Universidade da Beira Interior
Departamento de Informática

Nº 8 - 2024: *Relatório Técnico*

Elaborado por:

Bruno Salvador - 53588
Dinis Miranda - 54227
Duarte Rufino - 54539
Gabriel Teixeira - 53781
João Gonçalves - 53741
João Silva - 53748

Orientador:

Professor Doutor Tiago Simões

15 de dezembro de 2024

Agradecimentos

Em primeiro lugar, gostaríamos de expressar a nossa profunda gratidão aos nossos orientadores, Professor Doutor Tiago Roxo e Professor Doutor Tiago Simões, pela orientação, paciência e disponibilidade quanto para o desenvolvimento do trabalho como as aulas.

Agradecemos também aos nossos colegas de curso e amigos, que nos apoiaram e contribuíram com a sua ajuda neste trabalho. E também sem esquecer ás nossas famílias pela ajuda e apoio neste momento marcante das nossas vidas.

A todos, o nosso muito obrigado.

Conteúdo

Lista de Excertos de Código

Acrónimos

LP	Laboratório de Programação
UBI	Universidade da Beira Interior
VS Code	*Visual Studio Code*

Glossário

LaTeX *LaTeX*(lê-se latec) fornece um conjunto de macros alto-nível que torna mais fácil e rápida a produção de documentos em TeX e é utilizado para produzir todo o tipo de documentos como por exemplo livros, relatórios e artigos. 10

Visual Studio Code *Microsoft Visual Code* é um ambiente de desenvonvolvimento integrado desenvolvido pela Microsoft amplamente utilizado por programadores e desenvolvedores para escrever, editar e gerenciar código em diversas linguagens de programação. 10

Capítulo

1

Introdução

1.1 Enquadramento

Este relatório foi desenvolvido no âmbito da unidade curricular de Laboratórios de Programação do curso Licenciatura em Engenharia Informática da Universidade da Beira Interior (UBI) e tem como finalidade implementar um programa que peça ao utilizador 20 números inteiros e os guarde num vetor, para posteriormente providenciar forma de calcular algumas estatísticas ou fazer operações sobre esses valores. O projeto enquadra-se também na Álgebra e no Cálculo. Este também é importante porque ajuda a reforçar conhecimentos técnicos e a desenvolver habilidades para resolver problemas, o que é essencial para criar soluções práticas na área da tecnologia.

1.2 Motivação

Abordar o problema neste projeto é essencial, ao permitir aplicar de forma prática os conceitos de programação aprendidos, integrando-os na criação de um código de maior complexidade que simula desafios reais da Engenharia Informática. Além disso, o trabalho em grupo promove o desenvolvimento de competências importantes, como organização, colaboração e divisão de tarefas, fundamentais para projetos de larga escala. A utilização do LaTeX para documentar o relatório reforça a capacidade de comunicação técnica, garantindo a clareza e a estruturação necessária para justificar decisões e facilitar a manutenção do código. Assim, o projeto combina teoria, prática e trabalho colaborativo, preparando os alunos para desafios técnicos e profissionais futuros.

1.3 Objetivos

O objetivo principal deste projeto é desenvolver um programa em C que solicite ao utilizador 20 números inteiros, assegurando que cada valor esteja entre oito e 29, e os armazene num vetor. Posteriormente, o programa deverá apresentar um menu que permita ao utilizador realizar diversas operações e cálculos estatísticos sobre os valores inseridos, como ordenar o vetor, calcular somas específicas, gerar matrizes baseadas no vetor, entre outras funcionalidades.

Mas para além deste objetivo ainda existe mais um com uma versão mais elaborada do projeto que implementa funcionalidades adicionais que ampliam as capacidades do programa, tais como: permitir a leitura de um novo vetor e combiná-lo com o anterior, calcular o mínimo múltiplo comum de pares de números consecutivos, gerar e manipular matrizes resultantes do produto de vetores, e fornecer uma página de ajuda acessível tanto pelo menu quanto por meio de uma *flag* na linha de comandos. Estas funcionalidades avançadas têm como propósito enriquecer a experiência do utilizador e oferecer ferramentas mais robustas para a análise dos dados inseridos.

