PROJETO INTEGRADO INOVAÇÃO - ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

JOSE EUDES DE SOUZA SILVA

Análise e Desenvolvimento de Sistemas – EAD

São Paulo

2024

PROJETO INTEGRADO INOVAÇÃO - ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

Jose Eudes de Souza Silva

Instituição: Anhanguera

Curso: Analise e Desenvolvimento de Sistemas – EAD

São Paulo, 2024

ATIVIDADE PROPOSTA:

Cenário:

• O programa será usado por um sistema de cálculo que exige verificações matemáticas simples e validações lógicas com base nos números fornecidos.

• O foco é compreender como utilizar operadores aritméticos para calcular resultados e operadores relacionais e lógicos para validar condições.

Contexto:

Você foi contratado por uma empresa de comércio eletrônico que está em rápido crescimento. A empresa enfrenta dificuldades para gerenciar seu estoque de forma eficiente, o que tem levado a problemas como falta de produtos, excesso de estoque e dificuldade em rastrear a localização dos itens nos depósitos. A empresa deseja desenvolver um sistema de gerenciamento de estoque que otimize suas operações, melhorando a eficiência e a precisão na gestão de produtos.

Problema:

Desenvolver um sistema de gerenciamento de estoque para a empresa de comércio eletrônico que atenda aos seguintes requisitos:

• Cadastro de Produtos: O sistema deve permitir o cadastro de novos produtos, incluindo informações como nome, categoria, quantidade em estoque, preço e localização no depósito.

• Atualização de Estoque: O sistema deve permitir a atualização da quantidade de produtos em estoque quando novos itens são recebidos ou quando produtos são vendidos.

• Rastreamento de Localização: O sistema deve possibilitar o rastreamento da localização dos produtos dentro dos depósitos para facilitar a logística.

• Relatórios: O sistema deve gerar relatórios sobre o estado do estoque, destacando produtos com estoque baixo, excesso de estoque e movimentação de produtos.

• Tabela Verdade: Deve-se criar uma tabela verdade para definir as combinações de valores (True e False) para as variáveis booleanas da situação problema.

• Diagrama de Caso de Uso: Crie um Diagrama de Casos de Uso para o sistema de gerenciamento de estoque da empresa.

SOLUÇÃO:

Link Trello:

https://trello.com/b/kDMw5vjR/desenvolvimento-do-sistema-de-gerenciamento-de-estoque-eudes

Tabela Verdade:

P (Cadastro)	E (Estoque)	L (Localização)	R (Relatórios)	Descrição
FALSE	FALSE	FALSE	FALSE	Sem funcionalidades
FALSE	FALSE	FALSE	TRUE	Apenas relatórios
FALSE	FALSE	TRUE	FALSE	Apenas localização
FALSE	FALSE	TRUE	TRUE	Localização e relatórios
FALSE	TRUE	FALSE	FALSE	Apenas atualização de estoque
FALSE	TRUE	FALSE	TRUE	Estoque e relatórios
FALSE	TRUE	TRUE	FALSE	Estoque e localização
FALSE	TRUE	TRUE	TRUE	Estoque, localização e relatórios
TRUE	FALSE	FALSE	FALSE	Apenas cadastro
TRUE	FALSE	FALSE	TRUE	Cadastro e relatórios
TRUE	FALSE	TRUE	FALSE	Cadastro e Localização
TRUE	FALSE	TRUE	TRUE	Cadastro, localização e relatórios
TRUE	TRUE	FALSE	FALSE	Cadastro e atualização de estoque
TRUE	TRUE	FALSE	TRUE	Cadastro, estoque e relatórios
TRUE	TRUE	TRUE	FALSE	Cadastro, estoque e localização
TRUE	TRUE	TRUE	TRUE	Sistema completo

Codificação:

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

void DesenharLinha();
void InitProdutos();

void MainMenu();
void TrocarMenu(int);
void MenuEspera(int);

void CadastrarProduto();
void AtualizarEstoque();
void LocalizarProduto();
```

```c
void GerarRelatório();
void AtualizarHistorico();
void UltimasMovimentacoes();
void MenorMaiorQuantidade();

void MostrarTodos();
void InfoProduto(int);

struct Local{
    int Linha;
    int Coluna;
    int Andar;
};
struct Produto{
    int ID;
    char Nome[27];
    int Categoria;
    int QuantidadeEstoque;
    float Preco;
    struct Local Localizacao;
};
struct Historico{
    int IDMovimentado;
    int quantidadeAlterada;
};

#define MaxProdutos 108
struct Produto Produtos[MaxProdutos];
struct Historico History[MaxProdutos];
int IDMovimentacao;

int main(){
    InitProdutos();

