**UNIVERSIDADE FEDERAL DE CATALÃO – UFCAT**

**CIÊNCIAS DA COMPUTAÇÃO**

RAFAEL BERNARDES DE JESUS

RAFAEL DA SILVA MELO

**TRABALHO 1 – PONTO FLUTUANTE**

CATALÃO

2022

RAFAEL BERNARDES DE JESUS

RAFAEL DA SILVA MELO

**Trabalho 1 – Ponto flutuante**

Trabalho avaliativo apresentado à Universidade Federal de Catalão a fim de obtenção de nota na componente Introdução à Computação no curso de Ciências da Computação.

Professor: Tércio Alberto dos Santos Filho.

CATALÃO

2022

**RESUMO**

O presente trabalho pretende demonstrar um método simples de conversão de números binários no padrão IEEE-754 para ponto flutuante e vice-versa, destacando um pouco do uso do padrão citado e sua origem. Foram desenvolvidos programas em linguagem C para a obtenção de resultados. O uso das ferramentas proporcionadas pela linguagem C torna a conversão prática e mostrou resultados bem precisos e plausíveis.

**Palavras-chave:** IEEE-754.Binário. Conversão. Ponto flutuante. Computação. Linguagem C.

**ABSTRACT**

The present academic work pretends to show a simple method of IEEE-754 binary numbers conversion to float numbers and vice-versa, contrasting the usage and the origin of the quoted standard. C Language programs were developed to obtain the results. The use of C Language provided tools made the conversion praticle and showed very accurate and likely results.

**Keywords:** IEEE-754.Binary. Conversion. Float. Computing. C programming language.

SUMÁRIO

[1. INTRODUÇÃO 6](#_Toc113478887)

[1.1 Ponto Flutuante 6](#_Toc113478888)

[1.2 O padrão IEEE-754 6](#_Toc113478889)

[2. JUSTIFICATIVA 8](#_Toc113478890)

[3. PROGRAMAS CONVERSORES 8](#_Toc113478891)

[3.1 Binário IEEE-754 para decimal 8](#_Toc113478892)

[3.2 Decimal para Binário IEEE-754 11](#_Toc113478893)

[4. ANÁLISE DE DADOS E DEMONSTRAÇÃO DE RESULTADOS 13](#_Toc113478894)

[4.1 Programa “Trabalho.1\_754-Bi.c” 13](#_Toc113478895)

[4.2 Programa “Decimal\_para\_padrao\_Ieee\_754\_PRONTO.c” 14](#_Toc113478896)

[5. CONCLUSÕES 15](#_Toc113478897)

# INTRODUÇÃO

Para o entendimento completo do presente trabalho, é necessário situar os padrões e as ferramentas utilizadas durante seu desenvolvimento.

# Ponto Flutuante

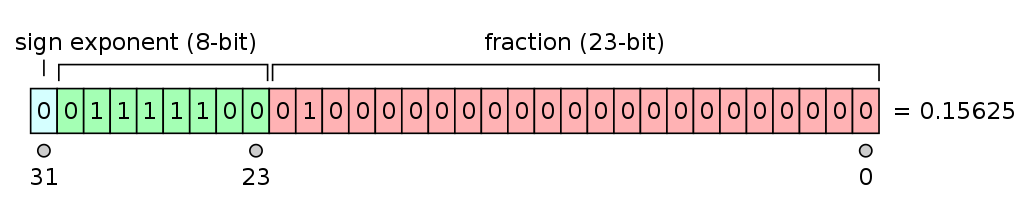
Ponto Flutuante é basicamente um formato de representação de números reais em um sistema digital. Linguagens de programação, como a usada nesse trabalho, fazem referência a esse formato (*float*) em sua sintaxe, denotando assim seu uso.

# 1.2 O padrão IEEE-754

IEEE-754 é um formato de aritmética de ponto flutuante estabelecido em 1985 pelo Instituto de Engenheiros Eletricistas e Eletrônicos ou, em inglês, *Institute of Electrical and Electronics Engineers* (IEEE). O padrão IEEE-754 descreve formatos de ponto flutuante e é usado em várias unidades de hardware que funcionam com tal descrição.

Existem duas precisões principais para a representação binária: Simples (32 bits) e Dupla (64 bits). Este trabalho se baseará apenas na abordagem de precisão simples.

