14. Pontos flutuantes

DIM0321

2015.1

Sumário

Uso

Fato

Muitas aplicações precisam tratar dados numéricos não inteiros

- Simulações físicas
- Cálculos numéricos com números reais
- Movimentação de robôs ou de personagens em um ambiente

- Para esse casos, usaremos uma categoria de tipo dito **pontos flutuantes**
- Em C:
  - `float`
  - `double`
  - `long double`

Padrão IEEE 754

Inspiração

Baseado na notação científica

$$300 = 3 * 10^2, 31, 25 = 3, 125x10^1, 1/3 = 3, 333...x10^-$$

Precisão

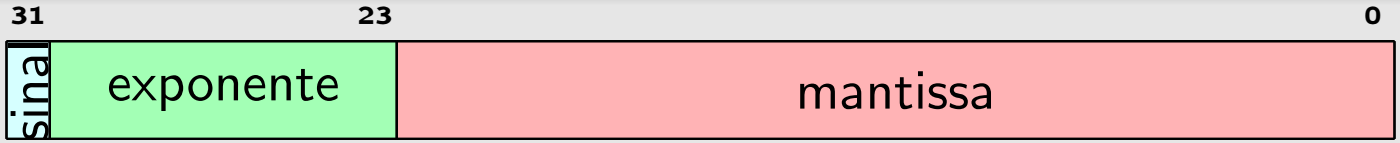
- O padrão inicial define 2 tipos:
  - precisão simples
  - precisão dupla
- `http://en.wikipedia.org/wiki/IEEE_floating_point#Representation_and_encoding_in_memory`

Precisão simples : `float`

Definição

- `float` são pontos flutuantes de precião simples
- 32 bits

Estrutura



Valor

$v = (-1)^s(1.m) \times 2^{e-137}$

Precisão dupla : `double`

Definição

- precisão dupla / `binary64`
- 64 bits

Estrutura

Valor

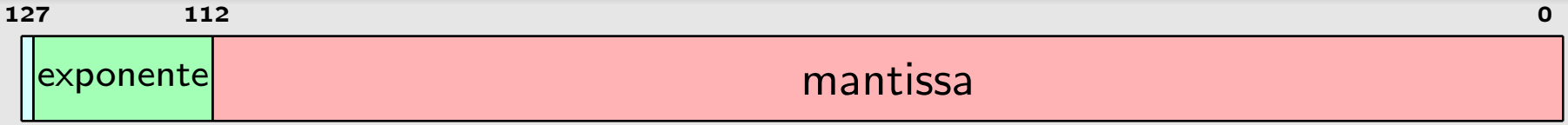
$v = (-1)^{s}(1.m) \times 2^{e-1023}$

Precisão quadrupla: `long double`

Definição

- precisão quadrupla / `binary128`
- 128 bits

Estrutura



Valor

$v = (-1)^{s}(1.m) \times 2^{e-\mathbf{16383}}$

Valore especiais

nan

- `nan` : *not a number*
- `nan` $\neq$ `nan` !
- `0. / 0.` $\rightarrow$ `nan`

Infinidade(s)

- Funciona mais ou menos como esperado
- `1. / 0.` $\rightarrow$ `inf`
- `-1. / 0.` $\rightarrow$ `-inf`

Exemplo com valores especiais

```c
#include <stdio.h>
#include <math.h>
int main(void)
{
     float f = 0. / 0. ;;

     if (isnan(f)) printf("It's a nan\n");
     if (f == f) printf("It's equal to nan\n");
     if ((unsigned) f == (unsigned) f) printf("foo\n");
     return 0;
}
```

```
It's a nan
foo
```

Resumo

Tipo	# bits	# bytes	Formato	Valor max	Valor min
`float`	32	4	%f / %e	`FLT_MAX`	`FLT_MIN`
`double`	64	8	%lf / %le	`DBL_MAX`	`DBL_MIN`
`long double`	128	16	%Lf /% Le	`LDBL_MAX`	`LDBL_MIN`

Tipo	Sinal (da mantissa)	Exponente	Mantissa
`float`	31	30-23 (8)	22-0 (23)
`double`	63	62-52 (11)	51-0 (52)
`long double`	127	126-112 (15)	111-0 (112)

Acessar as definições

```c
#include <stdio.h>
#include <float.h>  /* FLT_MIN, FLT_MAX */
int main(void)
{
    printf("float, size: %lu, min = %e, max = %e\n",
           sizeof(float), FLT_MIN, FLT_MAX);
    printf("double, size: %lu, min = %le, max = %le\n",
           sizeof(double), DBL_MIN, DBL_MAX);
    printf("long double, size: %lu, min = %Le, max = %Le\n",
           sizeof(long double), LDBL_MIN, LDBL_MAX);
    return 0;
}
```

float	size: 4	min = 1.175494e-38	max = 3.402823e+38
double	size: 8	min = 2.225074e-308	max = 1.797693e+308
long double	size: 16	min = 3.362103e-4932	max = 1.189731e+4932