1.4 Organização do Documento

De modo a refletir o trabalho feito, este documento encontra-se estruturado da seguinte forma:

1. O primeiro capítulo – **Introdução** – Apresenta o projeto, a motivação para a sua escolha, o enquadramento para o mesmo, os seus objetivos e a respetiva organização do documento.

2. O segundo capítulo – **Estado da Arte** – Aborda as ferramentas fundamentais aplicadas no desenvolvimento deste projeto e analisa outros trabalhos semelhantes com a mesma lógica.

3. O terceiro capítulo – **Tecnologias Utilizadas** – Explica todas as ferramentas usadas ao longo do projeto e os conceitos que precisam de mais detalhes.

4. O quarto capítulo – **Implementação e Testes** – Descreve os passos detalhados na criação do programa, explicando as etapas presentes do trabalho.

5. O quinto capítulo – **Conclusões e Trabalho Futuro** – Apresenta o desfecho do trabalho, analisando os resultados obtidos e sugerindo possíveis melhorias e aplicações futuras.

1.5 Delegação de Tarefas

O trabalho foi dividido em tarefas da seguinte forma:

- Código: Bruno Salvador;
- Documentação (Doxygen): Gabriel Teixeira;
- Edição do Relatório: Dinis Miranda e Duarte Rufino;
- Revisão do Trabalho: João Gonçalves e João Silva.

Capítulo

2

Estado da Arte

2.1 Introdução

Neste capítulo são abordadas as ferramentas essenciais para o desenvolvimento do trabalho. Este capítulo encontra-se estruturado da seguinte forma: a secção 2.2 descreve o estado-da-arte em termos de tecnologias importantes para o projeto e a secção 2.3 descreve outros trabalhos semelhantes ao que foi desenvolvido neste trabalho.

2.2 Estado Da Arte

2.2.1 Validação de Entrada

O processo de validação de entrada é necessário quando existe a obrigação de assegurar que os dados que o utilizador fornecer sejam viáveis, isto é, que estejam num intervalo de números específico.

No contexto deste projeto, os valores inseridos pelo utilizador devem estar entre oito e 29. Caso os números escolhidos não pertençam ao intervalo previamente definido, esses serão rejeitados, até que sejam introduzidos valores aceitáveis.

2.2.2 Matrizes e Operações Matemáticas

Este projeto envolve noções de álgebra linear, como a manipulação de matrizes. Isto deve-se ao fato de que um dos objetivos a ser concretizado é a geração de uma matriz 20×20, em que as suas linhas são permutações do vetor construído inicialmente.

Outro aspeto a salientar é o envolvimento do cálculo de funções trigonométricas, como o cosseno (cos) dos elementos da segunda metade do vetor.

A implementação desta funcionalidade requer o uso de bibliotecas matemáticas que forneçam as operações trigonométricas necessárias. Além disso, a geração de um vetor aleatório e a realização de operações sobre ele são técnicas comuns em áreas como a estatística e a análise de dados.

2.3 Trabalhos Relacionados

O *MatrixCalc*[1] é uma ferramenta *online* que permite realizar operações com matrizes e vetores, tais como soma, multiplicação e cálculo de determinantes. Embora ambos os projetos compartilhem o objetivo de simplificar a manipulação de matrizes, o projeto descrito neste relatório oferece funcionalidades mais específicas, como validação de entrada, geração de matrizes com permutações e cálculos personalizados, como o cosseno de elementos do vetor. Além disso, o *MatrixCalc* não conta com aspetos interativos, como menus ou geração de valores aleatórios.

O *Wolfram Alpha*[2] é uma plataforma que realiza cálculos complexos e responde a perguntas em várias áreas, como matemática e tecnologia. Assim como no trabalho, ele trabalha com dados fornecidos pelo utilizador para realizar operações como ordenação e cálculo de cossenos.

Enquanto o *Wolfram Alpha* é uma ferramenta ampla e automatizada, o trabalho foca na interação com o utilizador, manipulando um vetor de 20 números com validação de entrada e um menu para operações específicas.

A biblioteca *NumPy*[3], do *Python*[4], é amplamente usada para manipulação de matrizes e vetores em projetos de aprendizado de máquina e estatística. Embora compartilhe com o projeto atual a capacidade de realizar cálculos eficientes de álgebra linear, o *NumPy* exige programação e não oferece uma interface interativa. O projeto atual, por outro lado, foca em funcionalidades específicas, como validação de entrada e operações personalizadas com matrizes, o que o torna mais acessível para o utilizador final.

