

Tipos Básicos de Dados

TIPOS PONTOS FLUTUANTES

Introdução

- Computadores trabalham com diversos tipos de dados:
 - **Texto** (letras, números, pontuação, etc.)
 - **Números** (naturais, reais, complexos, etc.)
 - **Áudio** (wav, mp3, ogg, etc.)
 - **Imagem** (bmp, jpg, gif, png, tga, etc.)
 - **Vídeo** (avi, mpg, wmv, etc.)
- Todos estes dados são representados pelo computador como um **conjunto de bits**

Tipos de Dados

- Se distinguem pela natureza dos valores armazenados:
 - **Tipo Inteiro**: números inteiros positivos e negativos.
Ex.: 30; -20; 0; -1; 390065
 - **Tipo Caractere**: letras, símbolos, números, pontuação.
Ex.: a, x, k, {, }, !, \$, 3, #
 - **Tipo Ponto Flutuante**: números reais positivos e negativos.
Ex.: 1.25; -30.54; 0.003; 2×10^{-8}
 - **Tipo Booleano**: verdadeiro ou falso.
Ex.: true, false, 0, 1

Introdução

- Os **pontos flutuantes** são outro grande grupo de tipos
 - Suportados nativamente pelas CPUs

11 tipos

Inteiros

bool
char
unsigned char
short
unsigned short
int
unsigned int
long
unsigned long
long long
unsigned long long

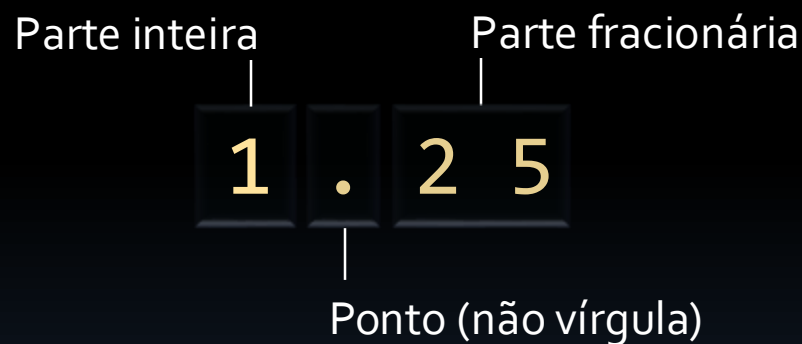
3 tipos

Pontos
Flutuantes

float
double
long double

Introdução

- Números em ponto flutuante tem uma **parte fracionária**



- Representam os **números reais**
- Utiliza-se **ponto** para indicar a parte fracionária
 - Ex.: 2.5, 3.14159, 0.442

Introdução

- Tipos **pontos flutuantes** fornecem uma **faixa maior de valores**
 - Alguns números muito grandes para os tipos inteiros podem ser armazenados em pontos flutuantes

Ex.: 872838789825284000000, 3×10^{26} , 8×10^{300}

| Tipos Inteiros | Bits | Faixa |
|----------------|------|--|
| short | 16 | -32.768 a 32.767 |
| int | 32 | -2.147.483.648 a 2.147.483.647 |
| long | 32 | -2.147.483.648 a 2.147.483.647 |
| long long | 64 | -9.223.372.036.854.775.808 a 9.223.372.036.854.775.807 |

