//LINGUAGEM C: LISTA 10

//1

#include <stdio.h>

#include <stdlib.h>

typedef struct Eloja{

char nome[30];

long int pecas;

float valor;

} Eloja;

long int conta\_pecas(Eloja \*p, int N); /\* protÃ³tipo funÃ§Ã£o que conta total de peÃ§as \*/

float conta\_valor(Eloja \*p, int N); /\* protÃ³tipo funÃ§Ã£o que conta valor total vendido \*/

void maior\_venda(Eloja \*p, int N); /\* protÃ³tipo funÃ§Ã£o que determina loja que vendeu mais \*/

int main() {

int i, N;

long int total\_pecas;

float valor\_total;

Eloja \*loja; /\* ponteiro para as lojas \*/

/\* Limpa a tela \*/

system("clear");

/\* Entrada de dados \*/

printf("\n\*\*\*\* Conta a quant. de peÃ§as vendidas e determina a loja que vendeu mais \*\*\*\*");

printf("\n\nCadastro - Entre com a quantidade de lojas: ");

scanf("%d",&N);

getchar();

/\* Aloca memÃ³ria \*/

loja = (Eloja \*) malloc( N\*sizeof(Eloja) );

if( loja == NULL ) /\* testa se a memÃ³ria foi alocada \*/

{

printf("ERRO - MemÃ³ria insuficiente");

exit(1);

}

/\* Entrada dos nÃºmeros \*/

for(i=0; i < N; i++)

{

printf("\nLoja %d: ", i+1);

printf("\nNome : ");

gets(loja[i].nome);

printf("No. peÃ§as : ");

scanf("%li",&loja[i].pecas);

printf("Valor das vendas: R$ ");

scanf("%f",&loja[i].valor);

getchar();

}

printf("\n\*\*\*\*\* Resultados \*\*\*\*\*\n");

/\* Conta a quantidade total de peÃ§as vendidas \*/

total\_pecas = conta\_pecas(loja,N);

printf("\nQuantidade total de peÃ§as vendidas: %li", total\_pecas);

/\* Conta a quantidade total de peÃ§as vendidas \*/

valor\_total = conta\_valor(loja,N);

printf("\nValor total de peÃ§as vendidas : R$ %.2f", valor\_total);

/\* Determina a loja que vendeu mais peÃ§as \*/

maior\_venda(loja,N);

/\* Libera memÃ³ria \*/

free(loja);

printf("\n\n");

return 0;

}

/\* FunÃ§Ã£o que calcula a quantidade total de peÃ§as vendidas \*/

long int conta\_pecas(Eloja \*p, int N)

{

/\* p - ponteiro para estrutura loja \*/

/\* N - quantidade de lojas \*/

int i;

long int cont;

cont = 0;

for(i=0; i<N; i++)

{

cont = cont + p[i].pecas;

}

return cont;

}

/\* FunÃ§Ã£o que calcula o valor total de peÃ§as vendidas \*/

float conta\_valor(Eloja \*p, int N)

{

/\* p - ponteiro para estrutura loja \*/

/\* N - quantidade de lojas \*/

int i;

float cont;

cont = 0.0;

for(i=0; i<N; i++)

{

cont = cont + p[i].valor;

}

return cont;

}

/\* FunÃ§Ã£o que determina que loja vendeu mais \*/

void maior\_venda(Eloja \*p, int N)

{

/\* p - ponteiro para as lojas \*/

/\* N - quantidade de lojas \*/

int i;

Eloja loja\_aux; /\* cria variÃ¡vel auxiliar do tipo Eloja \*/

/\* Busca da loja com maior nÃºmero de peÃ§as vendidas \*/

loja\_aux = p[0];

for(i=1; i < N; i++)

{

if( loja\_aux.pecas < p[i].pecas )

{

loja\_aux = p[i]; /\* se a quantidade de peÃ§as de loja\_aux for menor do a quantidade de peÃ§as da loja p[i], troca todo o conteÃºdo de loja\_aux pelo conteÃºdo da loja p[i] \*/

}

}

/\* Imprime resultados \*/

printf("\n\nLoja que vendeu mais: ");

printf("\nNome : ");

puts(loja\_aux.nome);

printf("No. peÃ§as : %li",loja\_aux.pecas);

printf("\nValor das vendas: %.2f",loja\_aux.valor);

}

//2

#include <stdio.h>

#include <stdlib.h>

typedef struct Eloja{

char nome[30];

long int pecas;

float valor;

} Eloja;

long int conta\_pecas(Eloja \*p, int N); /\* protÃ³tipo funÃ§Ã£o que conta total de peÃ§as \*/

float conta\_valor(Eloja \*p, int N); /\* protÃ³tipo funÃ§Ã£o que conta valor total vendido \*/

void maior\_venda(Eloja \*p, int N); /\* protÃ³tipo funÃ§Ã£o que determina loja que vendeu mais \*/

int main() {

int i, N;

long int total\_pecas;

float valor\_total;

Eloja \*loja; /\* ponteiro para as lojas \*/

/\* Limpa a tela \*/

system("clear");