O padrão é composto por uma sequência de 32 bits, esses são divididos em três partes: O Sinal (1 bit), o Expoente (8 bits) e a Mantissa (23 bits), como mostrado no exemplo a seguir:

**Figura 1 –** Representação ieee-754

Fonte: Wikipédia[[1]](#footnote-1)

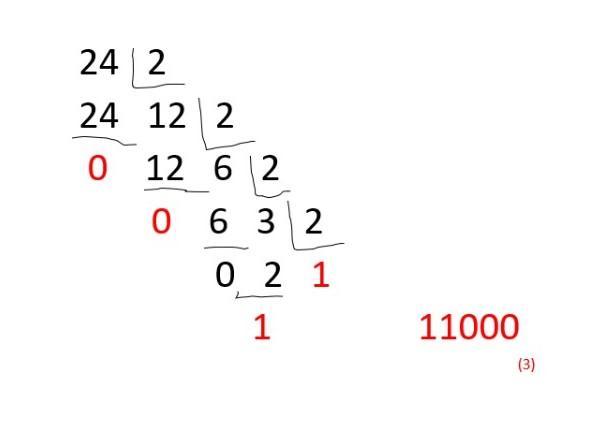
O primeiro bit, como seu próprio nome já mostra, representa o sinal do número. Em caso de 0, é positivo, em caso de 1 é negativo.

Os oito bits subsequentes são o expoente, no entanto é mais didático explicar a mantissa primeiro.

A mantissa são os 23 bits após o expoente. Eles representam a parte fracionária de um número binário o qual se encontrava em notação científica.

Voltando ao expoente, ele representa o expoente de 2, após a transformação em notação citada anteriormente, somado a 127.

Para clareamento da ideia, será usado o exemplo do número 24,6:

**Figura 2** – 24 em binário

Fonte: Brainly[[2]](#footnote-2)

2410 = 110002

0,6 x 2 = **1**,2

0,2 x 2 = **0**,4

0,4 x 2 = **0**,8

0,8 x 2 = **1**,6 0,610 = 100110011...2

0,6 x 2 = **1**,2

0,2 x 2 = **0**,4

0,4 x 2 = **0**,8

0,8 x 2 = **1**,6

Como mostrado, 24 em base decimal equivale a 11000 em base binária, e 100110011... tem valor 0,6. Logo, 24,6 equivale a 11000,100110011... e para a transformação no padrão ieee-754, seria necessário deslocar a vírgula até o primeiro número e compensar com uma multiplicação por 2n (n sendo quantas casas ela se deslocou), para não alterar o valor numérico. Nessa notação, 24,6 é equivalente a 1,10001001100110011001100 x 2**4**. Agora em notação, os 23 algarismos após a vírgula seriam a mantissa e o expoente de 2 (quantas vezes a vírgula foi deslocada para a esquerda) é o número somado a 127 para a descoberta do expoente, que nesse caso seria 131 (127 + 4, e 10000011 em binário).

Contudo, 24,6 no padrão IEEE-754 é representado por:

0 | 10000011 | 10001001100110011001100

# JUSTIFICATIVA

As duas formas principais de representação de números reais em binário são o ponto fixo (*fixed point*) e o ponto flutuante (*floating point*). Resumidamente, pontos fixos são representados fixando o tamanho da parte inteira e fracionária de um número binário a ser armazenado, sendo os valores à esquerda desse ponto são maiores que 1 (2, 4, 8 ...) e os à direita, menores que 1 (½, ¼, ⅛ ...).

Já pontos flutuantes utilizam o padrão binário da notação científica, de modo a possuir as três partes já mencionadas anteriormente nesse artigo: Sinal, Mantissa/ Número base (determinante da acurácia do número a ser armazenado) e Expoente (O intervalo do número a ser armazenado).

A utilização desse formato possibilita uma precisão maior e melhor garantia de resultados coerentes.

# PROGRAMAS CONVERSORES

Para a demonstração prática da conversão, foram desenvolvidos dois programas em linguagem de programação C, ambos utilizando a IDE gratuita Code Blocks. Os programas terão o funcionamento explicado a seguir.

# Binário IEEE-754 para decimal

O código organizado ocupou 49 linhas (por mais que na linguagem utilizada isso seja irrelevante para funcionamento).

