|  |  |
| --- | --- |
| Faculdade de Tecnologia de Ribeirão Preto | |
| Autonomous Parking System  Douglas Samuel Gonçalves: 2840482013025  Algoritmos e Lógica de Programação  Análise e Desenvolvimento de Sistemas  12/07/2020 | Uma imagem contendo desenho, placa  Descrição gerada automaticamente | |
| Uma imagem contendo desenho  Descrição gerada automaticamente | |
| **Relatório Referente ao Projeto Semestral** | | |

## Descrição do Problema

Foi solicitado que fosse criado um programa para um estacionamento que possuí 45 vagas para carros padrão, 60 vagas para camionetes e 15 vagas para carros SUPER. O estacionamento funciona de forma autônoma então sempre que o motorista chegar, é necessário verificar o tipo do carro e alocar ele a vaga correta (se tiver vaga disponível), obtendo a data e hora de entrada que será posteriormente usada para cálculo do valor a ser pago.

Quando o motorista voltar, deve-se gerar um ticket com a data de entrada, data de saída, valor total a ser pago assim como a vaga que o carro ficou e o tipo (carro padrão, super ou camionete) do mesmo.

O valor que deve ser usado para cálculo é a tabela: padrão R$5,00 a hora, camionete R$7,00 a hora e super R$10,00 a hora. O sistema deve ainda contabilizar a movimentação para exibir futuramente relatórios como: quantidade de carros atendidos, quantidade de camionetes atendidas, quantidade de veículos não atendidos por falta de vagas(lotação), faturamento médio por hora e faturamento total.

## Discussão

* Descrição da solução/implementação: O programa utiliza a função main como centralizadora do programa e organizando o que será feito com o input do usuário, nela está armazenada as matrizes que guardam as vagas do estacionamento para cada tipo assim como os tickets anteriores(limitados a 1000 tickets, sendo uma matriz fixa) e quantidade de veículos não atendidos por lotação. A função main exibe a tabela de preço para o usuário e lhe dá 3 opções, estacionar o carro (1), retirar o carro do estacionamento (2) e emitir relatórios com base em um período (3).
* Caso seja selecionado a opção 1 a função estacionaCarro é chamada, é passada como parâmetro para ela as 3 matrizes com as vagas do estacionamento por tipo e a variável inteira que carrega a quantidade de carros não atendidos por lotação, todas por referência. A função pede ao usuário que seja preenchido qual o tipo do carro e com base na resposta verifica na matriz correspondente se existe uma vaga(utilizando um for), caso seja encontrada uma vaga é questiono ao usuário a placa do veículo(que deve conter 7 caracteres e preferencialmente ser maiúscula para na hora de retirar o carro ficar mais fácil de encontrar), tendo uma entrada válida o sistema solicita que seja inserida a data de entrada e salva o ticket na matriz e exibe as informações que foram salvas. Caso não seja encontrada vaga, é emitido um aviso ao usuário sobre a lotação e adicionado 1 a variável que carrega a quantidade de veículos não atendidos por lotação.
* Caso seja selecionada a opção 2, a função retiraCarro é chamada com os parâmetros por referência: 3 matrizes que diz respeito as vagas por tipo do estacionamento e a matriz que contém todos os tickets anteriores já fechados. A função pede ao usuário que seja inserido o tipo do carro para saber qual matriz consultar para buscar o carro e a placa dele, com base na resposta percorre a matriz com um for até encontrar qual vaga está o carro. Ao encontrar o carro ele demonstra a vaga que o carro está e solicita que seja inserida a data de retirada do carro, assim o sistema chama a função calculaDifDatas que traz a diferença em segundos da data de entrada do carro e data de saída do carro utilizando a biblioteca ctime após isso salva o ticket na matriz tickets(que armazena todos os tickets já finalizados) e limpa a “vaga” na matriz correspondente ao tipo do carro exibindo as informações do ticket como: placa, tipo do carro, datas de entrada/saída, valor a ser pago, quantidade de horas e valor a hora. Caso não seja encontrada a placa do carro ele pede ao usuário que verifique a placa e exibe a mesma na tela.
* Caso seja selecionada a opção 3, a função emiteRelatórios é chamada, para ela é passada por referência duas variáveis, a matriz com os tickets fechados para consulta assim como os carros não atendidos por lotação. É solicitado ao usuário que insira o intervalo de datas em que o programa irá apurar os valores e exibir no relatório. É exibido os relatórios possíveis para ser executado, sendo 1-Para emitir a quantidade de veículos atendidos com divisões por tipo, 2-Para emitir a quantidade de veículos não atendidos por lotação, 3-Para emitir o faturamento médio por hora e 4-Para emitir o faturamento total. Para qualquer uma das entradas, o sistema percorre a matriz tickets e retira as informações tomando como base o período inserido, verificando se o ticket está no mesmo utilizando um if.
* Maiores dificuldades: Tive dificuldades em passar os parâmetros por referência, nunca havia utilizado antes dessa forma por achar arriscado, assim como trabalhar com o a biblioteca ctime para buscar a diferença de horas entre as datas, no início eu fiz um sistema que tomava como base o ano 2000 e buscava a diferença em segundos da data inserida até ele porém aumentou consideravelmente a quantidade de linhas do programa então achei melhor usufruir da biblioteca.
* Com questão as matrizes, poderia ser utilizada uma matriz com mais de uma dimensão para armazenar cada tipo, porém o código fica mais limpo utilizando 3 matrizes diferentes para cada tipo. Com o restante não tive dificuldades pois foi bem aplicado durante as listas.

## Bugs e Problemas Identificados

O programa trata a maioria das entradas, irei citar como são tratadas as entradas e quais não são com base na ordem de execução do programa (Estaciona o carro, retira o carro e emite relatório).

Caso na main seja inserido um valor fora do intervalo 0 a 3, ele pede ao usuário que insira novamente o valor até que esteja no intervalo, porém se for inserida uma letra o sistema automaticamente pula para a linha de finalização, uma exceção não tratada.

Função estacionaCarro, caso seja inserido um valor fora do intervalo 1 a 3 ele pede para que o usuário insira novamente porém ele quebra a execução caso seja uma letra visto que a variável tipo carrega apenas inteiros, a entrada da placa também é questionada novamente caso não tenha 7 caracteres mas o sistema não torna ela *uppercase* o que pode ser um problema para buscar a placa depois no sistema. As datas também são verificadas, se os dias estão no intervalo 1 a 31, meses 1 a 12 e anos não podem ser negativos.

Função retiraCarro faz as mesmas verificações que a função estacionaCarro com a diferença que a função também verifica a data de retirada do carro para verificar se ela é após a data de entrada. Todas as datas não tratam letras pois são variáveis inteiras.

Função emiteRelatórios, apenas verifica as datas inseridas como início e fim do período para garantir que o mês esteja entre 1 e 12 e o ano seja positivo.

## Lições Aprendidas

* Com exceção do tratamento de datas (que não consegui fazer sem custar 300 linhas de código), o restante fluiu bem, consegui criar uma linha de raciocínio e seguir ela com a programação.
* A tratativa de erros e datas, gostaria de implementar o sistema porém como uma biblioteca para o código não parecer tão poluído, a tratativa de erros, pretendo procurar formas para melhorar as entradas do usuário, tem questões como entrada de letras em variáveis inteiras que apenas param o programa e seria interessante apenas exibir o erro.
* O uso de struct, funções com passagem de parâmetro por referência e estruturas de repetição.
* Creio que apenas a tratativa de erros, o restante da matéria no geral eu consegui assimilar com facilidade.