Universidade do Algarve

Faculdade de Ciências e Tecnologia

Licenciatura em Engenharia Informática

Análise Numérica I

Trabalho 1 – Mudança de Base

Conversão de Números entre Bases (2 a 62)

Realizado por: João Andréz nº61066, Carlos Ferreira nº 71319, Diogo Freitas nº 90147, Diogo Carvalho nº 90247

Docente: Hermenegildo Borges de Oliveira

Ano Letivo: 2025/2026

Introdução

O presente trabalho foi desenvolvido no âmbito da unidade curricular Análise Numérica I e tem como objetivo principal a implementação de um programa capaz de realizar conversões numéricas entre diferentes sistemas de numeração, compreendidos entre a base 2 (binária) e a base 62.

A motivação para este trabalho prende-se com a importância da compreensão dos sistemas de numeração em diversas áreas da ciência e da computação. A manipulação de números em diferentes bases encontra aplicações práticas em criptografia, compressão de dados, codificação de informação e em diversas linguagens de programação.

O programa desenvolvido permite ao utilizador inserir um número em qualquer base entre 2 e 62 e obter a conversão do mesmo para outra base entre o mesmo intervalo (2 a 62)

Código

```
# TRABALHO 1
# João Andréz - 61066
# Carlos Ferreira - 71319
# Diogo Freitas - 90147
# Diogo Carvalho - 90247

import math

#Array of possible characters in numbers
CHAR_LIST = [ "0", "1", "2", "3", "4", "5", "6", "7", "8", "9",
              "A", "B", "C", "D", "E", "F", "G", "H", "I", "J", "K",
"L", "M", "N", "O", "P", "Q", "R", "S", "T",
              "U", "V", "W", "X", "Y", "Z", "a", "b", "c", "d", "e",
"f", "g", "h", "i", "j", "k", "l", "m", "n",
              "o", "p", "q", "r", "s", "t", "u", "v", "w", "x", "y",
"z"]

#Fractional part precision
PRECISION = 8

# Calls toBase10 or toBaseX, depending on the value of new_base
def changeBase(num: str, current_base: int, new_base: int):
    if new_base == 10:
        return toBase10(num, current_base)
    else:
        return toBaseX(num, current_base, new_base)

# Convert a number from current_base to new_base, up to base 62
# Expects number with "." separating integer from fractional part
def toBaseX(num: str, current_base: int, new_base: int):
```

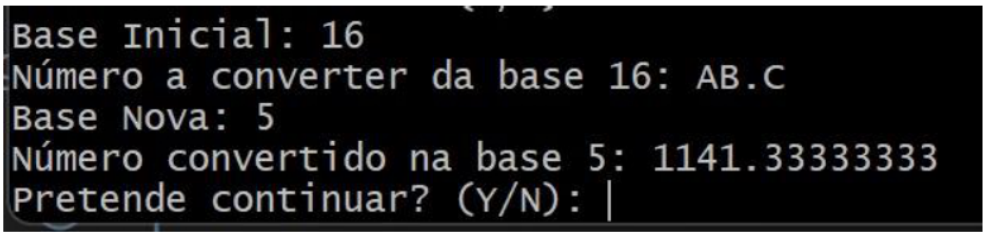
```
    new_num = ""

    if current_base != 10:                  # Check if number is
already in base 10
        num = toBase10(num, current_base)
    else:
        num = float(num)

    #-----INTEGER PART-----
    integer_part = math.floor(num)

    if integer_part == 0:
        new_num = "0"
    else:
        while integer_part != 0:
            digit = CHAR_LIST[integer_part % new_base]   # Get digit
from division remainder with new_base
            new_num = digit + new_num                    # Add digit
to the BEGINNING of the string
            integer_part //= new_base                    # Divide
integer_part by the new_base

    #------FRACTIONAL PART-------
    fractional_part = num % 1       # Get fraction part with
division remainder with 1
    if fractional_part != 0:        # Has a fractional part
        new_num += "."              # Add a "." to the end of the
string

    i = 0   # Counter for fractional part precision
    while fractional_part != 0 and i < PRECISION:
        integer_remainder = math.floor(fractional_part * new_base)
# Integer remainder
        digit = CHAR_LIST[integer_remainder]
# Get digit from CHAR_LIST
        fractional_part = fractional_part * new_base -
integer_remainder     # Set new fractional part
        new_num += digit
# Add the digit to the END of the string
        i += 1
# Increment counter for precision

    return new_num  # string

# Convert any number up to base 62 back to base 10
# Expects number with "." separating integer from fractional part
def toBase10(num: str, base: int):
    new_num = float(0)              # Float for storing following
values
    num = num.split(".")            # Split the integer part from
the fractional part in a string list

    #-----INTEGER PART-----
    integer_part = num[0]
```