2.4 Conclusões

Neste capítulo, foram analisadas ferramentas como o *MatrixCalc*, o *Wolfram Alpha* e o *NumPy*, que influenciaram o desenvolvimento do projeto. Embora compartilhem algumas funcionalidades, o projeto descrito se destaca por focar numa abordagem mais interativa e personalizada, com validação de entrada, geração de matrizes a partir de permutações e cálculos específicos. Essas características tornam o projeto mais acessível ao utilizador e formam a base para as próximas etapas do desenvolvimento.

Capítulo

3

Tecnologias e Ferramentas Utilizadas

3.1 Introdução

Para a realização deste trabalho foram necessárias diversas ferramentas, das quais o *Linux*, sendo o sistema operativo onde roda o executável a partir do *Makefile*, Visual Studio Code *(VS Code)* para a realização do código em linguagem C[5], o *Doxygen* para gerar automaticamente documentação relativa ao código, o *Overleaf* que, através do LaTeX, se fez o relatório.

3.2 Ferramentas

3.2.1 *Linux*

O *Linux* é um sistema operativo de código aberto baseado no núcleo *Linux*, desenvolvido por Linus Torvalds em 1991. Ele é amplamente utilizado em servidores, dispositivos móveis e computadores pessoais devido à sua estabilidade, segurança e flexibilidade. O *Linux* é gratuito e permite modificações, o que o torna uma escolha popular para desenvolvedores e empresas. A comunidade por trás do *Linux* garante um ecossistema rico em ferramentas e recursos, além de um suporte técnico robusto.

3.2.2 Terminal

O Terminal é uma interface de utilizador baseada em texto, proporcionando ao utilizador um canal direto para interagir com o sistema operativo subjacente. Dispensando a necessidade de interfaces gráficas, o Terminal opera através da interpretação de comandos de texto, estabelecendo uma comunicação precisa e eficiente entre o utilizador e o sistema.

3.2.3 LaTeX

O LaTeX é um sistema de preparação de documentos desenvolvido na década de 1980 por Leslie Lamport. Ao escrever, o escritor usa texto simples, ao invés do texto formatado encontrado em processadores de texto como *Microsoft Word* e *LibreOffice Writer*. O escritor utiliza padrões de marcação para definir a estrutura desejada do documento, formatando assim o texto em todo um documento e incluindo citações e referências de forma simples.

3.2.4 *Overleaf*

O Overleaf é um editor de LaTeX online que facilita a edição de documentos em LaTeX, permitindo a colaboração em tempo real e tornando mais ágil a criação de textos como relatórios técnicos. Além disso também permite a integração com ferramentas de gestão bibliográfica, tais como *Zotero*, *JabRef*, *Mendeley* e *EndNote*, facilitando as citações e referências de forma prática e organizada.

3.2.5 *VS Code*

O Visual Studio Code é um editor de código leve e gratuito, criado pela Microsoft, que se tornou bastante popular entre desenvolvedores. Ele suporta diversas linguagens de programação, como *JavaScript*, *Python* e C++[6], e oferece uma vasta coleção de extensões que permitem personalizar a experiência de codificação. Com ferramentas integradas, como depuração, controle de versão *Git*[7] e suporte a ambientes de desenvolvimento remoto, o *VS Code* é uma escolha prática e eficiente para quem busca agilidade e flexibilidade no desenvolvimento de *software*.

3.2.6 *Make*

O *Make* é uma ferramenta fundamental em ambientes de desenvolvimento, especialmente em projetos que envolvem compilação de código-fonte. Este automatiza o processo de construção de *software*, tornando-o mais eficiente e menos propenso a erros.Através da definição de um *Makefile*, o *Make* permite especificar as dependências entre arquivos e as ações necessárias para compilar e gerar o *software*, garantindo que apenas os componentes alterados sejam recompilados. A utilização do *Make* melhora a organização do código, promovendo uma estrutura mais clara.

3.2.7 *Doxygen*

O *Doxygen* é uma ferramenta poderosa e versátil utilizada para gerar documentação de código-fonte automaticamente. Este analisa o código-fonte e, a partir de comentários especiais inseridos, cria uma documentação completa e profissional, facilitando a compreensão e manutenção do seu projeto. Além disso, suporta várias linguagens de programação, como C, C++ e *Python*, e permite gerar documentação em diferentes formatos, como HTML, LaTeXe PDF.

