# ANÁLISE NUMÉRICA I -
## Bases Numéricas

UAlg FCT
UNIVERSIDADE DO ALGARVE
FACULDADE DE CIÊNCIAS E TECNOLOGIA

**Curso:**
Licenciatura em Engenharia Informática

**Unidade Curricular:**
Análise Numérica I

**Realizado por:**
Daniel Maryna (64611)
Miguel Silva (80072)
Francisco Nunes (80061)
Brandon Mejia (79261)

03/2024

# CONTEÚDO

# INTRODUÇÃO

O conceito de sistemas numéricos é fundamental para a compreensão de diversos temas em matemática e computação. Cada sistema numérico é definido por uma base, que determina o conjunto de símbolos utilizados para representar valores. Os sistemas mais amplamente conhecidos incluem o binário (base 2), decimal (base 10) e hexadecimal (base 16), mas também é possível trabalhar com sistemas de numeração que utilizam bases maiores, como até 62, que inclui caracteres alfabéticos e numéricos.

Este relatório descreve o desenvolvimento de um programa capaz de converter números entre diferentes bases numéricas, variando entre a base 2 e a base 62. O objetivo do programa é facilitar a conversão e o entendimento da representação de números em bases menos comuns, como bases alfanuméricas. Através de uma interface simples, o utilizador pode inserir um número numa base qualquer dentro deste intervalo e obter a sua representação em outras bases.

A possibilidade de trabalhar com bases até 62 é especialmente relevante para aplicações que utilizam codificações compactas, como em URLs curtas, codificação de dados, e em sistemas de criptografia. Este relatório detalhará o funcionamento do programa, sua implementação e os resultados obtidos.

# FUNDAMENTAÇÃO TEÓRICA

Os sistemas de numeração são formas de representar números usando um conjunto de símbolos. Cada sistema é definido por uma base, que determina a quantidade de símbolos permitidos e a maneira como os números são calculados. No caso de sistemas de numeração com base entre 2 e 62, o conjunto de símbolos é alargado, incluindo tanto números como letras. Abaixo são apresentados alguns dos sistemas de numeração mais comuns, além de uma explicação sobre a representação para bases superiores.

## Sistema Binário (Base 2)

O sistema binário é um dos mais utilizados em computação, onde apenas dois símbolos são usados: **0** e **1**. Cada posição em um número binário representa uma potência de 2. A computação digital moderna depende deste sistema, pois os computadores utilizam bits (que podem ser 0 ou 1) para realizar operações lógicas e aritméticas.

Exemplo:

O número binário **1011** corresponde ao decimal **11**, pois:

$$1 \times 2^3 + 0 \times 2^2 + 1 \times 2^0 = 11$$

## Sistema Decimal (Base 10)

O sistema decimal é o mais amplamente utilizado na vida cotidiana. É composto por dez símbolos: **0** a **9**, e cada posição em um número decimal representa uma potência de 10. Este é o sistema que a maioria das pessoas aprende primeiro, sendo utilizado em cálculos do dia a dia.

Exemplo:

O número decimal **235** é calculado como:

$$2 \times 10^2 + 3 \times 10^1 + 5 \times 10^0 = 235$$

## Sistema Hexadecimal (Base 16)

O sistema hexadecimal utiliza 16 símbolos: **0-9** e **A-F**, onde A representa o valor decimal 10, B representa 11, e assim por diante até F, que representa 15. Este sistema é frequentemente utilizado em programação e em eletrônica para representar valores compactos de forma mais legível do que o binário.

Exemplo:

O número hexadecimal **1A3** corresponde ao decimal **419**, pois:

$$1 \times 16^2 + A(10) \times 16^1 + 3 \times 16^0 = 419$$

## Sistemas com Maiores Bases (Até base 62)

Quando se lida com bases superiores a 16, além dos números de 0 a 9, também são utilizados caracteres alfabéticos. Em sistemas de base 62, por exemplo, os símbolos utilizados incluem os números **0-9**, as letras maiúsculas **A-Z**, e as letras minúsculas **a-z**. Este sistema é útil em situações onde é necessário representar grandes números de forma compacta, como em URLs curtas, hashes e outros cenários de codificação.

Exemplo:

Na base 62, o número **Z1a** corresponderia a um valor decimal elevado, já que **Z** representa **61**, **1** representa **1**, e **a** representa **36**.

# DESCRIÇÃO DO PROGRAMA

O programa foi implementado em Python com o objetivo de converter números reais entre duas bases quaisquer, no intervalo entre 2 e 62. O código trata tanto da parte inteira quanto da parte fracionária de um número, e pode lidar com números positivos e negativos. A precisão é ajustada para até 8 casas decimais para os cálculos envolvendo partes fracionárias.

## Principais Funções do Programa

- **Função** valor_para_char(valor):
  - Esta função converte um valor numérico (entre 0 e 61) no símbolo correspondente. Para valores de 0 a 9, retorna o caractere numérico equivalente. Para valores de 10 a 35, retorna as letras maiúsculas de A a Z. Para valores de 36 a 61, retorna as letras minúsculas de a a z. Esta função é fundamental para garantir a representação correta dos números em bases superiores à decimal.