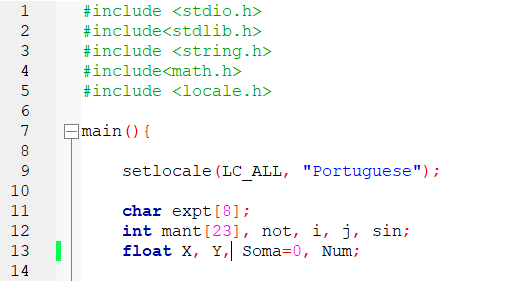
Para a obtenção de um resultado aproximado, os dados recolhidos serão tratados com base na seguinte fórmula:

(-1)sinal \* (1 + mantissa) \* 2expoente

A fórmula determina o sinal elevando -1 a 0 ou 1. Caso 0, (-1)0 = 1, tornando o resultado final positivo, e em caso de 1, (-1)1 = -1, tornando o resultado final negativo. A mantissa deve ser transformada em um número decimal como se todos seus componentes estivessem após a vírgula, ou seja, multiplicar o primeiro número por 2-1, o segundo por 2-2 e assim progressivamente, o resultado será um número decimal menor que 1. Deve se adicionar um a esse número e, após isso, multiplicá-lo por 2 elevado ao expoente descoberto com a subtração de 127 no número que representa o conjunto expoente no IEEE-754. Este cálculo resultará o número desejado.

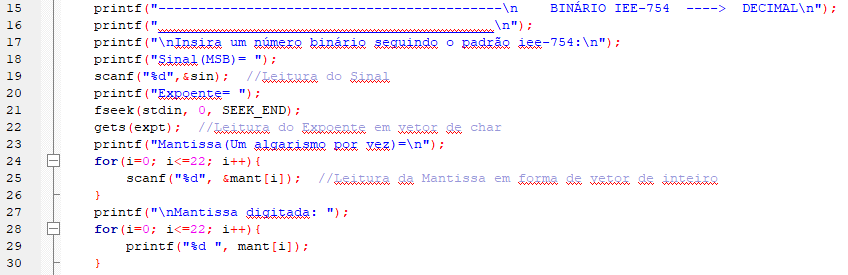
A primeira parte (1-13) apresenta apenas a inclusão de bibliotecas e a declaração de todas as variáveis usadas. As informações inseridas pelo usuário do programa serão armazenadas de formas distintas, o sinal será armazenado como inteiro, enquanto a mantissa e o expoente serão armazenados como vetores de inteiro e de character (respectivamente). Outras variáveis também já são declaradas para o uso de fórmulas no processo de transformação.

**Figura 3 –** Declarações



Fonte: Código do autor[[3]](#footnote-3)

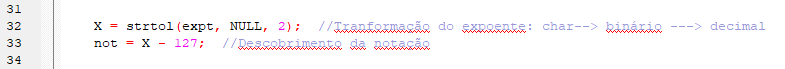
Logo após, a parte subsequente (15-26), trata da exposição do programa e da leitura dos dados. Primeiro, é mostrado uma espécie de “banner” do programa, em seguida, como devem ser inseridas a informações. Deve ser digitado o sinal, confirmar com Enter, e logo após inserir o expoente de uma vez só, esse será armazenado em um vetor de “char”. Em seguida, a mantissa deve ser digitada item por item, a fim de organizá-la em um vetor para que seja possível realizar as operações.

**Figura 4 –** Leitura dos dados

Fonte: Código do autor[[4]](#footnote-4)

Em sequência, é utilizada a função *strtol*, da biblioteca <stdlib.h>. Essa executa no vetor selecionado (expt, onde está armazenado o expoente), a mudança de vetor de char para um número float em decimal. Como será armazenado um número binário no vetor, essa função lerá item por item desse vetor e logo após o transformará diretamente em um número de base decimal.

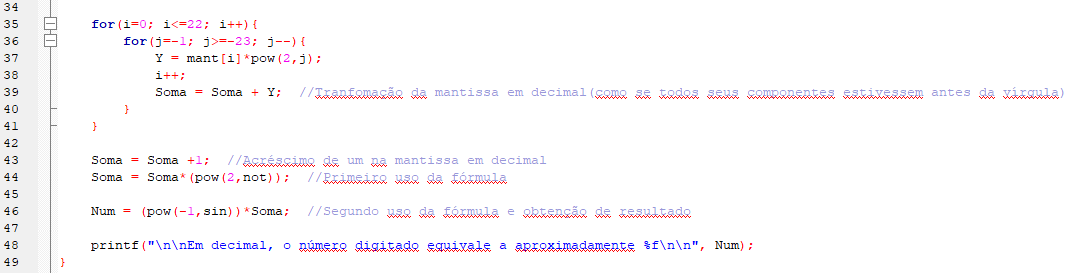
Assim, descobrindo o número que equivale ao expoente, será subtraído 127 do mesmo para o descobrimento do expoente de 2 na forma de notação.