3.3 Conclusões

Em síntese, as ferramentas apresentadas neste capítulo foram essenciais para o desenvolvimento do trabalho. O *Linux* e o Terminal forneceram um ambiente sólido para desenvolvimento, enquanto o *VS Code* facilitou a programação em C. O *Make* automatizou a compilação, e o *Doxygen* gerou documentação. Para o relatório, o LaTeXe o *Overleaf* permitiram a criação de um documento técnico bem estruturado e eficiente. A integração dessas tecnologias garantiu um bom fluxo de trabalho.

Capítulo

4

Implementação e Testes

4.1 Introdução

Durante a fase de implementação e de testes do código, foi utilizado o *VS Code* juntamente com o *Make* para a simplificação e aceleração do processo de compilação de código.

O projeto foi dividido em três ficheiros: `main-team-8.c`, `functions-team-8.c` e `functions-team-8.h`. Na secção seguinte são apresentados alguns excertos destes mesmos ficheiros.

4.2 Trechos de Código

```
void Menu(int *OriginalArray) {
    int MenuChoice, ValidMenuChoice;
    PrintIntegerArray(OriginalArray, "Vetor Original:", ArraySize);
    printf("Menu:\n");
    printf("1  - Vetor ordenado por ordem crescente\n");
        ⋮
    printf("12 - Sair\n");

    // Guardar se escolha é válida
    ValidMenuChoice = scanf("%d", &MenuChoice);
    ClearTerminal();
    // Verificar se existe uma escolha válida
    if (ValidMenuChoice == 1) {
        switch (MenuChoice) {
        case 1:
            ⋮
        default:
            printf("Opção Inexistente\n\n");
            // Abrir Menu
            Menu(OriginalArray);
        }
    } else {
        // Limpar Input
        ClearInput();
        // Abrir Menu
        Menu(OriginalArray);
    }
}
```

Excerto de Código 4.1: Mostrar Menu e Verificar Opção escolhida.

```
int main(int argc, char **argv) {
    // Iniciar Geração Aleatória com SEED baseada no tempo
    srand(time(NULL));
    for (int i = 0; i < argc; i++) {
        // Verificar se o argumento introduzido é "--help"
        if (strcmp(argv[i], "--help") == 0) {
            // Mostrar Ajuda
            Help(ArraySize, Minimum, Maximum);
            printf("Pressione Enter para fechar a Ajuda\n");
            // Esperar por Input do utilizador para fechar a ajuda
            getchar();
            ClearTerminal();
            // Terminar Programa com Sucesso
            return 0;
        }
    }
    // Alocar Memória para o Array
    OriginalArray = AllocateIntegerArray(OriginalArray, ArraySize);
    // Preencher o Array
    RequestArray(OriginalArray, ArraySize, Minimum, Maximum);
    // Abrir Menu
    Menu(OriginalArray);
}
```

Excerto de Código 4.2: Verificação dos argumentos e preenchimento do Vetor.

```
int *RequestArray(int *Array, int ArraySize, int Minimum, int Maximum) {
    int Number, ValidNumber;
    ClearTerminal();
    for (int i = 0; i < ArraySize;) {
        printf("Insira um número inteiro entre %d e %d (%dº Número): ",
            Minimum, Maximum, i + 1);
        // Guardar número de caráteres númericos válidos introduzidos
        ValidNumber = scanf("%d", &Number);
        ClearTerminal();
        // Verificar se número introduzido é um caráter não númerico
        if (ValidNumber == 1) {
            // Verificar se número introduzido está entre um Mínimo e um
                Máximo
            if (Number >= Minimum && Number <= Maximum) {
                // Guardar número no Array
                Array[i] = Number;
                i++;
            } else {
                printf("O número deve estar compreendido entre %d e %d.\
                    n", Minimum, Maximum);
            }
        } else {
            // Limpar Input em caso do caráter introduzido não ser um nú
                mero
            ClearInput();
        }
    }
    ClearTerminal();
    return Array;
}
```