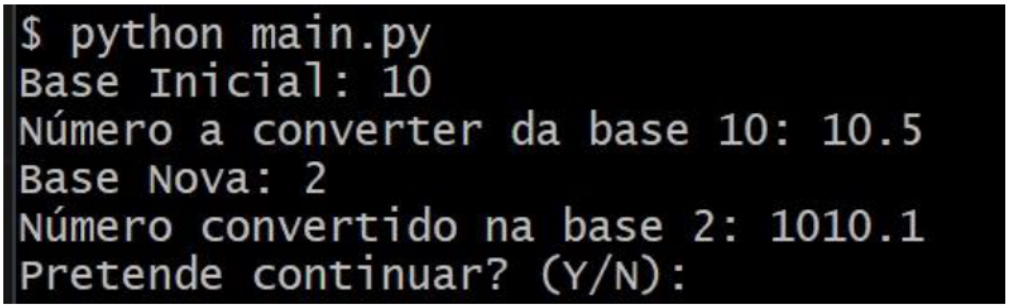
```
    for i, digit in enumerate(integer_part):        # Add each
character's value multiplied by base^position to the total float
value
        new_num += CHAR_LIST.index(digit) * base **
(len(integer_part) - i - 1)

    #-----FRACTIONAL PART------
    if len(num) > 1:                # Check if there's a fractional
part
        fractional_part = num[1]
        for i, digit in enumerate(fractional_part): # Add each
character's value multiplied by base^position to the total float
value
            new_num += CHAR_LIST.index(digit) * base ** ((i + 1) * -
1)

    return new_num  # float

# Number validation;
# Returns true for numbers with characters that belong to the base.
def numberValidation(base: int, num: str):
    num = num.replace(".", "")
    for digit in num:
        if CHAR_LIST.index(digit) >= base:
            return False
    return True

# Basic input function;
# Loops through current base, number and new base;
# Checks if 2 < bases < 63 (length of CHAR_LIST)
# Checks if number's characters belong to the current base
def inputNumber():
    # Input loop
    while True:
        negative_number = False

        # CURRENT BASE
        current_base = int(input("Base Inicial: "))
        if current_base > len(CHAR_LIST) or current_base < 2:
            print(f"Base ->{current_base}<- não está entre 2 e
{len(CHAR_LIST)}")
            continue

        # NUMBER
        number = input(f"Número a converter da base {current_base}:
")
        if number.find("-") != -1:              # Check for
negative number, saves state in flag negative_number
            negative_number = True
            number = number.replace("-", "")
        number = number.replace(",", ".")   # Force "." to separate
integer from fractional part
        if not numberValidation(current_base, number):
```

```
            print(f"Número ->{number}<- inválido na base
{current_base}")
            continue

        # NEW BASE
        new_base = int(input("Base Nova: "))
        if new_base > len(CHAR_LIST) or new_base < 2:
            print(f"Base ->{new_base}<- não está entre 2 e
{len(CHAR_LIST)}")
            continue

        # Heavy lifting function call
        new_num = changeBase(number, current_base, new_base)
        if negative_number:
            new_num = "-" + str(new_num)
        print(f"Número convertido na base {new_base}: {new_num}")

        # Loop if user doesn't input "N"
        if input(f"Pretende continuar? (Y/N): ") == "N":
            break

if __name__ == '__main__':
    inputNumber()
```

Resultados e Testes

Foram realizados diversos testes para demostrar a correta implementação do programa.





```
Base Inicial: 11
Número a converter da base 11: 1A.3
Base Nova: 16
Número convertido na base 16: 15.45D1745D
Pretende continuar? (Y/N): |
```

```
Base Inicial: 6
Número a converter da base 6: 55.3
Base Nova: 10
Número convertido na base 10: 35.5
Pretende continuar? (Y/N): |
```

```
Base Inicial: 10
Número a converter da base 10: 19.5
Base Nova: 12
Número convertido na base 12: 17.6
Pretende continuar? (Y/N): |
```

```
Base Inicial: 62
Número a converter da base 62: aZ.1
Base Nova: 5
Número convertido na base 5: 33032.00200200
Pretende continuar? (Y/N): |
```

```
$ python math.py
Base Inicial: 2
Número a converter da base 2: 0.1
Base Nova: 10
Número convertido na base 10: 0.5
Pretende continuar? (Y/N): |
```

```
Base Inicial: 11
Número a converter da base 11: A.5
Base Nova: 3
Número convertido na base 3: 101.11002110
Pretende continuar? (Y/N):
```

```
Base Inicial: 8
Número a converter da base 8: 123.4
Base Nova: 11
Número convertido na base 11: 76.55555555
Pretende continuar? (Y/N):
```

```
Base Inicial: 7
Número a converter da base 7: 12.3
Base Nova: 12
Número convertido na base 12: 9.5186A351
Pretende continuar? (Y/N):
```

```
Base Inicial: 16
Número a converter da base 16: A.F
Base Nova: 2
Número convertido na base 2: 1010.1111
Pretende continuar? (Y/N):
```

```
Base Inicial: 36
Número a converter da base 36: Z.1
Base Nova: 12
Número convertido na base 12: 2B.04000000
Pretende continuar? (Y/N):
```

```
Base Inicial: 4
Número a converter da base 4: 101.2
Base Nova: 62
Número convertido na base 62: H.V
Pretende continuar? (Y/N):
```

```
Base Inicial: 62
Número a converter da base 62: 0.Z
Base Nova: 10
Número convertido na base 10: 0.564516129032258
Pretende continuar? (Y/N): |
```

```
Base Inicial: 20
Número a converter da base 20: J.4
Base Nova: 9
Número convertido na base 9: 21.17171717
Pretende continuar? (Y/N): |
```

```
Base Inicial: 5
Número a converter da base 5: 12.34
Base Nova: 10
Número convertido na base 10: 7.76
Pretende continuar? (Y/N): |
```

```
Base Inicial: 62
Número a converter da base 62: Zz.9
Base Nova: 10
Número convertido na base 10: 2231.1451612903224
Pretende continuar? (Y/N): |
```

```
Base Inicial: 10
Número a converter da base 10: 10.25
Base Nova: 16
Número convertido na base 16: A.4
Pretende continuar? (Y/N):
```

```
Base Inicial: 12
Número a converter da base 12: 2B.A
Base Nova: 7
Número convertido na base 7: 50.55555555
Pretende continuar? (Y/N):
```