**Figura 5 –** Cálculos part.1 e função strtol

Fonte: Código do autor[[5]](#footnote-5)

Para o cálculo da forma decimal alterada da mantissa, é utilizada a ferramenta *for*, sendo feita uma multiplicação progressiva de elemento por elemento da mantissa (armazenada em um vetor de inteiro) com o uso da função *pow* (<math.h>) e armazenando em uma variável “Soma” a soma da multiplicação de cada item.

Com o descobrimento desse número, é adicionado 1. Esse resultado é multiplicado por 2 elevado ao expoente achado anteriormente e armazenado na variável “not”. O resultado final após esses cálculos é a representação decimal aproximada do número digitado em IEEE-754.

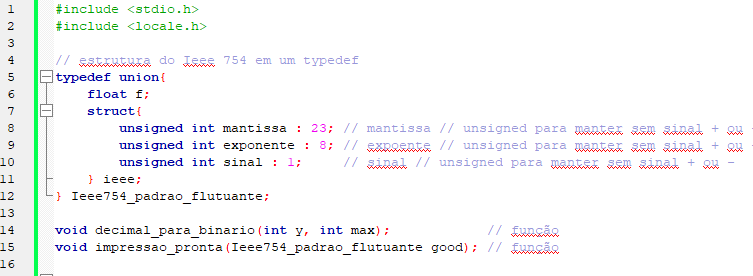
**Figura 6 –** Cálculos part.2 e demonstração de resultado

Fonte: Código do autor[[6]](#footnote-6)

# Decimal para Binário IEEE-754

O código apresentado converte número decimal com ponto flutuante para binário no padrão IEEE 754. Apenas bibliotecas stdio.h e locale.h foram necessárias.

Começando por uma estrutura de typedef que foi declarada 3 variáveis dentro da estrutura de acordo com o padrão requisitado: “f” (float), “mantissa” (inteiro -> com máximo de 23 números), “expoente” (inteiro -> com máximo de 8 números) e “sinal” (inteiro -> com máximo de 1 número). Tornando assim a estrutura do “Ieee754\_padrao\_flutuante”.

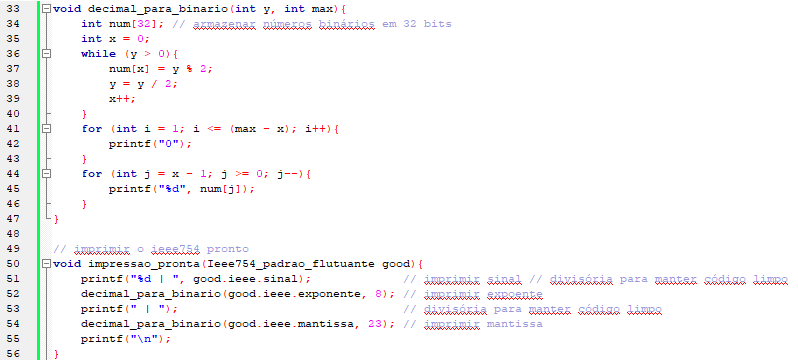
**Figura 7** – Typedef union

Fonte: código do autor[[7]](#footnote-7)

Logo após isso foram declaradas as funções:

“decimal\_para\_binario”: a qual tem o objetivo de realizar a conversão da parte inteira e fracionária para binário respeitando todos quesitos que a função solicita: “y” e “max”. Neste caso, “y” seria uma das três variáveis da estrutura definida no começo e, consequentemente, max seria o máximo de números que entraria no vetor;

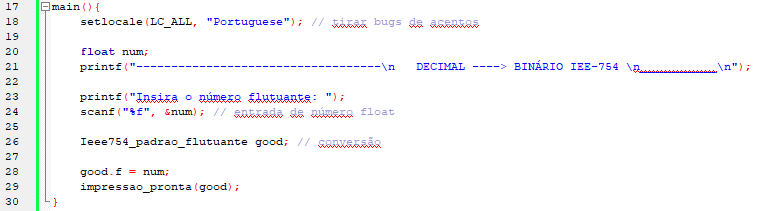
“impressao\_pronta”: com intuito de unir os procedimentos supracitados e realizar a formatação adequando no formato IEEE 754 de ponto flutuante.