Introdução

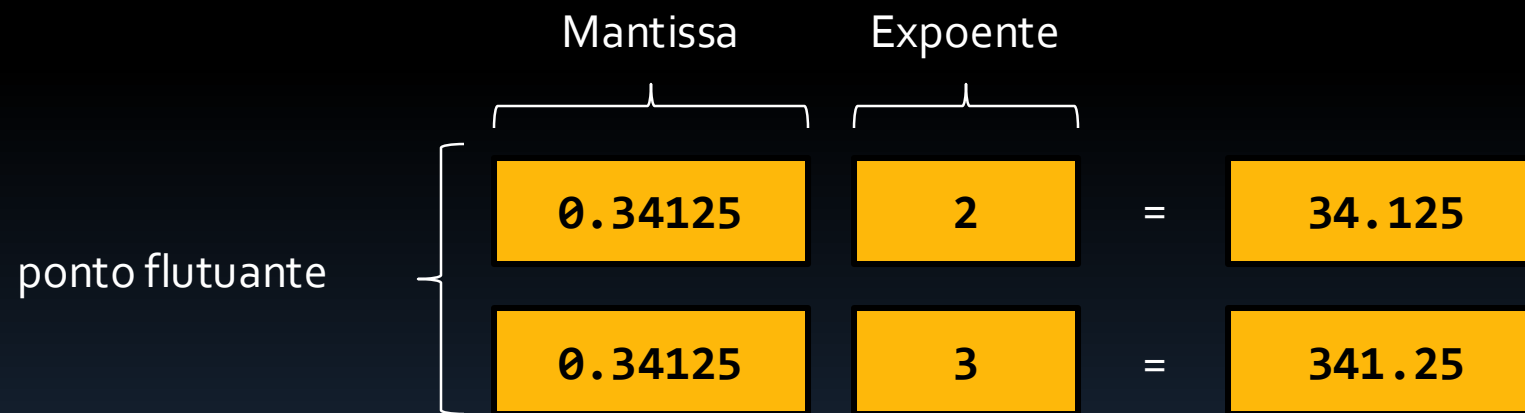
- O computador **armazena um número real em duas partes**
 - Um valor (**mantissa**)
 - Um fator de escala (**expoente**)



Os números 34.125 e 341.25 são idênticos,
a não ser pela posição da sua vírgula

Introdução

- O **fator de escala** não é armazenado na memória como um multiplicador, mas sim como um **expoente**



$$0.34125 * 10^2 = 34.125$$

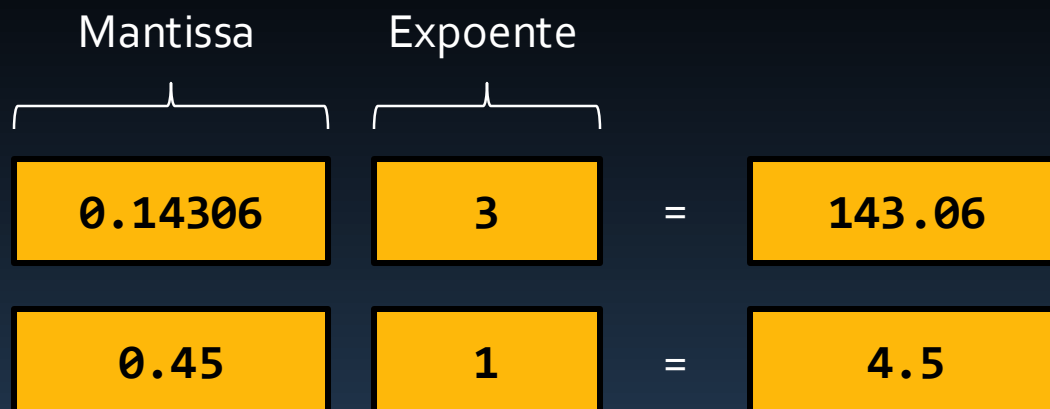
$$0.34125 * 10^3 = 341.25$$

Tipos Pontos Flutuantes

- A linguagem C++ tem três tipos pontos flutuantes:
 - `float`
 - `double`
 - `long double`
- A principal diferença entre os tipos está no:
 - Número de **dígitos significativos** para a mantissa
 - **Valor máximo** para o expoente



Tipos Pontos Flutuantes

- O que são **dígitos significativos** ?
 - **143.06** tem 5 dígitos significativos
 - **4.50** tem 2 dígitos significativos



Tipos Pontos Flutuantes

- Qual o valor do **expoente**?