- **Função** char_para_valor(char):
  - Esta função realiza o processo inverso. Dado um caractere (número ou letra), converte-o no valor numérico correspondente. É capaz de identificar se o caractere é um dígito ou uma letra, convertendo corretamente para o valor equivalente, com base na sua posição na tabela ASCII.

- **Função** converter_inteiro(número, base_atual, nova_base):
  - Responsável por converter a parte inteira de um número da base atual para a nova base. Utiliza divisões sucessivas pela nova base e regista os restos para formar o número na nova base.

- **Função** converter_fracionário(número, base_atual, nova_base):
  - Realiza a conversão da parte fracionária de um número. Esta função aplica multiplicações sucessivas pela nova base e regista os dígitos resultantes. O processo é repetido até atingir a precisão desejada (até 8 casas decimais).

- **Função** converter_numero(número, base_atual, nova_base):
  - Função principal que coordena as outras funções para realizar a conversão completa (parte inteira e fracionária). Ela desmembra o número em sua parte inteira e fracionária, converte cada parte separadamente, e então junta os resultados.

## Fluxo de Execução

1. **Entrada do Utilizador**: O programa começa pedindo ao utilizador um número em qualquer base entre 2 e 62, a base atual do número, e a nova base para a qual o número deve ser convertido.

2. **Verificação e Validação**: O programa verifica se os números e bases fornecidos são válidos. Por exemplo, não são permitidos valores fora do intervalo de bases ou caracteres inválidos.

3. **Conversão**: Através de funções de conversão, o número é processado, separando a parte inteira e fracionária, realizando a mudança de base de forma individual para cada parte.

4. **Saída**: O programa devolve o número convertido na nova base, com a parte fracionária sendo limitada a 8 casas decimais.

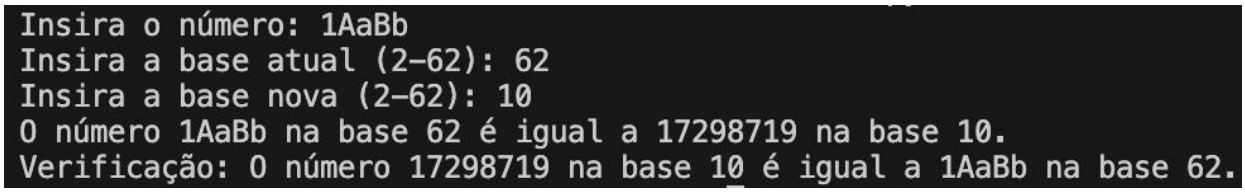


***Figura 1*** *- Exemplo de Execução do Programa*

## Precisão e Limitações

- **Limite de Precisão**: Para manter a precisão em números fracionários, o programa fixa o resultado em 8 casas decimais. Este é um ajuste prático para evitar loops infinitos ou erros de arredondamento que podem ocorrer durante a conversão de números com partes fracionárias em bases não decimais.

- **Limitação de Valores**: A base de destino deve estar entre 2 e 62, e o programa não suporta bases fora deste intervalo.

## Decisões de Implementação

O programa foi implementado de forma eficiente, minimizando a necessidade de arrays estáticos para mapear caracteres. Ao invés de armazenar os símbolos em arrays, utiliza-se a tabela ASCII diretamente, o que reduz a complexidade do código e torna o processo de conversão mais rápido.

# RESULTADOS

Nesta secção, apresentamos diversos exemplos de conversões realizadas pelo programa, entre diferentes bases, incluindo a verificação da exatidão dos resultados. Abaixo estão alguns exemplos típicos de entradas e saídas, acompanhados de uma breve explicação.

**Exemplo 1: Conversão de decimal (base 10) para base 11**

```
Insira o número: 50.3
Insira a base atual (2-62): 10
Insira a base nova (2-62): 11
O número 50.3 na base 10 é igual a 46.33333333 na base 11.
Verificação: O número 46.33333333 na base 11 é igual a 50.29999999 na base 10.
```

***Figura 2*** *- Captura do programa para o Exemplo 1*

- **Entrada:**
  Número: 50.3
  Base atual: 10
  Base nova: 11

- **Saída:**
  O número 50.3 na base 10 é igual a 46.33333333 na base 11.

- **Verificação:**
  O número 46.33333333 na base 11 é igual a 50.29999999 na base 10.

**Observação:** O processo de conversão fracionária está limitado a 8 casas decimais, como especificado, resultando em uma ligeira diferença devido ao arredondamento.

**Exemplo 2: Conversão de decimal (base 10) para base 62**

```
Insira o número: -13045810
Insira a base atual (2-62): 10
Insira a base nova (2-62): 62
O número -13045810 na base 10 é igual a -sjoI na base 62.
Verificação: O número -sjoI na base 62 é igual a -13045810 na base 10.
```

*Figura 3 - Captura do programa para o Exemplo 2*

- **Entrada:**
  Número: -13045810
  Base atual: 10
  Base nova: 62

- **Saída:**
  O número -13045810 na base 10 é igual a -sjol na base 62.