**Figura 8 -** Funções

Fonte: Código do autor[[8]](#footnote-8)

No bloco main() é aplicado o setlocale para evitar bugs no console do Windows, e então um “banner” assim como no código anterior. Assim, é declarado uma variável “num” (float), um texto é exibido solicitando preenchimento do número decimal flutuante que deverá ser passado pela formatação e logo após guardando-o em ”num”. Para tornar o código mais objetivo, é realizada a transformação da estrutura “Ieee754\_padrao\_flutuante” em apenas uma palavra “good”. Em seguida, é transforamada a variável “f” declarada no typedef na variável local do main “num” e então a função “impressao\_pronta” é chamada para finalizar o código realizando suas derivadas alterações.

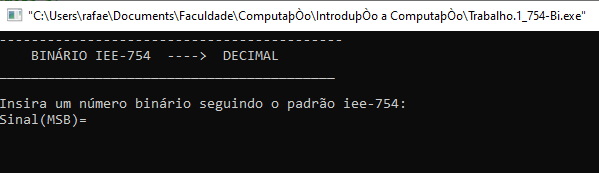
**Figura 9** – Bloco Main

Fonte: Código do autor[[9]](#footnote-9)

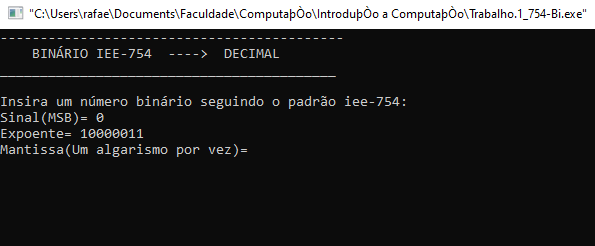
# ANÁLISE DE DADOS E DEMONSTRAÇÃO DE RESULTADOS

Para demonstrar o resultado dos programas desenvolvidos usaremos o exemplo citado ainda no item 1 deste artigo: 24,6 ou 0 | 10000011 | 10001001100110011001100.

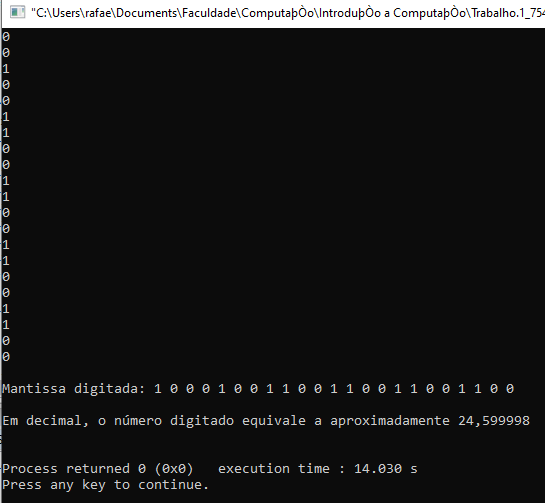
# Programa “Trabalho.1\_754-Bi.c”

**Figura 10 –** Abertura do programa

Fonte: Execução do código do autor[[10]](#footnote-10)

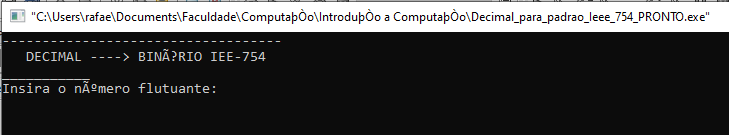
**Figura 11** – Após inserção de Sinal e Expoente

Fonte: Execução do código do autor[[11]](#footnote-11)

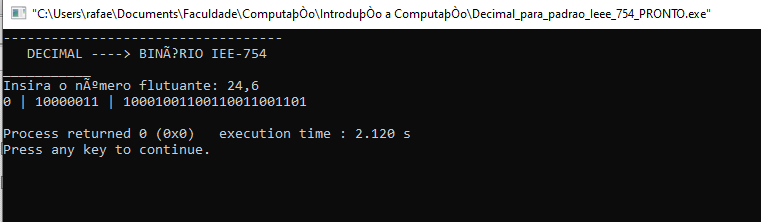
**Figura 12** – Após inserção de todos elementos da mantissa