    MostrarTodos();
    MainMenu();

    return 0;
}
```

```c
//Helper
void DesenharLinha(){
    for(int i = 0; i < 54; i++){
        printf("*");
        if(i == 53) printf("\n");
    };
}

void InitProdutos(){
    IDMovimentacao = 0;
    for(int i = 0; i < MaxProdutos/4; i++){
        History[i].IDMovimentado = 0;
        History[i].quantidadeAlterada = 0;

        char nome[27];

        int magicNumber = i%2;
        int randNum = rand() % 3;
        int randQuant = rand() % 72;

        int randLinha = rand() % 11;
        int randColuna = rand() % 9;
        int randAndar = rand() % 10;

        Produtos[i].ID = i+1;

        sprintf(nome, "produto%d", Produtos[i].ID);
        strcpy(Produtos[i].Nome, nome);

        Produtos[i].Categoria = (magicNumber + 1); //1 or
2

        Produtos[i].QuantidadeEstoque = randQuant;

        Produtos[i].Preco = i + ( 27 * 0.09 );

        Produtos[i].Localizacao.Linha = (randLinha);
        Produtos[i].Localizacao.Coluna = (randColuna);
```

```
        Produtos[i].Localizacao.Andar = (randAndar);

        AtualizarHistorico(Produtos[i].ID,
Produtos[i].QuantidadeEstoque);
    }
}

void MostrarTodos(){
    for(int i = 0; i < MaxProdutos; i++){
        if(Produtos[i].ID == 0){
            break;
        }else{
            InfoProduto(i + 1);
        }
    }
}

void InfoProduto(int id){
    id -= 1;
    printf("ID : %d\n", Produtos[id].ID);
    printf("Nome : %s\n", Produtos[id].Nome);
    printf("Categoria : %d\n", Produtos[id].Categoria);
    printf("QuantidadeEstoque : %d\n",
Produtos[id].QuantidadeEstoque);
    printf("Preco : %f\n", Produtos[id].Preco);
    printf("Linha %d, Coluna %d, Andar %d\n",
Produtos[id].Localizacao.Linha,
Produtos[id].Localizacao.Coluna,
Produtos[id].Localizacao.Andar);
    printf("\n");
}


//Menus
void MainMenu(){
    int escolha;
    DesenharLinha();
    printf("* ***** Gerestoque ****** *\n");
    printf("\n");
```

```
    printf("*  1 - Cadastrar Produto  *\n");
    printf("*  2 - Atualizar Estoque  *\n");
    printf("*  3 - Localizar Produto  *\n");
    printf("*  4 - Gerar Relatorios   *\n");
    printf("*  Ctrl + C - Sair        *\n");
    DesenharLinha();

    printf("\n");
    printf("Digite a opção desejada\n");
    scanf("%d", &escolha);

    TrocarMenu(escolha);
}

void TrocarMenu(int menuID){
    switch (menuID){
        case 0:
            MainMenu();
        break;

        case 1:
            CadastrarProduto();
        break;

        case 2:
            AtualizarEstoque();
        break;

        case 3:
            LocalizarProduto();
        break;

        case 4:
            GerarRelatório();
        break;
    }
}

void MenuEspera(int menuID){
```

```
    int escolha;

    switch (menuID){
        case 1:
            printf("* 1 para Cadastrar um novo
produto  *\n");
        break;

        case 2:
            printf("* 2 para atualizar mais um
produto  *\n");
        break;

        case 3:
            printf("* 3 para buscar novamente *\n");
        break;

        default:
        break;
    }

    printf("* 0 - para retornar ao menu principal *\n");
    printf("* Ctrl+C para sair                    *\n");
    scanf("%d", &escolha);

    if(escolha == menuID) { TrocarMenu(menuID); }
    if(escolha == 0){ TrocarMenu(0); }
}


//Metodos funcionais
void CadastrarProduto(){
    int prodID;
    DesenharLinha();

    for(int i = 0; i < MaxProdutos; i++){
        if(Produtos[i].ID != 0){
           continue;
        }else{
```

```
            prodID = i;
            Produtos[i].ID = prodID +1;
            break;
        }
    }

    //Nome
    char prodNome[28];
    printf("\n");
    printf("Qual o nome?\n");
    scanf("%27s", prodNome);
    strcpy(Produtos[prodID].Nome, prodNome);

    //Categoria
    int prodCat;
    printf("\n");
    printf("Qual a categoria?\n");
    printf("Categoria 1\n");
    printf("Categoria 2\n");
    printf("Categoria 3\n");
    printf("\n");
    scanf("%d", &prodCat);

    Produtos[prodID].Categoria = prodCat;

    //Quantidade
    int prodQuant;
    printf("\n");
    printf("Qual a quantidade\n");
    scanf("%d", &prodQuant);

    Produtos[prodID].QuantidadeEstoque = prodQuant;

    //Preco
    float prodPreco;
    printf("\n");
    printf("Qual o Preco?\n");
    scanf("%f", &prodPreco);
```

```
    Produtos[prodID].Preco = prodPreco;

    //Local
    int randLinha = rand() % 11;
    int randColuna = rand() % 9;
    int randAndar = rand() % 10;