Excerto de Código 4.3: Pedir, Verificar e Guardar Números Introduzidos.

```
int *AllocateIntegerArray(int *IntegerArray, int ArraySize) {
    IntegerArray = (int *)calloc(ArraySize, sizeof(int)); // Alocar Memó
        ria para um Array de Inteiros
    return IntegerArray;
}
```

Excerto de Código 4.4: Alocar Memória para um Vetor de Inteiros.

```
void PrintIntegerArray(int *Array, char *Text, int ArraySize) {
    printf("%s\n", Text);
    for (int i = 0; i < ArraySize; i++) {
        printf(" %d", Array[i]);
    }
    printf("\n\n");
}
```

Excerto de Código 4.5: Mostrar Vetor

```
int *SortAscendingOrder(int *Array, int *IntegerArray, int ArraySize) {
    // Alocar Memória para o Array
    IntegerArray = CopyArray(Array, AllocateIntegerArray(IntegerArray,
        ArraySize), ArraySize);
    int k;
    do {
        k = 0;
        for (int i = 0; i < ArraySize - 1; i++) {
            // Verificar se o número i é maior que o número i+1
            if (IntegerArray[i] > IntegerArray[i + 1]) {
                // Guardar número i na variável temporária
                int Temp = IntegerArray[i];
                // Colocar número i+1 na posição do número i
                IntegerArray[i] = IntegerArray[i + 1];
                // Colocar número na variável temporária na posição i+1
                IntegerArray[i + 1] = Temp;
                k++;
            }
        }
    } while (k != 0); // Repetir enquanto não ordenado
    return IntegerArray;
}
```

Excerto de Código 4.6: Ordenar Vetor por Ordem Crescente

```
int *AddFirstSecondHalf(int *Array, int *IntegerArray, int ArraySize) {
    // Alocar Memória para o Array
    IntegerArray = AllocateIntegerArray(IntegerArray, ArraySize / 2);
    for (int i = 0; i < ArraySize / 2; i++) {
        // Somar número i com número i+ArraySize/2
        IntegerArray[i] = Array[i] + Array[i + ArraySize / 2];
    }
    return IntegerArray;
}
```

Excerto de Código 4.7: Adicionar Primeira metade à Segunda Metade do Vetor.

```
int *ShuffleArray(int *Array, int ArraySize) {
    for (int i = ArraySize - 1; i > 0; i--) {
        // Gerar posição aleatória para a qual o número vai ocupar
        int j = rand() % ArraySize;
        // Guardar número na posição gerada aleatóriamente na variável
            temporária
        int Temp = Array[j];
        // Colocar número i na posição do número j
        Array[j] = Array[i];
        // Colocar número na variável temporária na posição i
        Array[i] = Temp;
    }
    return Array;
}

int **GeneratePermutedMatrix(int *Array, int **Matrix, int ArraySize) {
    Matrix = AllocateMatrix(Matrix, ArraySize);             // Alocar
       Memória para o Array Bidimensional
    CopyArray(Array, Matrix[0], ArraySize);                 // Copiar
        Array para a primeira linha do Array Bidimensional
    for (int i = 1; i < ArraySize; i++) {                   // Repetir
        para o Array todo excluindo a primeira linha
        Matrix[i] = CopyArray(Array, Matrix[i], ArraySize); // Copiar
            Array para a linha i do Array Bidimensional
        ShuffleArray(Matrix[i], ArraySize);                 // Baralhar
            Linha i do Array
    }
    return Matrix;
}
```

Excerto de Código 4.8: Gerar Matriz de Permutações.

Vale realçar que a geração aleatório de números (rand()), usada no Excerto de código 4.8, sendo esta baseada no tempo, a mesma gera um número igual se esta for executada no mesmo segundo.

```
int *PositionsMultipleof3(int *Array, int *IntegerArray, int ArraySize)
    {
    // Alocar Memória para o Array
    IntegerArray = AllocateIntegerArray(IntegerArray, ArraySize / 3);
    for (int i = 0; i * 3 + 2 < ArraySize; i++) {
        // Guardar os números em posições múltiplas de 3 num Array
        IntegerArray[i] = Array[i * 3 + 2];
    }
    return IntegerArray;
}
```

Excerto de Código 4.9: Posições Múltiplas de três.

