Faculdade de Ciências Exatas e da Engenharia

UNIVERSIDADE da MADEIRA

2021/2022

Estruturas de Dados e Algoritmos

Docente: Filipe Quintal

Projeto 2 - Plantação EDA

Grupo 5

Licenciatura em Engenharia Informática

Índice

Introdução e Objetivos

Neste projeto o tema e o objetivo são quase os mesmos que do primeiro projeto, desenvolver um programa em C++ que simule o funcionamento de uma horta (“PlantaçãoEDA”). Com este objetivo em mente, procurámos solucionar este problema da melhor maneira possível, utilizando apenas o código imprescindível sem que o bom funcionamento do projeto fosse comprometido. Para tal utilizamos o IDE Visual Studio sob a forma de descrição em C++.

O programa terá que simular as seguintes funcionalidades propostas pelos docentes:

Inicialização das hortas;

Inicialização dos produtos;

Funcionamento;

Colheita manual;

Atualizar tempo de rega;

Gravar Plantação;

Carregar Plantação;

Imprimir Plantação;

Imprimir produtos colhidos;

Criar nova área;

Alterar área;

Ao contrário do primeiro projeto, foi proposta a utilização de listas ligadas e árvores binárias em vês de arrays.

Implementação

Structs

Para o início do nosso projeto começámos por implementar as nossas *structs* completando cada uma delas ao longo do programa conforme fosse preciso. A explicação de cada *struct* e elementos das mesmas está comentado no seguinte código:

```
// Struct to contain all the files
struct Filepaths {
    string pathAreas = ""; // String that holds the directory of "areas.txt"
    string pathProviders = ""; // String that holds the directory of "fornecedores.txt"
    string pathProducts = ""; // String that holds the directory of "produtos.txt"
    string pathGardens = ""; // String that holds the directory of "savedPlantation.txt"
    string pathDataNeeded = ""; // String that holds the directory of "savedDataNeeded.txt"
    string pathStorage = ""; // String that holds the directory of "savedStorage.txt"
};

// Struct that holds essential data for the program
struct DataNeeded {
    unsigned short numberofGardens = 0; // Contains the number of gardens on the plantation
    unsigned short sizeofArea = 0; // Contains the size of the array area
    unsigned short sizeofProvider = 0; // Contains the size of the array provideer
    unsigned short sizeofProductname = 0; // Contains the size of the array name
    unsigned short numberofProductsToCreate = 0; // Contains the number os products to create each cycle
    string* areaArray = new string; // Array that will keep all the areas from the file
    string* providerArray = new string; // Array that will keep all providers from the file
    string* productnameArray = new string; // Array that will keep all the names from the file
};
```

```
// Struct of the product
struct Product {
    string name = ""; // Name of the product
    string area = ""; // Product area (equal to the Garden area)
    string provider = ""; // The provider of the product
    float resistance = 0.0; // Resistance of the product
    unsigned short watering = 0; // Watering time of the product
    unsigned short wateringcycle = 0; // Count cycles before each watering
};

// Struct that holds all the Products in a Linked List
struct LLProducts {
    Product data;
    LLProducts* next = NULL;
};

// Struct that holds the Record of all harvested products for each garden in a BST
struct ProductHarvestedRecord {
    Product data;
    ProductHarvestedRecord* left = NULL;
    ProductHarvestedRecord* right = NULL;
};
```

```
// Struct of the garden
struct Garden {
    char gardenorder = ' '; // Order of the Garden
    string owner = ""; // Owner of the Garden
    string area = ""; // Area of the Garden
    unsigned short capacity = 0; // Max number of products the Garden can support
    unsigned short numberproducts = 0; // Number of products in the Garden
    unsigned short quantityharvested = 0; // Number of harvested products of the Garden
    LLProducts* ingarden = NULL; // Products in the Garden
    ProductHarvestedRecord* gardenrecord = NULL; // Harvested Products of the Garden
};

// Struct that holds all the Gardens in a Linked List
struct LLGardens {
    Garden oneGarden;
    LLGardens* next = NULL;
};
```

Inicialização das hortas

Para inicializar as hortas, criámos a função *initializeGardens* que recebe o número de hortas (calculadas aleatoriamente) e a referência à estrutura que vai conter toda a plantação. Cada horta é criada na função *createGarden* e posteriormente é inserida na lista ligada de hortas (*LLGardens*).

Inicialização dos produtos

Para inicializar os produtos, criámos a função *initializeProducts* que um conjunto de dados essenciais e a referência à estrutura que vai conter esses produtos criados. Cada produto é criado na função *createProduct* e inserido na lista ligada de produtos (*LLProducts*).

Funcionamento

1. Para a colheita de produtos, visto que cada produto tinha 25% de chance de ser colhido a cada ciclo, criámos uma função *verifyProductHarvested* que origina um número aleatório entre 0 e 3. Tendo em conta que temos 4 números entre o 0 e o 3, a probabilidade de cada número ser escolhido é de 25% e assim estipulámos que quando o número aleatório fosse o 0, o produto é colhido, ou seja removido da lista ligada de produtos da horta, e é adicionado a uma árvore de pesquisa binária associada à horta. De realçar que sempre que um produto é adicionado à árvore, é aplicado o método DSW para balancear a mesma.

2. Para a rega de produtos foi adicionada um contador decrescente para cada produto na função *plague* que está no ponto 5.

3. Para a criação de 10 novos produtos (15 no início do programa) foi criada a função *initializeProducts*, cada um deles criado aleatoriamente através da função *createProduct* que serão adicionados no armazém.

