UNIVERSIDADE da MADEIRA

2022/2023

Faculdade de Ciências Exatas e da Engenharia

Licenciatura em Engenharia Informática (LEI)

Estruturas de Dados e Algoritmos

Docente Regente – Filipe Magno de Gouveia Quintal

Projeto 1 – "OficinaEDA"

Trabalho elaborado por:
João Castro Nº2079521
Manuel Gonçalves Nº2078621
Sara Rodrigues Nº2118822

Índice

Introdução e Objetivos

Neste projeto pretende-se desenvolver um programa em linguagem de programação C++, que simule o funcionamento de uma oficina (“OficinaEDA”).

O projeto tem como objetivo implementar as seguintes funcionalidades:

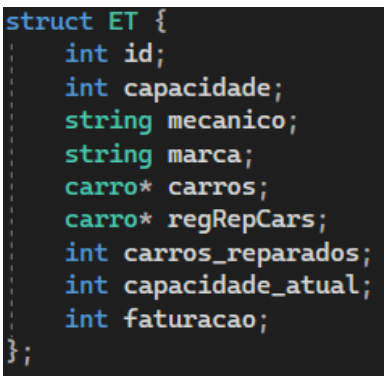
- Inicialização das Estações
- Inicialização dos Carros
- Funcionamento
- Reparação Manual
- Atualizar Tempo de Reparação
- Adicionar Prioridade
- Remover Mecânico
- Gravar Oficina
- Carregar Oficina
- Imprimir Oficina (por ordem alfabética e por ordem de tempo de reparação)

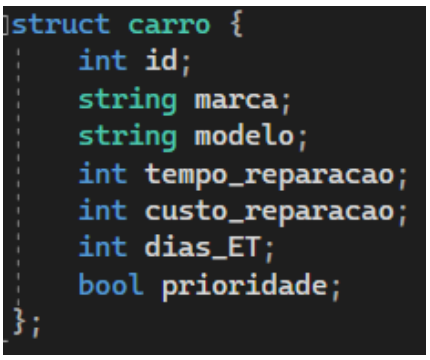
Para tal foi utilizado o IDE Visual Studio 2022 sob a forma de C++, sendo proibida a utilização da utilização da classe/biblioteca vector, de listas ligadas e das classes/bibliotecas Stack e Queue.

Implementação de funcionalidades

Structs

Para este projeto, foram criadas duas structs no ficheiro structs.h, que serviram como base do nosso trabalho, sendo que uma diz respeito aos carros e outra às estações (ET).





Inicialização das Estações

A função inicializarEstacoes é, como o próprio nome diz, responsável por inicializar as estações da oficina. Como cada oficina tem que ter atribuída uma marca de carro, começámos por guardar no array “marcas” todas as marcas de carros presentes no ficheiro marcas.txt.

Em seguida, tendo em conta o número de estações presentes na oficina é adicionado sequencialmente as seguintes informações para cada uma: um mecânico responsável pela ET, a ID da estação, a capacidade total da mesma (varia aleatoriamente entre 2 e 5), a capacidade atual (que inicialmente corresponde a 0), a marca cuja estação será responsável (retirada aleatoriamente do array marcas inicialmente criado), o valor total de faturação e o número de carros reparados (ambos inicializados a 0).

Temos ainda a inicialização de dois arrays, o primeiro “carros” é o array de carros de cada estação, a dimensão deste corresponde à capacidade da estação e o segundo array “regRepCars” é responsável por guardar os veículos que foram reparados por estação.

Inicialização dos Carros

Foi criada a função criarCarros para efetuar a inicialização dos veículos a entrar na oficina. Para isso, começámos por guardar no array “modelos” todos os modelos de carros presentes no ficheiro modelos.txt.

Depois, e seguindo a devida ordem para o índice, são adicionados ao array “carrosCriados” 10 novos carros, sendo que os atributos de cada carro são definidos como indicado no enunciado. O ID é incrementado cada vez que um carro é criado, sendo que é impossível haver dois carros com o mesmo ID. A variável NUM CARROS CRIADOS mantém o registo do número total de carros criados, sendo que é incrementada em 10 cada vez que a função é chamada.

Por fim é utilizado um sort, em conjunto com a função comparaCarros que coloca os carros com prioridade no início do array.

Funcionamento

O funcionamento em ciclos é assegurado pela função simulateDay, que cada vez que o utilizador inserir “s” + enter na consola, irá ser simulado um dia na Oficina.

Dado isto, a primeira função desenvolvida foi a reparar_carros. Esta função irá percorrer todas as estações, e primeiramente vai verificar se existe algum carro cujos dias na ET sejam iguais ao tempo máximo de reparação. Caso isto se confirme, o carro é adicionado ao registo de carros reparados da respectiva estação (regRepCar), e o número de carros reparados da ET é incrementado.