Fonte: Execução do código do autor[[12]](#footnote-12)

# Programa “Decimal\_para\_padrao\_Ieee\_754\_PRONTO.c”

**Figura 13 –** Abertura do programa

Fonte: Execução do código do autor[[13]](#footnote-13)

**Figura 13 –** Após inserção do número

Fonte: Execução do código do autor[[14]](#footnote-14)

# CONCLUSÕES

Muitos hardwares do mercado atual utilizam do padrão binário IEEE-754 e sua compreensão é indispensável para qualquer membro da comunidade da computação. A transformação de um número em tal padrão (ou vice-versa), pode ser complicada por conta do tamanho que o mesmo apresenta, mas os resultados mostraram que os programas desenvolvidos têm uma acurácia boa e podem realizar o trabalho árduo de conversão com rapidez e qualidade.

**REFERÊNCIAS**

ALMEIDA, Mateus. **Convertendo IEEE 754 32 bits para decimal e vice-versa**. ead.urcamp.edu.br, 2020. Disponível em: <https://imsouza.github.io/convertendo-ieee32bits-para-decimal-e-vice-versa>. Acesso em: 07 set. 2022.

VALDISIO, Gerardo. **Padrão IEEE 754 para Aritmética Binária de Ponto Flutuante.** lia.ufc.br. Disponível em: <https://www.lia.ufc.br/~valdisio/download/ieee.pdf>. Acesso em: 07 set. 2022.

ANDRADE, Hélio. **IEEE 754 – Números binários com ponto flutuante.** guiatech.net,2014. Disponível em: <https://guiatech.net/ieee-754-conversao/>. Acesso em: 07 set. 2022.

WIKIPEDIA. **IEEE 754.** en.wikipedia.org**,** 2022. Disponível em: https://en.wikipedia.org/wiki/IEEE\_754. Acesso em: 07 set. 2022.

FELIPE, Luiz. **Representações numéricas — fixa e flutuante.** pt.stackoverflow.com, 2021. Disponível em: https://pt.stackoverflow.com/questions/493487/o-que-%C3%A9-ieee-754. Acesso em: 07 set. 2022.

1. Disponível em: <https://pt.m.wikipedia.org/wiki/Ficheiro:IEEE\_754\_Single\_Floating\_Point\_Format.svg>. Acesso em: 07 set. 2022. [↑](#footnote-ref-1)
2. Disponível em: < https://brainly.lat/tarea/39912406>. Acesso em: 07 set. 2022. [↑](#footnote-ref-2)
3. Captura de tela do trecho 1-14 do código Trabalho.1\_754-Bi.c. [↑](#footnote-ref-3)
4. Captura de tela do trecho 15-30 do código Trabalho.1\_754-Bi.c. [↑](#footnote-ref-4)
5. Captura de tela do trecho 31-34 do código Trabalho.1\_754-Bi.c. [↑](#footnote-ref-5)
6. Captura de tela do trecho 34-49 do código Trabalho.1\_754-Bi.c. [↑](#footnote-ref-6)
7. Captura de tela do trecho 1-16 do código Decimal\_para\_padrao\_Ieee\_754\_PRONTO.c. [↑](#footnote-ref-7)
8. Captura de tela do trecho 33-56 do código Decimal\_para\_padrao\_Ieee\_754\_PRONTO.c. [↑](#footnote-ref-8)
9. Captura de tela do trecho 17-30 do código Decimal\_para\_padrao\_Ieee\_754\_PRONTO.c. [↑](#footnote-ref-9)
10. Captura de tela da execução do código Trabalho.1\_754-Bi.c. [↑](#footnote-ref-10)
11. Captura de tela da execução do código Trabalho.1\_754-Bi.c. [↑](#footnote-ref-11)
12. Captura de tela da execução do código Trabalho.1\_754-Bi.c. [↑](#footnote-ref-12)
13. Captura de tela da execução do código Decimal\_para\_padrao\_Ieee\_754\_PRONTO.c. [↑](#footnote-ref-13)
14. Captura de tela da execução do código Decimal\_para\_padrao\_Ieee\_754\_PRONTO.c. [↑](#footnote-ref-14)