

# Operações com Vetores em C



# Conteúdo

<b>1</b>	<b>Introdução</b>	<b>4</b>
1.1	Objetivos do Trabalho . . . . .	4
1.2	Estrutura do Relatório . . . . .	4
<b>2</b>	<b>Planeamento e Execução</b>	<b>5</b>
2.1	Tecnologias Utilizadas . . . . .	5
2.2	Estrutura do Projeto . . . . .	5
2.3	F . . . . .	5
2.4	Detalhes de Implementação . . . . .	5
2.5	Exemplos . . . . .	6
2.6	Dependências . . . . .	6
<b>3</b>	<b>Testes</b>	<b>7</b>
3.1	Detalhes . . . . .	7
3.2	Execução dos Testes . . . . .	7
3.3	Resultados . . . . .	7
<b>4</b>	<b>Conclusões</b>	<b>8</b>
4.1	Ficheiros . . . . .	8

# Resumo

Este relatório descreve um programa em C que realiza operações com vetores. O programa foi feito a separar as funções (no ficheiro **functions-PL403.c** e **funcions-PL403.h**) do programa principal (**main-PL403.c**). O programa pede ao utilizador  $N$  números inteiros (onde  $N$  é definido pela constante **TAM**), validando a entrada entre -3 a 27. O programa disponibiliza um menu interativo para manipulações de vetores e transformações com matrizes. A documentação do programa foi gerada de forma automática utilizando o Doxygen e a compilação é feita através um makefile.

**Palavras-chave:** C, vetores, matrizes, funções,makefile, Doxygen.

# Acrónimos

**TAM** Tamanho do vetor

# Capítulo 1

## Introdução

### 1.1 Objetivos do Trabalho

O objetivo principal do trabalho é desenvolver um programa que cumpra os seguintes requisitos:

- Implementar um programa em C que peça ao utilizador  $N$  (onde  $N=\mathbf{TAM}=14$ ) números inteiros no intervalo de -3 a 27.
- Verificar a entrada para todos os números.
- Separar o código em dois ficheiros diferentes (**.c** e **.h**).
- Apresentar um menu de operações para manipulação do vetor.
- Incluir uma funcionalidade de ajuda acessível via menu (opção 7) ou através da flag **-help**.
- Gerar a documentação do código de forma automática através do Doxygen.

### 1.2 Estrutura do Relatório

Este relatório está estruturado da seguinte forma:

- O Capítulo 2 tem a estrutura do projeto, as tecnologias utilizadas e as funcionalidades implementadas.
- O Capítulo 3 tem os testes realizados para avaliar o programa.
- O Capítulo 4 resume os resultados.

# Capítulo 2

## Planeamento e Execução

### 2.1 Tecnologias Utilizadas

- Linguagem de Programação: C;
- Compilador: GCC;
- **Makefile** para automatização da compilação e limpeza.
- Ferramenta de Documentação: Doxygen para geração de documentação HTML.

### 2.2 Estrutura do Projeto

O projeto tem uma estrutura que separa os ficheiros:

- **main-PL403.c**: Contém a função **main()**;
- **functions-PL403.h**: Ficheiro cabeçalho que declara a constante **TAM** e as todas as funções que manipulam os vetores;
- **functions-PL403.c**: Ficheiro que contém o corpo das funções.

A compilação do projeto é automatizada pelo **Makefile**, que permite a compilação do executável e a geração da documentação com um comando (**make**).

### 2.3 F

lag **–help** Além do menu interativo, foi adicionada uma funcionalidade de ajuda.

- A função **main()** foi modificada para aceitar argumentos ("argumento" e "flag").
- Se o programa for executado com o argumento **–help** **./main-PL403.exe –help**, o programa chama a função **ajuda()** e termina sem pedir a introdução do vetor.

### 2.4 Detalhes de Implementação

O programa foi construído em torno de um vetor de tamanho(TAM)=14. As funcionalidades implementadas são:

- **ordenarvetor**: Ordena o vetor de inteiros por ordem crescente (Método Bubble Sort).

- **simetricovetor**: Calcula e apresenta o vetor que contém os valores simétricos dos elementos originais ordenando-os.
- **somavetor**: Calcula a soma dos elementos da primeira metade do vetor com os elementos da segunda metade.
- **multiplovetor**: Devolve os valores do vetor que estão em posições múltiplas de três.
- **aleatoriovetor**: Retorna um elemento aleatório do vetor.
- **matrizvetor**: Constrói uma matriz  $N \times N$ , onde cada linha é composta por permutações dos valores do vetor.
- **raizvetor**: Calcula a raiz quadrada de cada elemento do vetor.
- **misturavetor**: Mistura a primeira metade do vetor original com um novo vetor de números inteiros introduzido pelo utilizador.
- **mdcvetor**: Calcula o Máximo Divisor Comum (**mdc**) de dois elementos consecutivos do vetor.
- **matriz2vetor**: Constrói uma matriz que resulta do produto de elemntos do vetor original com o vetor original ordenado de forma crescente.
- **transpostavetor**: Constrói e apresenta a matriz transposta da matriz gerada em **matriz2vetor**.
- **ajuda**: Mostra o menu de ajuda do programa.

## 2.5 Exemplos

Para um vetor de exemplo [10, -1, 5, 2, 7, 3], a função **somavetor** retorna:

- $10+2=12$
- $-1+7=6$
- $5+3=8$

Output: 12, 6, 8

## 2.6 Dependências

O projeto depende das bibliotecas padrão de C (**stdio.h**, **stdlib.h**, **time.h**, **math.h**, **string.h**) e do **makefile** para compilação.

# Capítulo 3

## Testes

### 3.1 Detalhes

Testes para cada função:

- Validação de Entrada: Teste de valores válidos (dentro de  $[-3, 27]$ ) e inválidos (fora do intervalo) para verificar a rejeição e a repetição do pedido.
- Verificação Matemática: Verificação da saída para operações como soma (**somavetor**), cálculo de **mdc** e raízes (**raizvetor**).
- Funcionalidade do programa: Teste de por exemplo; **ordenarvetor** e teste da flag **-help**.

### 3.2 Execução dos Testes

Cada função foi executada separadamente para garantir que todas as operações estavam corretas. A flag **-help** foi testada na linha de comandos.

### 3.3 Resultados

Os testes confirmaram:

- A validação de entrada não deixa inserir números fora do intervalo.
- As operações matemáticas e de manipulação dão resultados corretos.
- A funcionalidade **-help** está a funcionar.
- Os elementos aleatórios mudam a cada execução.



# Capítulo 4

## Conclusões

Todos os objetivos foram atingidos com sucesso.

### 4.1 Ficheiros

- main-PL403.c
- functions-PL403.c
- functions-PL403.h
- makefile
- Doxyfile

# Bibliografia

- Kernighan, B. W., & Ritchie, D. M. (1988). *The C Programming Language*. Prentice Hall.
- Curso de C - Playlist no YouTube.: <https://www.youtube.com/watch?v=2w8GYzBjNj8&list=PLpaKFn4Q4GMOBAeq...>
- <https://labex.io/pt/tutorials/cpp-how-to-use-compiler-flags-correctly-434220>
- <https://www.doxygen.nl/manual/markdown.html>
- <https://tableless.github.io/iniciantes/manual/html/estruturabasica.html>
- <https://jothepro.github.io/doxygen-awesome-css>