

14. Pontos flutuantes

DIM0321

2015.1

Outline

① Introdução

② Descrição

③ Exemplos

DIM0321

14. Pontos flutuantes

2015.1 1 / 25

DIM0321

14. Pontos flutuantes

2015.1 2 / 25

① Introdução

② Descrição

③ Exemplos

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

DIM0321

14. Pontos flutuantes

2015.1 3 / 25

DIM0321

14. Pontos flutuantes

2015.1 4 / 25

1 Introdução

2 Descrição

3 Exemplos

Padrão IEEE 754

Inspiração

Baseado na notação científica

$$300 = 3 * 10^2, 31,25 = 3,125 \times 10^1, 1/3 = 3,333... \times 10^{-1}$$

Precisão

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

DIM0321

14. Pontos flutuantes

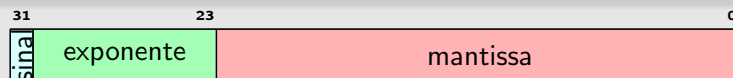
2015.1 5 / 25

Precisão simples : float

Definição

- float são pontos flutuantes de precisão simples
- 32 bits

Estrutura



Valor

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

DIM0321

14. Pontos flutuantes

2015.1 7 / 25

DIM0321

14. Pontos flutuantes

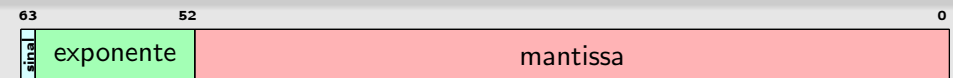
2015.1 6 / 25

Precisão dupla : double

Definição

- precisão dupla / binary64
- 64 bits

Estrutura



Valor

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

DIM0321

14. Pontos flutuantes

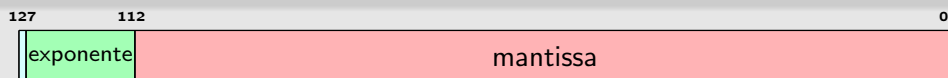
2015.1 8 / 25

Precisão quadrupla: long double

Definição

- precisão quadrupla / binary128
- 128 bits

Estrutura



Valor

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

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

```
1 #include <stdio.h>
2 #include <math.h>
3 int main(void)
4 {
5     float f = 0. / 0. ;;
6
7     if (isnan(f)) printf("It's a nan\n");
8     if (f == f) printf("It's equal to nan\n");
9     if ((unsigned) f == (unsigned) f) printf("foo\n");
10    return 0;
11 }
```

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)
double		63	62-52 (11)
long double		127	126-112 (15)

1 Introdução

2 Descrição

3 Exemplos

Acessar as definições

```
1 #include <stdio.h>
2 #include <float.h> /* FLT_MIN, FLT_MAX */
3 int main(void)
4 {
5     printf("float, size: %lu, min = %e, max = %e\n",
6           sizeof(float), FLT_MIN, FLT_MAX);
7     printf("double, size: %lu, min = %le, max = %le\n",
8           sizeof(double), DBL_MIN, DBL_MAX);
9     printf("long double, size: %lu, min = %Le, max = %Le\n",
10           sizeof(long double), LDBL_MIN, LDBL_MAX);
11     return 0;
12 }
```

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 = 1.797693e+308	max = 6.953187e-310

DIM0321

14. Pontos flutuantes

2015.1 13 / 25

O tipo float

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     float x = 3.45f; // float x = (float) (double) 3.45;
6     float y = -190.444f;
7     /* Notação científica */
8     printf("x = %e, y = %e \n", x, y);
9     /* Notação decimal */
10    printf("x = %f, y = %f \n", x, y);
11    /* Só as 3 primeiras decimais */
12    printf("x = %.5f, y = %.3f", x, y);
13    return 0;
14 }
```

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

DIM0321

14. Pontos flutuantes

2015.1 15 / 25

O tipo double

```
1 #include <stdio.h>
2 #include <float.h> /* DBL_MIN, DBL_MAX */
3
4 int main(void)
5 {
6     double m = 3.45 * -190.444;
7     double d = 3.45 / -190.444;
8     printf("m = %le, d = %le \n", m, d);
9     printf("m = %lf, d = %lf \n", m, d);
10    printf("m = %.31f, d = %.31f", m, d);
11    return 0;
12 }
```

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

DIM0321

14. Pontos flutuantes

2015.1 16 / 25

O tipo long double

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     long double m = 3.45 * -190.444;
6     long double d = 3.45 / -190.444;
7     printf("m = %Le, d = %Le \n", m, d);
8     printf("m = %Lf, d = %Lf \n", m, d);
9     printf("m = %.3Lf, d = %.3Lf", m, d);
10    return 0;
11 }
```

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

Representações especiais

```
1 #include <stdio.h>
2 #include <float.h> /* FLT_MIN, FLT_MAX */
3
4 int main(void)
5 {
6     float x = 3.45f;
7     float y = .0f;
8     float d1 = x / y;
9     float d2 = -x / y;
10    float d3 = y / y;
11    printf("d1 = %e, d2 = %e, d3 = %e\n", d1, d2, d3);
12
13    /* The following gives a division by zero error.
14     int c = 1 / 0;
15     printf("%d\n", c);
16     */
17    return 0;
18 }
```

d1 = inf d2 = -inf d3 = -nan

DIM0321

14. Pontos flutuantes

2015.1 17 / 25

DIM0321

14. Pontos flutuantes

2015.1 18 / 25

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

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     float x = 0.1;
6
7     printf("Value of x = %.9f, 4 * x = %.9f\n", x, 4 * x);
8     return 0;
9 }
```

Value of x = 0.100000001 4 * x = 0.400000006

DIM0321

14. Pontos flutuantes

2015.1 19 / 25

DIM0321

14. Pontos flutuantes

2015.1 20 / 25

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 explícita

- É melhor explicitar conversões de tipos (casts) com () antes da expressão.
- Operações de conversão têm uma precedência maior do que os operadores binários.
- Assim $(\text{float})\ x / 1 \equiv ((\text{float})\ x) / 1$

```
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 }
```

DIM0321

14. Pontos flutuantes

2015.1 21 / 25

Conversões

De inteiros

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

Para inteiros

- $\text{float} \rightarrow \text{int}$: 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 */
```

DIM0321

14. Pontos flutuantes

2015.1 23 / 25

DIM0321

14. Pontos flutuantes

2015.1 24 / 25

Perguntas ?



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