Ao escrever a função demonstrada no Excerto de código 4.9 assumimos que o primeiro elemento do Vetor ocupa a posição um.

```
int **ProductBetweenTwoArrays(int *Array, int *IntegerArray, int **
    Matrix, int ArraySize, int Minimum, int Maximum) {
    // Alocar Memória para o Array
    IntegerArray = AllocateIntegerArray(IntegerArray, ArraySize);
    // Alocar Memória para o Array Bidimensiona
    Matrix = AllocateMatrix(Matrix, ArraySize);
    for (int i = 0; i < ArraySize; i++) {
        // Gerar número Aleatório entre dois número e guardar num Array
        IntegerArray[i] = rand() % (Maximum + 1 - Minimum) + Minimum;
  }
    for (int i = 0; i < ArraySize; i++) {
     for (int j = 0; j < ArraySize; j++) {
            // Calcular produto entre os dois Arrays e Guardar no Array
                Bidimensional
            Matrix[i][j] = IntegerArray[i] * Array[j];
        }
    }
    DisposeIntegerArray(IntegerArray); // Libertar Memória do Array
    return Matrix;
}
```

Excerto de Código 4.10: Produto entre o vetor Introduzido e um vetor aleatório.

4.3 Conclusões

Os exemplos de código apresentados mostram que é importante saber programar em C, mas também escrever de forma clara e organizada. Usar comentários no código ajuda muito, porque facilita a leitura e o entendimento por outras pessoas. Além disso, trocar ideias e trabalhar em equipa é essencial, já que há muitas maneiras diferentes de resolver os mesmos problemas em programação.

Capítulo

5

Conclusões e Trabalho Futuro

5.1 Conclusões Principais

Esta secção aborda as principais conclusões obtidas ao longo do desenvolvimento do trabalho.

Quais foram as conclusões principais alcançadas pelo(a) aluno(a) ao final deste trabalho?

Com os aprendizados e as informações adquiridas durante o semestre na unidade curricular de Laboratório de Programação (LP), foi possível realizar este trabalho com relativa facilidade. O conhecimento consolidado ao longo das aulas e práticas foi fundamental para alcançar os objetivos propostos, permitindo o desenvolvimento eficiente e estruturado das atividades.

5.2 Trabalho Futuro

Esta secção responde às seguintes questões:

O que ficou por fazer e por quê?

Felizmente, todos os objetivos propostos para o programa foram concluídos com êxito. Não restaram pendências ou funcionalidades incompletas, e os erros encontrados durante o desenvolvimento foram devidamente corrigidos. Assim, o programa foi finalizado e está funcional, sem quaisquer impedimentos para sua execução.

O que seria interessante fazer, mas não foi implementado por não ser o objetivo principal deste trabalho?

No contexto deste trabalho, todos os conteúdos lecionados na unidade curricular foram devidamente incorporados, não deixando lacunas para melhorias adicionais. Contudo, há sempre oportunidades para explorar funcionalidades extras ou aprimoramentos que poderiam ser interessantes, mas que excederiam os objetivos inicialmente definidos.

Em que outros cenários ou aplicações o trabalho aqui descrito pode ser útil, e por quê?

A realização deste trabalho proporcionou um aprofundamento significativo nos conhecimentos sobre elaboração de relatórios no *Overleaf*, uma habilidade que será essencial em futuros projetos académicos e profissionais. Além disso, a experiência prática adquirida no uso da linguagem C se mostrou indispensável para a resolução de problemas e desenvolvimento de códigos robustos, abrindo caminhos para aplicações em projetos futuros e em diferentes contextos relacionados ao desenvolvimento de *software*.

Bibliografia

[1] Calculadora de Matrizes — matrixcalc.org. https://matrixcalc.org/pt/. [Accessed 15-12-2024].

[2] Wolfram|Alpha: Making the world's knowledge computable — wolframalpha.com. https://www.wolframalpha.com/. [Accessed 15-12-2024].

[3] NumPy - — numpy.org. https://numpy.org/. [Accessed 15-12-2024].

[4] Welcome to Python.org — python.org. https://www.python.org/. [Accessed 15-12-2024].

[5] DevDocs 2014; C documentation — devdocs.io. https://devdocs.io/c/. [Accessed 15-12-2024].

[6] DevDocs 2014; C++ documentation — devdocs.io. https://devdocs.io/cpp/. [Accessed 15-12-2024].

[7] Git — git-scm.com. https://git-scm.com/. [Accessed 15-12-2024].