Курс «Алгоритмы и алгоритмические языки»
1 семестр 2013/2014

Лекция 10

Операции над числами с плавающей точкой

- По определению:

$$x +_{FP} y = Round(x + y)$$

$$x \times_{FP} y = Round(x \times y)$$

где $Round()$ означает округление

- Выполнение операции
  - Сначала вычисляется точный результат (получается более длинная мантисса, чем запоминаемая, иногда в два раза)
  - Потом фиксируется исключение (например, переполнение)
  - Потом результат округляется, чтобы поместиться в поле $frac$

Умножение чисел с плавающей точкой

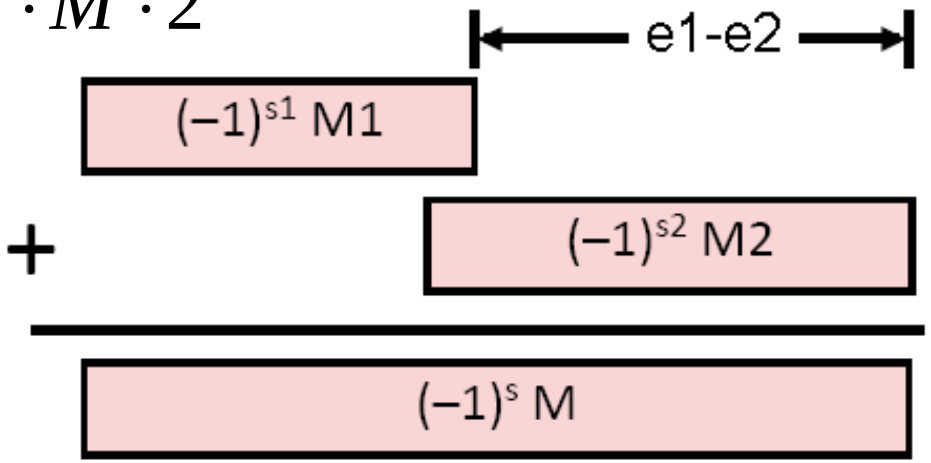
- $(-1)^{s1} \cdot M1 \cdot 2^{e1} \times (-1)^{s2} \cdot M2 \cdot 2^{e2}$
- Точный результат $(-1)^{s} \cdot M \cdot 2^{e}$
  - Знак s $s1 \wedge s2$
  - Значащие цифры M $M1 \times M2$
  - Порядок e $e1 + e2$
- Преобразование
  - Если $M \geq 2$, сдвиг M вправо с одновременным увеличением e
  - Если e не помещается в поле exp, переполнение
  - Округление M, чтобы оно поместилось в поле $frac$
- Основные затраты на перемножение мантисс

Сложение чисел с плавающей точкой

- $(-1)^{s1} \cdot M1 \cdot 2^{e1} + (-1)^{s2} \cdot M2 \cdot 2^{e2}$

  Пусть $e1 > e2$

- Точный результат $(-1)^{s} \cdot M \cdot 2^{e}$
  - Знак s и значащие цифры M вычисляются как показано на рисунке
  - Порядок суммы – $e1$



- Преобразование
  - Если $M \geq 2$, сдвиг M вправо с одновременным увеличением e
  - Если $M < 1$, сдвиг M влево на k позиций с одновременным вычитанием k из e
  - Если e не помещается в поле exp, переполнение
  - Округление M, чтобы оно поместилось в поле $frac$

Плавающие типы языка Си

`float`, `double`, `long double`

- ***Операции над данными с плавающей точкой***.
  - *Одноместные*: изменение знака («одноместный минус»: **–**), одноместный плюс (**+**).
  - *Двухместные*: сложение (**+**), вычитание (**–**), умножение (*****), деление (**/**).
- ***Порядок выполнения арифметических операций в выражениях (приоритет).***
  - самый низкий приоритет у двуместных **+** и **–**,
  - более высокий приоритет у двуместных ***** и **/**,
  - еще более высокий приоритет у одноместных **+** и **–**.
  - В выражениях без скобок операции с более высоким приоритетом выполняются раньше.
  - Скобки позволяют изменить порядок выполнения операций.

Пример 1. Вычисление суммы 5 чисел типа **`float`**
(мантисса – 6 десятичных цифр, порядок – 2 десятичных цифры):

$0.231876*10^{02} + 0.645391*10^{-03} + 0.231834*10^{-01} + 0.245383*10^{-02} + 0.945722*10^{-03} =$

a) $\mathbf{0.231876*10^{02} + 0.645391*10^{-03} + 0.231834*10^{-01} + 0.245383*10^{-02} + 0.945722*10^{-03} = 0.2321\underline{4}7*10^{02}}$;

23.1876 + 0.000645391 = 23.188245391 = 23.1882 = 0. $231882*10^{02}$;

23.1882 + 0.0231834 = 23.2113834 = 23.2114 = $0.232114*10^{02}$;