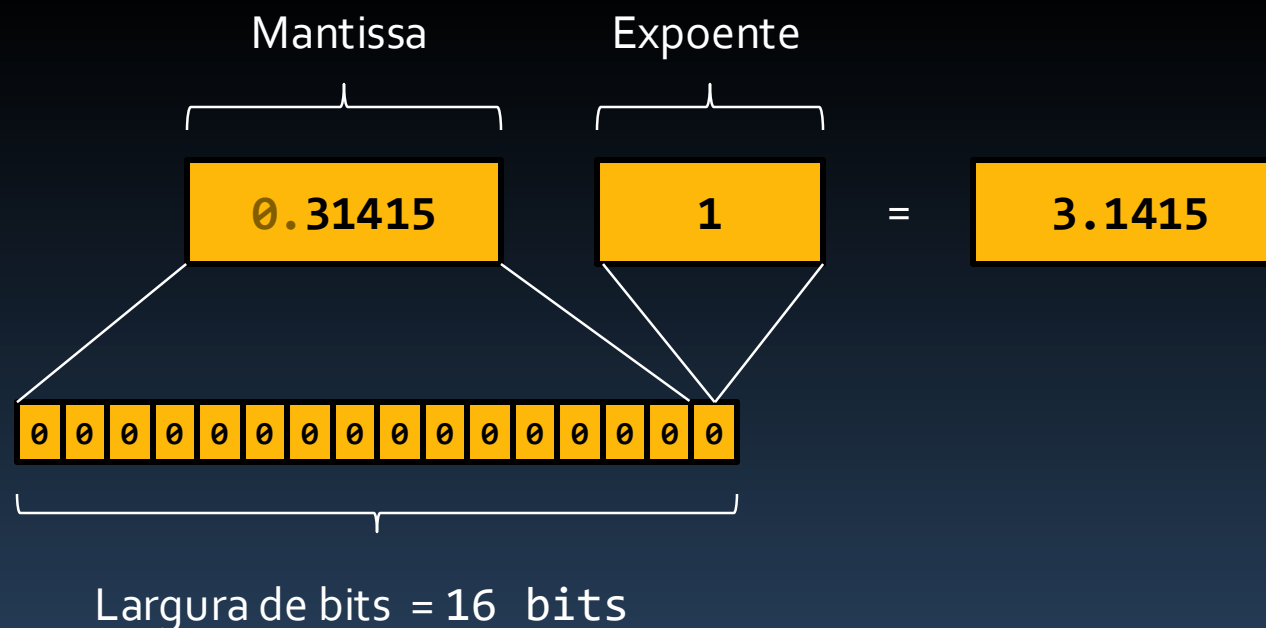
| | | Mantissa | Expoente |
|---------|---|--|---|
| | |  |  |
| 1785.16 | = | 0.178516 | 4 |
| 0.02145 | = | 0.2145 | -1 |

Apenas a parte depois do ponto é armazenada na mantissa

Ele é obtido depois de **isolar os dígitos significativos** da mantissa

Tipos Pontos Flutuantes

- Quanto maior a **largura de bits** usada para:
 - **Mantissa**: maior a precisão do número
 - **Expoente**: maior a magnitude do número



Tipos Pontos Flutuantes

- Em C++, os pontos flutuantes têm as seguintes **larguras**:
 - **float** tem pelo menos 32 bits
 - **double** tem pelo menos 48 bits
(e é pelo menos tão grande quanto float)
 - **long double** tem pelo menos o mesmo número de bits do double



Tipos Pontos Flutuantes

- Na prática, normalmente os valores são:
 - `float` tem 32 bits
 - `double` tem 64 bits
 - `long double` tem 64, 80, 96 ou 128 bits *
- Os valores específicos para uma plataforma podem ser encontrados no arquivo de cabeçalho `cfloat`

```
#include <cfloat>
```

Tipos Pontos Flutuantes

```
#include <iostream>
#include <float>
using namespace std;
int main()
{
    cout << "Número de Dígitos Significativos" << endl;
    cout << "float:      " << FLT_DIG << endl;
    cout << "double:     " << DBL_DIG << endl;
    cout << "long double: " << LDBL_DIG << endl;

    cout << "Valores Máximos do Expoente" << endl;
    cout << "float:      " << FLT_MAX_10_EXP << endl;
    cout << "double:     " << DBL_MAX_10_EXP << endl;
    cout << "long double: " << LDBL_MAX_10_EXP << endl;

    cout << "Número de Bits na Mantissa" << endl;
    cout << "float:      " << FLT_MANT_DIG << endl;
    cout << "double:     " << DBL_MANT_DIG << endl;
    cout << "long double: " << LDBL_MANT_DIG << endl;
}
```

