

**PROJETO INTERDISCIPLINAR**

**Título**

Subtítulo

**Alunos:**

|  |  |
| --- | --- |
| **RGM** | **Nome** |
| 28604288  29117321 | Fernanda S. Carillo  Felipe Moreira Maciel |
| 28805445 | Felipe Alves Ferreira |
| 29072123 | Rodrigo da S. Campos |
| 29052190 | Gabriel Sanches |

São Paulo

2022

**UNIVERSIDADE CRUZEIRO DO SUL**

**PROJETO INTERDISCIPLINAR**

**Título**

Subtítulo

Trabalho apresentado como parte do requisito para aprovação na Disciplina de Projeto Interdisciplinar do curso de Programação de Computadores e Organização e Arquitetura de Computadores da Universidade Cruzeiro do Sul.

**Orientadores:** Prof. Marco Antonio Sanches Anastacio e Prof. Luiz Antonio Brigatti Junior.

São Paulo

2022

Sumário

[1. Apresentação 4](#_Toc101357442)

[1.1 Justificativa e Motivação 4](#_Toc101357443)

[1.2 Dados do Programa. 4](#_Toc101357444)

[2 Requisitos de Programação de Computadores 5](#_Toc101357445)

[3 Requisitos de Organização e Arquitetura de Computadores 9](#_Toc101357446)

[4 Consideração finais 11](#_Toc101357447)

[5 BIBLIOGRAFIA 12](#_Toc101357448)

# 1. Apresentação

## 1.1 Justificativa e Motivação

O tema proposto teve como objetivo passar os conhecimentos adquiridos na matéria de Organização e Arquitetura de Computadores sobre a conversão de bases binarias, octal e hexa para decimal, com isso exercitando uma característica muito comum no ambiente profissional de converter processos físicos para serem executados por meio digital através da codificação de um programa.

## 1.2 Dados do Programa.

O programa recebe um número de base binarias, octal ou hexa e através das operações matemáticas realiza a conversão para a base decimal, assim possibilitando saber a conversão sem realizar as contas necessárias manualmente.

# 2 Requisitos de Programação de Computadores

Seguindo os requisitos solicitados na disciplina de programação de computadores não foi utilizado nenhuma função pronta que realize a conversão dos números para decimal. Foram feitas todas as funções de conversão em python conforme a figura 1 e a figura 2, ademais foi realizada um menu onde o usuário pode realizar a escolha de qual base (binário, hexadecimal e octal decimal) quer converter como demonstrado na figura 4.

Figura 1 - Código referente a função de validação

Texto

Descrição gerada automaticamente

Fonte - Imagem do autor

Figura 2 - Código referente a conversão da sequência

Texto

Descrição gerada automaticamente

Fonte - Imagem do autor

Figura 3 - Código referente a entrada de dados e validações necessárias

Texto

Descrição gerada automaticamente

Fonte - Imagem do autor

Figura 4 - Saída de dados

Texto

Descrição gerada automaticamente

Fonte - Imagem do autor

Figura 5 - Exemplo de conversão de hexadecimal para decimal

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Fonte – Imagem retirada do site ToConvert.¹

# 3 Requisitos de Organização e Arquitetura de Computadores

Como demostrado nas aulas de OAC (Organização e arquitetura de computadores) utilizamos a fórmula matemática como na figura 7 para realizar a função em python que realiza os cálculos de conversão, passamos a fórmula para um código que lê cada item da sequência e de acordo com a base (binário, hexadecimal e octal decimal) realiza a conversão conforme a figura 6. A figura 8 demonstra a conversão do mesmo exemplo presente na figura 7.

Figura 6 - Código referente a conversão da sequência

Texto

Descrição gerada automaticamente

Fonte - Imagem do autor

Figura 7 – Exemplo da fórmula para conversão de binário para decimal

Texto

Descrição gerada automaticamente

Fonte – Slide aula: 003 do professor Luiz Antonio Brigatti Junior – Arquitetura e organização de computadores

Figura 8 – Saída do programa convertendo uma sequência binaria em decimal

Texto

Descrição gerada automaticamente

Fonte - Imagem do autor

# 4 Consideração finais

Uma das dificuldades encontradas no trabalho foi como iria validar as letras no caso de hexadecimal como valores numéricos. A forma que encontramos de fazer isso foi utilizando um dicionário com os valores armazenados para buscá-los posteriormente. Outro ponto interessante ainda no hexadecimal, foi na função de validar a sequência, onde seria necessário percorrer todos os itens e verificar se cada item é valido, porém o *range* não cria sequência entre letras, então foi preciso utilizar o *ord* que converte uma letra para *unicode* e assim o python consegue criar o *range* entre dois *unicodes.*

# 5 BIBLIOGRAFIA

PYTHON SOFTWARE FOUNDATION. Funções embutidas. *In*: **Funções embutidas**. [*S. l.*], 2022. Disponível em: https://docs.python.org/pt-br/3/library/functions.html?highlight=ord#ord. Acesso em: 7 abr. 2022.

PYTHON SOFTWARE FOUNDATION. Unicode HOWTO. *In*: **Unicode HOWTO**. [*S. l.*], 2022. Disponível em: https://docs.python.org/3/howto/unicode.html. Acesso em: 7 abr. 2022.

PYTHON ENHANCEMENT PROPOSALS. **PEP 636 – Structural Pattern Matching: Tutorial**. [*S. l.*], 27 fev. 2022. Disponível em: https://peps.python.org/pep-0636/. Acesso em: 14 abr. 2022.

¹CONVERTER HEXADECIMAL PARA DECIMAL. [*S. l.*], [21-]. Disponível em: https://www.to-convert.com/pt/numero/converter-hexadecimal-para-decimal.php. Acesso em: 14 abr. 2022.