23.2114 + 0.00245383 = 23.21385383 = $23.2138*10^{02}$;

23. 2138 + 0.000945722 = 23.214745722 = 23.2147 = $0.232147*10^{02}$;

b) $\mathbf{0.645391*10^{-03} + 0.9457*10^{-03} + 0.245383*10^{-02} + 0.231834*10^{-01} + 0.231876*10^{02} = 0.2321\underline{5}7*10^{02}}$;

0.000645391 + 0.000945722 = 0.001591113 = 0.00159111 = 0. $159111*10^{-02}$;

0.00159111 + 0.00245383 = 0.00494493 = 0. $494493*10^{-02}$;

0.00494493 + 0.0231834 = 0.02812833 = 0.0281283 = 0. $281283*10^{-01}$;

0.0281283 + 23.1876 = 23.2157283 = 23.2157 = $0.232157*10^{02}$;

Пример 2. Вычисление разности плавающих чисел

(мантисса – 6 десятичных цифр, порядок – 2 десятичных цифры):

$0.238617*10^{02} – 0.238616*10^{02} + 0.645391*10^{04} – 0.645392*10^{04} + 0.845791*10^{00} – 0.835790*10^{00} =$

a) $0.238617*10^{02} – 0.238616*10^{02} + 0.645391*10^{04} – 0.645392*10^{04} + 0.845791*10^{00} – 0.835790*10^{00} =$ **$0.100000*10^{-05}$**

$0.238617*10^{02} – 0.238616*10^{02} = 23.8617 – 23.8616 = 0.0001 =$ **$0.100000*10^{03}$**

$0.100000*10^{-03} + 0.645391*10^{04} = 0.0001 + 6453.91 = 6453.9101 =$ **$0.645391*10^{04}$**

$0.645391*10^{04} – 0.645392*10^{04} = – 0.000001*10^{04} =$ **$– 0.100000*10^{-01}$**

$– 0.100000*10^{-01} + 0.845791*10^{00} = – 0.01 + 0.845791 = 0.835791 *10^{00}$

$0.835791 *10^{00} – 0.835790*10^{00} =$ **$0.000001*10^{00} = 0.100000*\underline{10^{-05}}$**

b) $0.238617*10^{02} + 0.645391*10^{04} + 0.845791*10^{00} – (0.238616*10^{02} + 0.645392*10^{04} + 0.835790*10^{00}) =$ **$0.100000*10^{00}$**

$0.238617*10^{02} + 0.645391*10^{04} = 23.8617 – 6453.91 = 6478.6 =$ **$0.647777*10^{04}$**

$0.647777*10^{04} + 0.845791*10^{00} = 6477.77 + 0.845791 = 6478.615791 =$ ***$0.647862*10^{04}$***

$0.238616*10^{02} + 0.645392*10^{04} = 23.8616 + 6453.92 = 6477.7816 =$ **$6477.78*10^{04}$**

$6477.78*10^{04} + 0.835790*10^{00} = 6477.78 + 0.835790 = 6478.61579 =$ ***$0.647852*10^{04}$***

$0.647862*10^{04} – 0.647852*10^{04}=$ **$0.000010*10^{04} = 0.100000*\underline{10^{-00}}$**

Выводы

(1) ***При вычислении суммы чисел с одинаковыми знаками*** необходимо упорядочить слагаемые по возрастанию и складывать, начиная с наименьших слагаемых.

(2) ***При вычислении суммы чисел с разными знаками*** необходимо сначала сложить все положительные числа, потом – все отрицательные числа и в конце выполнить одно вычитание.

(3) ***Вычитание*** (сложение чисел с противоположными знаками) ***часто приводит к потере точности***, которая у чисел с плавающей точкой определяется количеством значащих цифр в мантиссе (при вычитании двух близких чисел мантисса «исчезает», что ведет к резкой потере точности).
Итак, ***чем меньше вычитаний, тем точнее результат*.**

Значащими цифрами числа с плавающей точкой называются все цифры его мантиссы за исключением нулей, стоящих в ее конце. Например, у числа 0.**6700089**0000 * 10^3 все цифры, выделенные жирным шрифтом, значащие. При вычитании двух близких чисел почти все значащие цифры пропадают. Например, 0.67000890 * 10^3 - 0.67000880 * 10^3 = 0.00000010 * 10^3 = 0.10 * 10^{-4}. Таким образом, у результата всего одна значащая цифра, хотя у операндов было по 7 значащих цифр.