Tipos Pontos Flutuantes

- A saída do programa (Linux/g++):

Número de Dígitos Significativos

float : 6

double : 15

long double: 18

Valores Máximos do Expoente

float : 38

double : 308

long double: 4932

Número de Bits na Mantissa

float : 24

double : 53

long double: 64

Tipos Pontos Flutuantes

- A saída do programa (Windows/ MSVC):

Número de Dígitos Significativos

float : 6

double : 15

long double: 15

Valores Máximos do Expoente

float : 38

double : 308

long double: 308

Número de Bits na Mantissa

float : 24

double : 53

long double: 53

Tipos Pontos Flutuantes



- Essas informações ajudam a **decidir qual tipo usar**
 - **Ex.:** Calcular a área de um círculo ($A = \pi r^2$)
 - Em que tipo guardar o valor de π ?

| | | | Mantissa | Expoente |
|--------|------------|---|-------------|----------|
| | | | | |
| float | 3.14 | = | 0.314 | 1 |
| float | 3.14159 | = | 0.314159 | 1 |
| double | 3.14159265 | = | 0.314159265 | 1 |

float é suficiente para até 6 **dígitos significativos** e **expoente 38**

Tipos Pontos Flutuantes

- E **que tipo usar** para os valores abaixo?
 - A massa da terra = 5.9722×10^{24} Kg
 - O número de átomos no universo = 3×10^{79}

| | | | Mantissa | Expoente |
|--------|-------------------------|---|---|---|
| | | |  |  |
| float | 5.9722×10^{24} | = | 0.59722 | 25 |
| double | 3×10^{79} | = | 0.3 | 80 |

double é suficiente para até 15 **dígitos significativos** e **expoente** 308

Tipos Pontos Flutuantes

- Os tipos **int** e **float** ambos têm **32 bits**, mas o computador representa-os de forma diferente na memória

int (32 bits) = 1042284544

0 0 1 1 1 1 1 0 0 0 1 0

float (32 bits) = 0.15625

0 1 1 1 1 1 0 0 0 0 1 0

8 bits
(expoente)

24 bits
(mantissa)

Representação do Ponto Flutuante

- A conversão de **binário para ponto flutuante** é dada por:

$$v = s * 2^{(e-127)} * (1 + m)$$

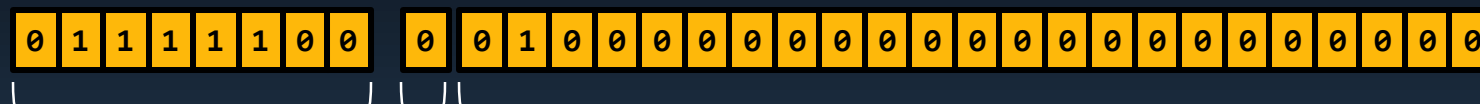
$$v = (+1) * 2^{(124-127)} * (1 + 0.25)$$

$$v = 2^{-3} * 1.25 = 0.125 * 1.25 = 0.15625$$

Os bits da mantissa representam um somatório de potências inversas de 2 decrescentes

$$\left(\frac{1}{2^1} + \frac{1}{2^2} + \frac{1}{2^3} + \frac{1}{2^4} + \frac{1}{2^5} + \frac{1}{2^6} + \frac{1}{2^7} + \frac{1}{2^8} + \frac{1}{2^9} + \frac{1}{2^{10}} + \frac{1}{2^{11}} + \frac{1}{2^{12}} + \frac{1}{2^{13}} + \frac{1}{2^{14}} + \dots + \frac{1}{2^{23}}\right)$$

float (32 bits) = 0.15625



8 bits
(expoente)

1 bit
(sinal)

23 bits
(mantissa)