Caso o número de dias na ET seja inferior ao tempo de reparação, a variável probabilidade vai determinar se um carro deverá ser reparado ou não. Se o carro for reparado, é adicionado ao regRepCar e a faturação da ET é incrementada.

Feito isto, é criado um array dinâmico, para cada ET, que vai conter todos os carros da ET exceto aqueles que foram reparados. Depois, o array antigo dos carros da ET é apagado e substituído pelo novo.

A cada ciclo, são também criados 10 novos carros utilizando a função descrita no tópico acima.

Para serem adicionados 8 carros às ETs a cada ciclo, é utilizado a função adicionarCarrosETs. Esta função irá percorrer todos os carros na lista de espera, e caso haja estação com capacidade para alojar o carro, o carro é adicionado à ET e o seu ID é adicionado ao array car_ids, que mantém registo de todos os carros que já passaram pelas oficinas. Caso o número de carros criados ultrapasse os 8, o resto mantém-se na lista de espera.

Foi também criada uma função incrementaDiasET que a cada ciclo irá incrementar em uma unidade o atributo dias_ET de cada carro na estação.

Reparação Manual

Na reparação manual é pedido ao utilizador as características do veículo (marca e modelo) que pretende remover da estação. Quando encontrado, é adicionado ao array de carros reparados, a função informa ao utilizador o preço de reparação e adiciona-o ao valor de faturação da estação e finalmente é eliminado da oficina. Entretanto, como pode haver mais do que um veículo com características iguais, a função só conclui a procura quando a estação, correspondente à marca do carro, for analisada por completo.

Finalmente é passado ao utilizador a informação de quantos veículos foram reparados manualmente.

Atualizar Tempo de Reparação

Esta função pede ao utilizador um novo tempo de reparação para atribuir a um ou mais veículos presentes na fila de espera. Quando encontrados os veículos o seu valor é alterado, caso contrário a função informa ao utilizador que não existe nenhum carro na lista de espera com as características pedidas.

Adicionar Prioridade

Para adicionar prioridade a um veículo é necessário o utilizador inserir o seu ID. Esta função só adiciona a prioridade a um veículo que esteja na fila de espera e que não seja prioritário. Caso contrário, a função não fará qualquer alteração.

Remover Mecânico

Aqui, é pedido ao utilizador o nome do mecânico que pretende remover da oficina. Caso o mecânico exista, este é removido. Logo depois é pedido ao utilizador que insira o nome do novo mecânico e é gerada aleatoriamente uma nova marca para a ET. Caso o mecânico removido fosse especialista numa marca única na ET, os seus carros nunca mais serão reparados e permaneceriam na lista de espera. Adicionalmente, poderão ser criados novos carros da nova marca do novo mecânico, sendo que esta foi adicionada ao array marcas_ET que contém todas as marcas das estações e que é usado na função criaCarros.

Gravar Oficina

Esta funcionalidade tem como objetivo gravar em ficheiros .txt todos os dados da oficina. Para tal foram criadas funções, de modo a gravar todos os dados dos carros criados, dos carros da lista de espera e dos carros das estações (presentes nelas e reparados). Foi preciso então, nas diversas funções de gravação de ficheiros, buscar todos os dados presentes nos arrays e o número de carros em cada array.

Carregar Oficina

Para carregar a oficina, todos os arrays e variáveis presentes no ficheiro oficina.cpp são igualados aos arrays carregados dos ficheiros. Este processo permite ao utilizador escolher de que local pretende carregar os dados para a oficina. O utilizador pode introduzir o diretório dos quatro ficheiros, ou, caso não o faça, o carregamento da oficina será feito através dos ficheiros locais. É feita a verificação da introdução dos caminhos dos ficheiros no main.

Imprimir Oficina

Para esta função, são criados dois arrays dinâmicos que vão conter todos os carros da ET, e da lista de espera, respetivamente. Por fim, é pedido ao utilizador o método de ordenação destes, e, depois de percorridos os ciclos de ordenação, são imprimidos na consola ambos os arrays, permitindo ao utilizador perceber que carros estão nas ET's e na lista de espera num determinado momento.

Conclusão

Com este trabalho, podemos concluir que é possível implementar diversas funcionalidades de uma oficina em linguagem C++ e através do IDE Visual Studio

2022. Este trabalho foi desenvolvido recorrendo a structs e a vetores dinâmicos (arrays).

Por fim, chegamos ao trabalho finalizado com todas as funcionalidades propostas.

Divisão das tarefas

João Vítor Afonso de Castro:

- Remover Mecânico
- Gravar Oficina
- Carregar Oficina
- Imprimir Oficina (por ordem de tempo de reparação)

Manuel Luís de Sousa Gonçalves:

- Inicialização dos Carros
- Funcionamento
- Remover Mecânico
- Imprimir Oficina (por ordem alfabética)

Sara Margarida Sousa Rodrigues:

- Inicialização das Estações
- Reparação Manual
- Atualizar Tempo de Reparação
- Adicionar Prioridade