- **Verificação**:
  O número -sjol na base 62 é igual a -13045810 na base 10.

**Observação:** Neste caso, o programa lida corretamente com números negativos e realiza a conversão sem problemas.

**Exemplo 3: Conversão de base 62 para decimal (base 10)**

```
Insira o número: 1AaBb
Insira a base atual (2-62): 62
Insira a base nova (2-62): 10
O número 1AaBb na base 62 é igual a 17298719 na base 10.
Verificação: O número 17298719 na base 10 é igual a 1AaBb na base 62.
```

*Figura 4 - Captura do programa para o Exemplo 3*

- **Entrada:**
  Número: 1AaBb
  Base atual: 62
  Base nova: 10

- **Saída:**
  O número 1AaBb na base 62 é igual a 17298719 na base 10.

- **Verificação:**
  O número 17298719 na base 10 é igual a 1AaBb na base 62.

**Observação:** O programa também é capaz de converter números que contêm letras, como 1AaBb, em bases altas, como 62, demonstrando a flexibilidade de trabalhar com bases alfanuméricas.

**Exemplo 4: Conversão entre bases altas (base 62 para base 13)**

```
Insira o número: 123BVx
Insira a base atual (2-62): 62
Insira a base nova (2-62): 13
O número 123BVx na base 62 é igual a 1211087A7 na base 13.
Verificação: O número 1211087A7 na base 13 é igual a 123BVx na base 62.
```

*Figura 5 - Captura do programa para o Exemplo 4*

- **Entrada:**
  Número: 123BVx
  Base atual: 62
  Base nova: 13

- **Saída:**
  O número 123BVx na base 62 é igual a 1211087A7 na base 13.

- **Verificação:**
  O número 1211087A7 na base 13 é igual a 123BVx na base 62.

**Observação:** Este exemplo mostra a versatilidade do programa ao trabalhar com bases não usuais, como 13.

**Exemplo 5: Erro de caracteres inválidos**

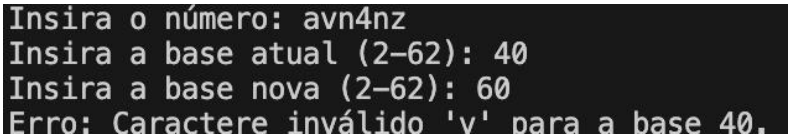


***Figura 6*** *- Captura do programa para o Exemplo 5*

- **Entrada:**
  Número: avn4nz
  Base atual: 40
  Base nova: 60

- **Saída**:
  Erro: Caractere inválido v para a base 40.

**Observação**: O programa valida corretamente os caracteres inseridos pelo utilizador e gera mensagens de erro adequadas quando os caracteres fornecidos não correspondem à base de entrada.

# CONCLUSÃO

O programa desenvolvido demonstrou ser capaz de realizar com sucesso a conversão de números reais entre bases numéricas variadas, dentro do intervalo de 2 a 62. Ao longo dos testes, o programa mostrou-se funcional tanto para números inteiros quanto para números fracionários, respeitando o limite de precisão de 8 casas decimais na parte fracionária. Além disso, o programa lida adequadamente com números negativos e verifica a validade dos caracteres de entrada, prevenindo a conversão de valores inválidos.

Uma das principais vantagens do programa é a sua flexibilidade para trabalhar com bases maiores, que incluem letras alfabéticas (maiúsculas e minúsculas), além dos números. Isto abre a possibilidade de utilizar o programa em aplicações mais específicas, como codificações compactas em bases alfanuméricas, muito utilizadas em áreas como criptografia e sistemas de identificação.

Durante a execução do programa, foram realizados vários testes que mostraram que a precisão das conversões é consistente, e o programa consegue realizar a verificação inversa, confirmando a correção dos cálculos realizados. No entanto, pequenas variações na parte fracionária foram observadas devido ao arredondamento inerente às conversões não exatas.

Entre os pontos fortes do programa, destacam-se:

- Capacidade de conversão de números reais (positivos e negativos) entre bases exóticas.
- Limitação automática da precisão fracionária, prevenindo loops infinitos ou longos.
- Tratamento adequado de caracteres inválidos, garantindo robustez nas entradas fornecidas pelos utilizadores.
- No entanto, existe a possibilidade de expandir o programa para suportar precisão mais elevada, quando necessário, ou otimizar o desempenho para lidar com números de grande magnitude em bases extremamente altas.

Em suma, o programa alcançou os objetivos propostos, mostrando-se eficiente e confiável na conversão de números entre várias bases, e poderá ser facilmente adaptado para outras aplicações que exijam manipulação de diferentes sistemas numéricos.

# REFERÊNCIAS

Aulas Teóricas do Professor Hermenegildo Augusto Vieira Borges De Oliveira;
RL Burden and JD Faires - Numerical Analysis - 9th edition 2011