    Produtos[prodID].Localizacao.Linha = randLinha;
    Produtos[prodID].Localizacao.Coluna = randColuna;
    Produtos[prodID].Localizacao.Andar = randAndar;

    AtualizarEstoque(Produtos[prodID].ID,
Produtos[prodID].QuantidadeEstoque);

    printf("\n");
    printf("Produto Cadastrado com sucesso\n");
    printf("ID: %d\n", Produtos[prodID - 1].ID);
    printf("Categoria: %d\n", Produtos[prodID -
1].Categoria);
    printf("Quantidade em Estoque: %d\n", Produtos[prodID
- 1].QuantidadeEstoque);
    printf("Preco: %.2f\n", Produtos[prodID - 1].Preco);
    printf("Localizacao: Linha: %d, Coluna: %d, Andar:
%d\n", Produtos[prodID - 1].Localizacao.Linha,
Produtos[prodID - 1].Localizacao.Coluna, Produtos[prodID
- 1].Localizacao.Andar);

    printf("\n");
    MostrarTodos();

    MenuEspera(1);
}

void LocalizarProduto(){
    int id;

    DesenharLinha();
    printf("Qual o ID do produto?");
```

```
    printf("\n\n");

    printf("* 0 - para retonrar\n\n");
    DesenharLinha();
    scanf("%d", &id);

    if(id != 0){
        for(int i = 0; i < 99; i++){
            printf("ID: %d, Quantidade: %d\n", i,
Produtos[i].QuantidadeEstoque);
            if(Produtos[i].ID == id){

                int escolha;
                printf("\n");
                InfoProduto(i);

                printf("\n");
                DesenharLinha();

                MenuEspera(3);

                break;
            }
        }
    }else{
        TrocarMenu(0);
    }
}

void AtualizarEstoque(){
    int id;
    printf("\nQual o ID do produto?");
    scanf("%d", &id);

    int quant;
    printf("\nQual a quantidade a ser adicionada?");
    scanf("%d", &quant);

    Produtos[id - 1].QuantidadeEstoque += quant;
```

```
    AtualizarHistorico(Produtos[id - 1].ID, quant);

    printf("\nNova quantidade em estoque: %d\n",
Produtos[id - 1].QuantidadeEstoque);
    InfoProduto(id);

    MenuEspera(2);
}

void GerarRelatório(){
    MenorMaiorQuantidade();
    UltimasMovimentacoes();

    MenuEspera(4);
}

void MenorMaiorQuantidade(){
    int menorQuant = -1;
    int menorQuantID = -1;

    int maiorQuant = -1;
    int maiorQuantID = -1;

    for(int i = 0; i < MaxProdutos; i++){
        if(Produtos[i].ID != 0){
            if(menorQuant == -1 || maiorQuant == -1){
                menorQuant =
Produtos[i].QuantidadeEstoque;
                maiorQuant =
Produtos[i].QuantidadeEstoque;
                menorQuantID = Produtos[i].ID;
                maiorQuantID = Produtos[i].ID;
            }

            if(Produtos[i].QuantidadeEstoque <
menorQuant){
                menorQuant =
Produtos[i].QuantidadeEstoque;
```

```
                menorQuantID = Produtos[i].ID;
            }

            if(Produtos[i].QuantidadeEstoque >
maiorQuant){
                maiorQuant =
Produtos[i].QuantidadeEstoque;
                maiorQuantID = Produtos[i].ID;
            }
        }
    }

    printf("\n*** Menor Quantidade ***\n");
    InfoProduto(menorQuantID);

    printf("\n*** Maior Quantidade ***\n");
    InfoProduto(maiorQuantID);
}

void UltimasMovimentacoes(){
    DesenharLinha();
    printf("***** Ultimas Movimentacoes *****\n");
    DesenharLinha();

    for (int i = 0; i < MaxProdutos; i++){
        if(History[i].IDMovimentado != 0){
            InfoProduto(History[i].IDMovimentado);
            printf("Quantidade Alterada %d",
History[i].quantidadeAlterada);

            printf("\n");
            DesenharLinha();
        }
    }
}

void AtualizarHistorico(int idMovimentado, int
qtdAlterada){
```

```
    History[IDMovimentacao].IDMovimentado =
idMovimentado;
    History[IDMovimentacao].quantidadeAlterada =
qtdAlterada;
    IDMovimentacao++;
}
```

Diagrama de Caso de Uso:

Crie um Diagrama de Casos de Uso para o sistema de gerenciamento de estoque da empresa, para isso este sistema deve incluir funcionalidades que permitam a interação de três tipos de usuários: Estoquista, Usuário e Gerente de Setor. O Estoquista deve ser capaz de registrar a entrada de produtos, o que inclui a validação da nota fiscal de entrada de produto. O Usuário precisa ter a

capacidade de emitir relatórios de posição semanal e solicitar a compra de produtos, ambos os quais necessitam do processo de consolidação de compras. O Gerente de Setor, por sua vez, deve ter a responsabilidade de autorizar a compra de produtos.