Поразрядные операции

- **&** (поразрядное И)
- **|** (поразрядное включающее ИЛИ)
- **^** (поразрядное исключающее ИЛИ))
- **<<** (сдвиг влево)
- **>>** (сдвиг вправо)
  - Беззнаковое число – заполнение нулями
  - Знаковое число – заполнение значением знакового разряда («арифметический сдвиг») или нулями («логический сдвиг»)
- **~** (дополнение до 1, или *инверсия*)

Тип «степень множества» (булеан)

- Пусть U – множество. Множество всех подмножеств множества U называется *степенью множества* U.
- Пусть B – степень конечного множества U. Тогда $|B| = 2^{|U|}$.
- *Характеристической функцией* χ_C подмножества C множества U называется функция, принимающая значение 1 на элементах U, входящих в состав C, и значение 0 на остальных элементах U.
- Множества удобно задавать через их характеристические функции. При этом в зависимости от количества элементов базового множества U его характеристическая функция, кодированная битами целого типа, может иметь тип

  `unsigned char` (если $|U| \le 8$)

  `unsigned int` (если $|U| \le 32$)

  `unsigned long long` (если $|U| \le 64$)

Тип «степень множества»

- **Пример**. Пусть $U = \{r, o, y, g, c, b, v, w\}$. Тогда его подмножества задаются переменными типа **`unsigned char`**: $|B| = 2^{|U|}$

  $\{\}$ задается значением **`00000000`**,

  $\{r, y, g\}$ – значением **`10110000`**

  $\{r, o, y, g, c, b, v, w\}$– значением **`11111111`**

буквы r, o, y, g, c, b, v, w являются первыми буквами семи цветов спектра и белого цвета:

$r - red$	$c - cyan$
$o - orange$	$b - blue$
$y - yellow$	$v - violet$
$g - green$	$w - white$

Реализация операций над множествами с помощью поразрядных операций

- **&** (поразрядное И) соответствует пересечению множеств:

  $B = \{r, y, g, b, w\}$, $C = \{r, o, y, g, v\}$

  χ_B **= 10110101,** χ_C **= 11110010**

  χ_B **&** χ_C **= 10110101 & 11110010 = 10110000**

  $B \cap C = \{r, y, g\}$

Реализация операций над множествами с помощью поразрядных операций

- **|** (поразрядное включающее ИЛИ) соответствует объединению множеств:

$B = \{r, y, g, b, w\}$, $C = \{r, o, y, g, v\}$

χ_B **= 10110101,** χ_C **= 11110010**

χ_B **|** χ_C **= 10110101 | 11110010 = 11110111**

$B \cup C = \{r, o, y, g, b, v, w\}$

Реализация операций над множествами с помощью поразрядных операций

- ~ (инверсия) соответствует дополнению до множества U:

  $B = \{r, y, g, b, w\}$

  χ_B **= 10110101**

  $\chi_{\sim B}$ **= 01001010**

  $\sim B = \{o, c, v\}$

Структуры

- Структура – это совокупность нескольких переменных, часто разных типов, сгруппированных под одним именем для удобства
  - Переменные, перечисленные в объявлении структуры, называются ее *полями*, *элементами*, или *членами*.
- ***Объявление структуры***:

  struct *метка структуры* **{** *поля структуры* **};**

```
struct point
{
  int x;
  int y;
} f, g;
struct point h, center = {32, 32};
```

Структуры

- Поля структуры могут иметь любой тип, например, тип массива или тип другой структуры

```
struct rect
{
  struct point pt1;
  struct point pt2;
};
```

- ***Инициализация структуры***:

```
struct rect r = {.pt1 = {4, 4},
                 .pt2 = {7, 6}};
/* Остальные элементы – нулевые */
struct rect r2 = {.pt2.x = 5};
```

- Размер структуры в общем случае **не равен** сумме размеров ее элементов (**выравнивание**)

Структуры

- Доступ к полям структуры: операция точка "**.**"
  - **f.x, g.y, r.pt1.x**
- Присваивание структур целиком: **f = g;**
- Массивы структур

```
#define NRECT 15
/* Первый прямоугольник вокруг 0, 0 */
struct rect rectangles[NRECT]
= {{-1, -1, 1, 1}};
/* Последний прямоугольник – большой */
#define BOUND 1024
struct rect bounded_rectangles[NRECT]
= {[NRECT-1] = {-BOUND, -BOUND,
                 BOUND,  BOUND}};
```

Указатели на структуры

- ```
  struct rect r = {.pt1 = {4, 4},
                   .pt2 = {7, 6}};
  struct rect *pr = &r;
  ```
- Доступ к полям структуры через указатель:

  ```
  pr->pt1 (= (*pr).pt1), pr->pt2.x
  ```
- Адресная арифметика:

  ```
  struct rect *pr = &bounded_rectangles[0];
  while (pr->pt1.x != -BOUND)
      pr++;
  ```
  ```

Составные инициализаторы структур (C99)

- ```
  struct rect r;
  r = (struct rect) { {4, 4},
                      {7, 6}};
  ```
- Можно передавать и указатель:

```
double area (struct rect *r) {
  return (r->pt1.x – r->pt2.x)
          * (r->pt1.y – r->pt2.y);
}
double da
= area (& (struct rect) {{4, 4}, {7, 6}});
```
```