UNIVERSIDADE da MADEIRA

2021/2022

Faculdade de Ciências Exatas e da Engenharia

Estruturas de Dados e Algoritmos

Projeto 1 - Plantação EDA

Docente: Filipe Quintal

Trabalho realizado por:

Bernardo Coelho - 2082121

Nuno Fernandes - 2082021

Guilherme Teixeira - 2079121

João Gomes - 2081321

Índice

Introdução e Objetivos

Neste projeto pretende-se 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;

Fertilização;

Carregar plantação;

Imprimir plantação (por ordem alfabética e por tempo de plantação)**;**

Criar nova área;

Mostrar registo de colheitas;

Alterar área;

Também foi previamente proibida a utilização de listas ligadas e da classe/biblioteca vetor.

Nota: Todos os avisos deste programa devem ser ignorados!

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 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
    int resistance = 0; // Resistance of the product
    int watering = 0; // Watering time of the product
    int wateringcycle = 0; // Count cycles before each watering
    int timeplanted = 0; // Count the time planted of the product
};

// Struct to hold the Record of all harvested products for each garden
struct ProductHarvestedRecord {
    string name = "";
};
```

```
// Struct that holds essential data for the program
struct DataNeeded {
    int numberofGardens = 0; // Contains the number of gardens on the plantation
    int sizeofArea = 0; // Contains the size of the array area
    int sizeofProvider = 0; // Contains the size of the array provideer
    int sizeofProductname = 0; // Contains the size of the array name
    int sizeofAreasChoosen = 0; // Contains the size of the array that has all the areas being used
    int 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
    string* areasChoosenArray = new string; // Array that will contain only the areas that are being used
};

// Struct of the garden
struct Garden {
    char gardenorder = ' '; // Order of the Garden
    int capacity = 0; // Max number of products the Garden can support
    int numberproducts = 0; // Number of products in the Garden
    Product* ingarden = new Product; // Products in the Garden
    ProductHarvestedRecord* gardenrecord = new ProductHarvestedRecord; // Harvested Products of the Garden
    string owner = ""; // Owner of the Garden
    string area = ""; // Area of the Garden
    int quantityharvested = 0; // Number of harvested products of the Garden
    int cyclesFertilization = 0; // Counter of fertilization cycles
};

// Struct of the Storage
struct Storage {
    Product* inStorage = new Product; // Array that holds all the products in Storage
    int numProducts = 0;  // Number of products in the Storage
};
```

Inicialização das hortas

Para inicializar as hortas, criámos a função *inicializeGardens* que recebe o número de hortas (calculadas aleatoriamente) e a estrutura que vai conter toda a plantação. Cada horta tem um carater identificador, uma capacidade, os produtos plantados, os produtos colhidos, um responsável e uma área. É atribuído um carater diferente a cada horta e a capacidade de cada uma é calculada aleatoriamente (entre 3 e 8), pedindo, posteriormente, ao utilizador para inserir o nome do responsável por cada uma das hortas. Após termos estes dados, é atribuída uma área aleatória a cada horta.

Inicialização dos produtos

Para inicializar os produtos, criámos a função *inicializeProducts* que escolhe aleatoriamente um nome para cada produto, uma área das já escolhidas para as hortas e um fornecedor dos respetivos ficheiros fornecidos. Recebe ainda um número aleatório entre 1 e 5 para a rega do produto e outro número entre 0 e 100 para a resistência. Depois de termos todos estes parâmetros, cada produto é adicionado ao armazém. Quando o programa é inicializado são adicionados 15 produtos ao armazém e sempre que for iniciado um novo ciclo são adicionados mais 10 produtos.

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 e é adicionado ao registo de produtos colhidos.

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 *createNewProducts*, cada um deles criado aleatoriamente através da função *inicializeProducts* que serão adicionados no armazém. No final da função é implementado o *Bubble Sort* para a colocar os produtos de mesma área e mesmo nome juntos mantendo também os restantes produtos ordenados por ordem de chegada. Qualquer outro *Sort* comprometeria a ordem como os produtos estavam colocados!

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, neste caso, *plantationStorage->inStorage[0]* é o elemento que está á mais tempo no armazém. 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, mostrando 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.

Fertilização

Para permitir a fertilização, foi criada a função *fertilization*. Esta função começa por pedir o tempo de fertilização e a área a fertilizar. Após obter estes dados a função *verifyFertilization* irá fertilizar a respetiva área e a resistência dos produtos que lá se encontram aumentará 10% a cada ciclo.

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 pedir ao utilizador que escolha que tipo de local pretende carregar, podendo ser em ficheiros locais ou outros ficheiros. Caso os ficheiros sejam locais, a função executa o carregamento dos ficheiros do diretório. Caso contrário a função solicita que o utilizador introduza os diretórios dos ficheiros necessários e em seguida executa o carregamento dos ficheiros. Em ambos os casos ocorre a verificação da localização dos ficheiros.

Imprimir Plantação

Para imprimir a plantação, criamos uma função *printProducts*. Esta função começa por cria um *array* com o número de produtos que existem na plantação e no armazém e põe todos os produtos no *array*. De seguida, pede ao utilizador que escolha o método de impressão, podendo ser por ordem alfabética ou por tempo de plantação. Uma vez que esta escolha tenha sido feita, a função. Uma vez que o processo anterior esteja completo, uma função auxiliar vai ordenar o *array* recorrendo a um *bubble sort* e logo de seguida imprimir os produtos recorrendo ao método escolhido pelo utilizador.

Criar nova área

Para criar uma nova área, implementamos a função *createnewarea*. Esta função pede ao utilizador que introduza a nova área que pretende criar. Com essa informação, a função cria um novo *array* para as áreas escolhidas para a criação de produtos com uma posição a mais do que o anterior. Assim, a função introduz todas as áreas utilizadas anteriormente no novo *array* e adicionando por último a nova área. Após ter este novo *array* completo, o programa substitui o antigo *array* de áreas pelo novo.

Mostrar registo de colheitas

Para mostrar o registo de colheitas, criamos a função *harvestRecord*. Esta função começa por pedir ao utilizador que introduza o nome do responsável da horta que tenciona verificar o registo de colheitas. Uma vez que tenha o nome do responsável, a função recorre ao registo de produtos colhidos e imprime os nomes dos produtos colhidos pelo responsável escolhido.

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, sendo, por fim preenchida novamente com produtos da mesma área em armazém. A função também tem em conta a chance de não ser possível preencher a horta indicada anteriormente, se não existirem produtos suficientes no armazém. 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*) e a vetores dinâmicos (*arrays*), dos quais foram otimizados ao máximo obtendo sempre resultados positivos.

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

Divisão das tarefas

Bernardo Francisco Costa Coelho

- Realizou as funções:
  1. Colheita manual;
  2. Fertilização;
  3. Criar nova área;

Nuno Oliveira Fernandes

- Realizou as funções:
  1. Funcionamento;
  2. Gravar da plantação;
  3. Carregar plantação;

João Pedro Sousa Gomes

- Realizou as funções:
  1. Inicialização das hortas;
  2. Atualizar tempo de rega;
  3. Imprimir plantação;

João Guilherme Rodrigues Teixeira

- Realizou as funções:
  1. Inicialização dos produtos;
  2. Mostrar registo de colheitas;
  3. Alterar área;

A idealização das *structs* e a realização do relatório foi feita por todos os membros do grupo.