечность" который может возникать при выполнениях различных операций с плавающей точкой.

Например, `1.0/0.0` дает результат `Inf`, а `-1.0/0.0` дает результат `-Inf`.

Все функции перевода из строкового представления в вещественное значение, такие как `*scanf`, `strtod` и т. п. распознают строку `[+|-]INF` (независимо от регистра букв) и возвращают значение `Inf` с соответствующим знаком.

Все функции перевода из вещественного значения в строковое представление, такие как `*printf` распознают значение `Inf` и выводят его в виде строки `inf` с соответствующим знаком.

Значения `Inf`, `-Inf` в операциях сравнения ведут себя естественным образом, например, следующие условия истинны:

```
  inf == inf
  !(inf != inf)
  inf >= inf
  inf <= inf
  !(inf > inf)
  !(inf < inf)
```

Значение `Inf` больше любого конечного значения, а значение `-Inf` меньше любого конечного значения.

Однако, обратите внимание, что

```
  !(inf - inf == 0)
```

Так как `inf - inf` дает в результате `NaN`.

Чтобы проверить вещественное значение на принадлежность к классу `Inf` можно использовать функции `fpclassify` или

isinf.

```
#include <math.h>

//...
    if (fpclassify(value) == FP_INFINITE) {
        // значение value является Inf (любого знака)
    }

// или другой вариант
    if (isinf(value) < 0) {
        // значение value является -inf
    }
    if (isinf(value) > 0) {
        // значение value является +inf
    }
```

Отрицательный `0`

Множество значений вещественных чисел содержит два нуля: положительный и отрицательный. Когда в результате некоторых вычислений (например, при умножении или делении) получается нулевой результат, его знак вычисляется по обычным правилам и сохраняется. Поэтому в результате может получиться как обычное значение `0.0`, так и отрицательное `-0.0`. Эти значения равны друг другу и, соответственно, больше всех отрицательных чисел и меньше всех положительных чисел.

Если при операции деления делитель равен нулю, то его знак учитывается при вычислении знака получающейся бесконечности. Таким образом, `1.0/-0.0 == -inf`, `-1.0/-0.0 == inf`.

Last modified: Friday, 21-Jun-2013 16:47:24 MSK
Alexander Chernov