O tipo float

```c
#include <stdio.h>

int main(void)
{
    float x = 3.45f; // float x = (float) (double) 3.45;
    float y = -190.444f;
    /* Notação científica */
    printf("x = %e, y = %e \n", x, y);
    /* Notação decimal */
    printf("x = %f, y = %f \n", x, y);
    /* Só as 3 primeiras decimais */
    printf("x = %.5f, y = %.3f", x, y);
    return 0;
}
```

x = 3.450000e+00 y = -1.904440e+02
x = 3.450000 y = -190.444000
x = 3.45000 y = -190.444

O tipo double

```c
#include <stdio.h>
#include <float.h>  /* DBL_MIN, DBL_MAX */

int main(void)
{
    double m = 3.45 * -190.444;
    double d = 3.45 / -190.444;
    printf("m = %le, d = %le \n", m, d);
    printf("m = %lf, d = %lf \n", m, d);
    printf("m = %.3lf, d = %.3lf", m, d);
    return 0;
}
```

m = -6.570318e+02 d = -1.811556e-02
m = -657.031800 d = -0.018116
m = -657.032 d = -0.018

O tipo long double

```c
#include <stdio.h>

int main(void)
{
    long double m = 3.45 * -190.444;
    long double d = 3.45 / -190.444;
    printf("m = %Le, d = %Le \n", m, d);
    printf("m = %Lf, d = %Lf \n", m, d);
    printf("m = %.3Lf, d = %.3Lf", m, d);
    return 0;
}
```

m = -6.570318e+02 d = -1.811556e-02
m = -657.031800 d = -0.018116
m = -657.032 d = -0.018

Representações especiais

```c
#include <stdio.h>
#include <float.h>  /* FLT_MIN, FLT_MAX */

int main(void)
{
    float x = 3.45f;
    float y = .0f;
    float d1 = x / y;
    float d2 = -x / y;
    float d3 = y / y;
    printf("d1 = %e, d2 = %e, d3 = %e\n", d1, d2, d3);

    /* The following gives a division by zero error.
    int c = 1 / 0;
    printf("%d\n", c);
    */
    return 0;
}
```

d1 = inf d2 = -inf d3 = -nan

Finitude

- A representação decimal dos números reais ($\in \mathbb{R}$) pode ter um número infinito de dígitos
- A representação computacional dos números é finita
- Arredondamentos são necessariamente efetuados
- O acúmulo de arredondamentos torna os cálculos imprecisos
- Para calcular se dois valores float são iguais, devemos em certos casos nos satisfazer em testar se eles são "próximos" o suficiente.

Problemas de precisão

```c
#include <stdio.h>

int main(void)
{
    float x = 0.1;

    printf("Value of x = %.9f, 4 * x = %.9f\n", x, 4 * x);
    return 0;
}
```

Value of x = 0.100000001 4 * x = 0.400000006

Regras de conversão (bis)

1. Se um operando for `long double`, o outro é convertido em `long double`.
2. Senão, se um for `double` , o outro é convertido em ~ double~ .
3. Senão, se um for `float` , o outro é convertido em `float` .

Conversão explicita

- É melhor explicitar conversões de tipos (*casts*) com `( )` antes da expressão.
- Operações de conversão têm uma precedência maior do que os operadores binários.
- Assim `(float) x / 1` ≡ `((float) x) / 1`

```
1  #include <stdio.h>
2  int main(void)
3  {
4      int i1 = 4, i2 = 10;
5      float f;
6
7      f = i1 / i2;
8      printf("f = %f.\n", f);
9
10     f = (float) i1 / i2;
11     printf("f = %f.\n", f);
12
13     f = i1 / (float) i2;
14     printf("f = %f.\n", f);
15
16     f = (float) (i1 / i2);
17     printf("f = %f.\n", f);
18     return(0);
19 }
```

Conversões

De inteiros

- `int` $\rightarrow$ `float`: A parte decimal fica nula.
- Todo inteiro tem uma representação exata em ponto flutuante.
- Assim se `int x`, `x == (int) (float) x`

Para inteiros

- `float` $\rightarrow$ `int` : A parte decimal fica truncada.

Biblioteca de métodos matemáticos math.h

Confira `man math.h` para mais detalhes.

```
1  double cos( double );
2  double sin( double );
3  double tan( double );
4  double acos( double );
5  double asin( double );
6  double atan( double );
7  double exp( double );      /* exp em base e  */
8  double exp2( double );     /* exp em base 2  */
9  double log( double );      /* log em base e  */
10 double log2( double );     /* log em base 2  */
11 double log10( double );    /* log em base 10 */
12 // Funções exponenciais e logarítmicas para float
13 float expf( float );
14 float log10f( float );
15 double fabs( double );          /* valor absoluto */
16 double pow( double, double );  /* potência */
17 double sqrt(double);            /* raiz quadrada */
18 double ceil(double);            /* arredondamento superior */
19 double floor(double);           /* arredondamento inferior */
20 double round(double);           /* arredondamento */
21 double fmax(double, double);    /* maior dos argumentos */
22 double fmin(double, double);    /* menor dos argumentos */
```

Perguntas ?

http://dimap.ufrn.br/~richard/dim0321