Representação do Ponto Flutuante

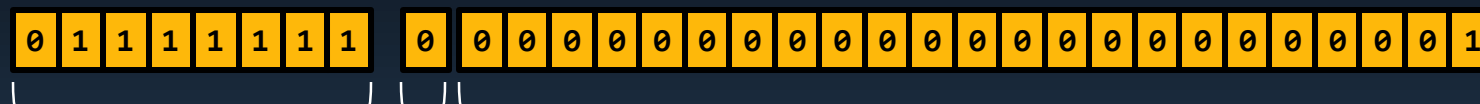
- Nem todo número real possui uma **representação exata**
 - O somatório não consegue gerar **todos os números possíveis**

$$\left(\frac{1}{2^1} + \frac{1}{2^2} + \frac{1}{2^3} + \frac{1}{2^4} + \frac{1}{2^5} + \frac{1}{2^6} + \frac{1}{2^7} + \frac{1}{2^8} + \frac{1}{2^9} + \frac{1}{2^{10}} + \frac{1}{2^{11}} + \frac{1}{2^{12}} + \frac{1}{2^{13}} + \frac{1}{2^{14}} + \dots + \frac{1}{2^{23}}\right)$$

$$1/2^{23} = 0.00000011920928955078125 = \text{FLT_EPSILON}$$

$$1/2^{22} = 0.0000002384185791015625$$

float (32 bits) = 0.00000011920928955078125



8 bits
(expoente)

1 bit
(sinal)

23 bits
(mantissa)

Representação do Ponto Flutuante

```
#include <iostream>
using namespace std;

int main()
{
    float f = 6.1;
    cout << "f = " << f << endl;

    // exibe número com 8 casas depois de vírgula
    cout.setf(ios_base::fixed, ios_base::floatfield);
    cout.precision(8);

    cout << "f = " << f << endl;

    return 0;
}
```

Representação do Ponto Flutuante

- A saída do programa:

f = 6.1

f = 6.09999990

- Nem todo **número real** possui uma **representação exata** em formato ponto flutuante
- A **exibição padrão sempre arredonda** o valor ponto flutuante, eliminando os zeros à direita

Constante Ponto Flutuante

- Uma **constante** ponto flutuante **pode ser representada:**

- Usando a **notação decimal padrão**

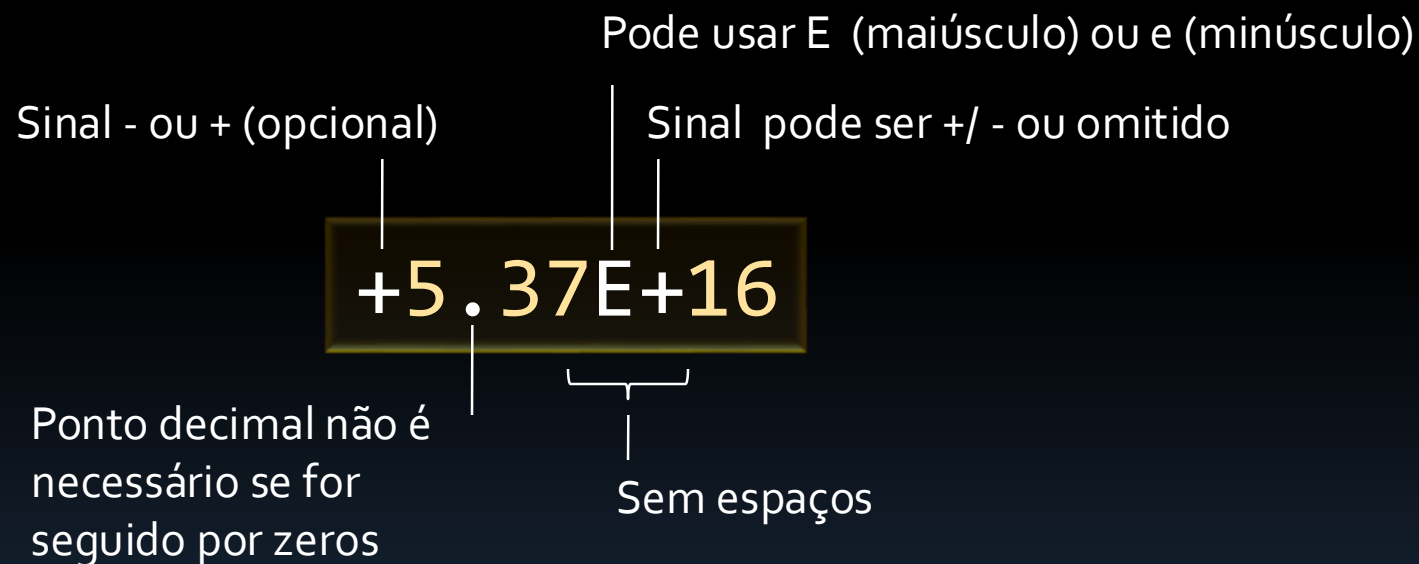
```
12.34      // ponto flutuante
939001.32  // ponto flutuante
0.00023    // ponto flutuante
8.0        // também ponto flutuante
```