4. Para a atribuição dos produtos do armazém às respetivas hortas, criámos a função *assignProductsToGardens*. Tendo em conta que o armazém funciona como uma fila, o produto que está há mais tempo no armazém deverá ser o próximo a ser analisado e a ser levado para a horta. Se o produto que esteja a ser analisado for de uma área em que as hortas da área correspondente não possuam mais capacidade, a função passa a analisar o produto seguinte, mantendo o produto anterior no armazém, até que haja espaço na horta dessa área.

5. Na implementação da praga, criámos a função *plague* que determina se, no dia anterior à rega de um produto, este é atacado ou não por uma praga. Para efetuarmos esta verificação, atribuímos à chance de um produto ser atacado por uma praga o "contrário" da sua resistência (se a resistência for 60%, então a chance de ser atacado por uma praga é de 40%). Assim sendo, geramos aleatoriamente um número entre 1 e 100 e se este número for menor ou igual à probabilidade de ser atacado, o produto é então removido, adicionado a uma lista ligada que guarda o registo de todos os produtos atacados por pragas e por fim é mostrado na consola uma mensagem de que o produto foi perdido e a que horta pertencia. Caso contrário, nada acontece ao produto e este recomeça um novo ciclo de rega.

Colheita manual

Para realizar a colheita manual, introduzimos a função *harvestProduct* em que é pedido ao utilizador que escreva o nome do produto que deseja colher. A função vai percorrer todas as hortas e retirar o produto escolhido das mesmas, além de adicioná-lo à lista de produtos colhidos das hortas a que o mesmo pertencia.

Atualizar tempo de rega

Para atualizar o tempo de rega, implementamos a função *updatewatering* em que é pedido ao utilizador que introduza o nome do produto que deseja atualizar tempo de rega. Em seguida pede ao utilizador para introduzir o novo tempo de rega e altera o mesmo em todos os produtos com esse nome, esteja na plantação ou no armazém.

Gravar Plantação

Para gravar plantação, implementamos a função *recordPlantation*. Este processo começa por verificar se o diretório dos ficheiros a salvar é válido através da função *verifyFiles*. Se o diretório for válido então ele salva toda a *essential Data* no ficheiro *savedDataNeeded.txt*, seguido das plantações no ficheiro *savedPlantation.txt* e por fim o armazém no ficheiro *savedStorage.txt*.

Carregar Plantação

Para carregar uma plantação, implementamos a função *loadPlantation*. Este processo começa por verificar se o diretório dos ficheiros a salvar é válido através da função *verifyFiles*. Se o diretório for válido então ele carrega toda a *essential Data* do ficheiro *savedDataNeeded.txt*, seguido das plantações do ficheiro *savedPlantation.txt* e por fim o armazém do ficheiro *savedStorage.txt*.

Imprimir Plantação

Para imprimir a plantação, criamos uma função *printProducts*. Esta função começa por criar uma lista ligada onde coloca todos os produtos da plantação e do armazém. De seguida uma função auxiliar vai ordenar a lista ligada recorrendo a uma espécie de *bubble sort,* posteriormente imprimindo os produtos por ordem alfabética. Além disso a função também imprime os produtos perdidos recorrendo ao mesmo processo acima.

Imprimir produtos colhidos

Para mostrar o registo de colheitas, criamos a função *printHarvestedProducts*. Esta função começa por pedir ao utilizador que introduza a identificação da horta que tenciona verificar o registo de colheitas. Uma vez que tenha o nome do responsável, a função recorre à árvore binária que guarda o registo de produtos colhidos e imprime os nomes dos mesmos através de uma travessia infixa.

Criar nova área

Para criar uma nova área, implementamos a função *createnewarea*. Esta função cria um *array* auxiliar com as áreas do sistema e pede posteriormente ao utilizador para inserir a nova área no qual é adicionada ao *array*. Tendo este *array* completo é igualado o novo *array* ao *array* das áreas do sistema e incrementado o tamanho do *array*.

Alterar área

Para alterar a área de uma horta, concebemos uma função *changeArea*, a mesma irá permitir ao utilizador selecionar uma horta já existente e mudar a sua área. Depois de selecionada a horta e ter-lhe sido atribuída uma nova área, a função irá retirar todos os produtos da dita horta. A retirada de elementos do armazém para a nova horta segue a prioridade já definida no Funcionamento.

Conclusão

Com este trabalho, podemos concluir que é possível criar uma Plantação através do programa Visual Studio.

Este trabalho foi desenvolvido recorrendo a tipos de dados definidos pelos alunos (*structs*), vetores dinâmicos (*arrays*), listas ligadas (linked-lists) e árvores de pesquisa binária (*BST*) dos quais foram otimizados ao máximo obtendo sempre resultados positivos.

Neste projeto, resolvemos aperfeiçoar a usabilidade do mesmo e melhorar certos aspetos referidos pelo docente em relação ao projeto anterior, além de substituir todos os tipos de dados *int* para *unsigned short*, não só para ocupar menos espaço na memória, como também maximizar o alcance dos números que iriamos utilizar. Todo o código aplicado foi aperfeiçoado ao máximo, muitos deles utilizando matéria dada na cadeira.

Por fim, podemos admitir que este projeto foi um êxito pois chegámos ao resultado esperado, cumprindo todos os requisitos propostos.

Divisão das tarefas

Bernardo Coelho

- Realizou as funções:

  1.Imprimir plantação;

  2.Imprimir produtos colhidos;

Nuno Fernandes

- Realizou as funções:

  1.Colheita manual;

  2.Atualizar tempo de rega;

João Gomes

- Realizou as funções:

  1.Carregar plantação;

  2.Gravar plantação;

Guilherme Teixeira

- Realizou as funções:

  1.Criar nova área;

  2.Alterar área;

A idealização das *structs,* inicialização das hortas, produtos, funcionamento e realização do relatório foram feitas por todos os membros do grupo em conjunto.