- Usando a **notação exponencial**

```
2.52e+8    // pode usar e ou E, + é opcional
8.33E-4    // expoente pode ser negativo
7E5        // o mesmo que 7.0E+05
-18.32e13  // pode ter + ou - na frente
```

Constante Ponto Flutuante

- A notação E:



mE+n significa mova o ponto decimal **n casas para a direita**

mE-n significa mova o ponto decimal **n casas para a esquerda**

Constante Ponto Flutuante

- A notação E
 - **Garante** que o número será um **ponto flutuante**

```
70.0      // valor ponto flutuante
70         // valor inteiro
70E        // ponto flutuante igual a 70.0
```

- O **sinal da frente** se aplica a mantissa
- O **sinal no expoente** se aplica ao fator de escala

```
-8.33E4    // é igual a -83300
 8.33E-4    // é igual a  0.000833
-8.33E-4    // é igual a -0.000833
```

Constante Ponto Flutuante

- Por padrão as **constantes ponto flutuante** são armazenadas em um **double** (8 bytes)

```
float a = 7.0;           // constante double e variável float
```

- É possível indicar o **tipo das constantes**:

```
float flt = 7.0f;        // constante float e variável float  
double dbl = 7.0;        // constante double e variável double  
long double ldb = 7.0l;  // constante long double e variável long double
```

Precisão do Ponto Flutuante

```
#include <iostream>
using namespace std;

int main()
{
    // exibe números sempre com 6 casas depois da vírgula
    cout.setf(ios_base::fixed, ios_base::floatfield);

    float  fltvar = 10.0 / 3.0;      // bom para até 6 dígitos
    double dblvar = 10.0 / 3.0;      // bom para até 15 dígitos
    float  milhao = 1.0e6;

    cout << "float var = " << fltvar;
    cout << ", vezes um milhão = " << milhao * fltvar << endl;
    cout << "double var = " << dblvar;
    cout << ", vezes um milhão = " << milhao * dblvar << endl;
}
```

Precisão do Ponto Flutuante

- A saída do programa:

```
float var  = 3.333333, vezes um milhão = 3333333.250000  
double var = 3.333333, vezes um milhão = 3333333.333333
```

- A função **setf()** fixa o número de casas depois da vírgula
 - A variável **float** garante precisão para 6 dígitos
 - A variável **double** garante precisão para 15 dígitos

Precisão do Ponto Flutuante

```
#include <iostream>
using namespace std;

int main()
{
    float a = 2.34E+8;
    float b = a + 1.0f;

    cout << "a = " << a << endl;
    cout << "b - a = " << b - a << endl;

    return 0;
}
```

Precisão do Ponto Flutuante

- A saída do programa:

$a = 2.34e+8$

$b - a = 0$

- O valor armazenado em b possui mais de 6 dígitos significativos e é **perdido no armazenamento**
- É preciso sempre observar a precisão das constantes e dos **resultados das operações matemáticas**

Comparação com Inteiros

- **Vantagens:**

- Eles podem **representar valores entre dois números inteiros**
- Devido ao fator de escala, podem **representar uma faixa de valores maior** que os tipos inteiros

- **Desvantagens:**

- **Operações** com pontos flutuantes são **mais lentas** (que operações com inteiros)*
- Pontos flutuantes **perdem precisão**

Resumo

- Os tipos pontos flutuantes são:
 - `float`
 - `double`
 - `long double`
- Valores pontos flutuantes são armazenados em
 - Uma `mantissa`
 - Um `expoente`
- O arquivo `cfloat` deve ser consultado para saber os tamanhos de cada tipo em uma plataforma específica