/\* Entrada de dados \*/

printf("\n\*\*\*\* Conta a quant. de peÃ§as vendidas e determina a loja que vendeu mais \*\*\*\*");

printf("\n\nCadastro - Entre com a quantidade de lojas: ");

scanf("%d",&N);

getchar();

/\* Aloca memÃ³ria \*/

loja = (Eloja \*) malloc( N\*sizeof(Eloja) );

if( loja == NULL ) /\* testa se a memÃ³ria foi alocada \*/

{

printf("ERRO - MemÃ³ria insuficiente");

exit(1);

}

/\* Entrada dos nÃºmeros \*/

for(i=0; i < N; i++)

{

printf("\nLoja %d: ", i+1);

printf("\nNome : ");

gets(loja[i].nome);

printf("No. peÃ§as : ");

scanf("%li",&loja[i].pecas);

printf("Valor das vendas: R$ ");

scanf("%f",&loja[i].valor);

getchar();

}

printf("\n\*\*\*\*\* Resultados \*\*\*\*\*\n");

/\* Conta a quantidade total de peÃ§as vendidas \*/

total\_pecas = conta\_pecas(loja,N);

printf("\nQuantidade total de peÃ§as vendidas: %li", total\_pecas);

/\* Conta a quantidade total de peÃ§as vendidas \*/

valor\_total = conta\_valor(loja,N);

printf("\nValor total de peÃ§as vendidas : R$ %.2f", valor\_total);

/\* Determina a loja que vendeu mais peÃ§as \*/

maior\_venda(loja,N);

/\* Libera memÃ³ria \*/

free(loja);

printf("\n\n");

return 0;

}

/\* FunÃ§Ã£o que calcula a quantidade total de peÃ§as vendidas \*/

long int conta\_pecas(Eloja \*p, int N)

{

/\* p - ponteiro para estrutura loja \*/

/\* N - quantidade de lojas \*/

int i;

long int cont;

cont = 0;

for(i=0; i<N; i++)

{

cont = cont + p[i].pecas;

}

return cont;

}

/\* FunÃ§Ã£o que calcula o valor total de peÃ§as vendidas \*/

float conta\_valor(Eloja \*p, int N)

{

/\* p - ponteiro para estrutura loja \*/

/\* N - quantidade de lojas \*/

int i;

float cont;

cont = 0.0;

for(i=0; i<N; i++)

{

cont = cont + p[i].valor;

}

return cont;

}

/\* FunÃ§Ã£o que determina que loja vendeu mais peÃ§as\*/

void maior\_venda(Eloja \*p, int N)

{

/\* p - ponteiro para as lojas \*/

/\* N - quantidade de lojas \*/

int i;

long int aux;

/\* Busca o maior nÃºmero de peÃ§as vendidas \*/

aux = p[0].pecas;

for(i=1; i < N; i++)

{

if( aux < p[i].pecas )

{

aux = p[i].pecas; /\* se a quantidade de peÃ§as de aux for menor do a quantidade de peÃ§as da loja p[i], troca o conteÃºdo de aux pelo conteÃºdo de loja p[i].pecas \*/

}

}

/\* Imprime resultados \*/

printf("\n\nLoja que vendeu mais peÃ§as: ");

for(i=0; i < N; i++)

{

if( aux == p[i].pecas )

{

printf("\n\nNome : ");

puts(p[i].nome);

printf("No. peÃ§as : %li",p[i].pecas);

printf("\nValor das vendas: %.2f",p[i].valor);

}

}

}

//3

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define LIVRE 0

#define LOCADO 1

#define TAM 100

typedef struct Efilme{

char nome[TAM]; /\* nome do filme \*/

char nome\_cliente[TAM];

int status;

} Efilme;

int N; /\* variÃ¡vel global - quantidade de filmes \*/

Efilme \*filme; /\* variÃ¡vel global - ponteiro para os filmes \*/

void cadastra\_filmes(void); /\* protÃ³tipo funÃ§Ã£o que cadastra os filmes \*/

void loca\_filmes(void); /\* protÃ³tipo funÃ§Ã£o que loca os filmes \*/

void consulta\_filmes(void); /\* protÃ³tipo funÃ§Ã£o que consulta o status de todos os filmes \*/

void libera\_filmes(void); /\* protÃ³tipo funÃ§Ã£o que libera os filmes \*/

int main() {

int op;

/\* Limpa a tela \*/

system("clear");

printf("\n\*\*\*\*\* Cadastra e loca filmes \*\*\*\*\*\n");

while(1)

{

printf("\n\nEntre com uma das opÃ§Ãµes abaixo:\n");

printf("\n(1) - Cadastrar filmes");

printf("\n(2) - Locar filmes");

printf("\n(3) - Liberar filmes");

printf("\n(4) - Consultar cadastro total");

printf("\n(5) - Sair");

printf("\nOpÃ§Ã£o: ");

scanf("%d",&op);

getchar();

switch (op) {

case 1: cadastra\_filmes();

break;

case 2: loca\_filmes();

break;

case 3: libera\_filmes();

break;

case 4: consulta\_filmes();

break;

case 5: free(filme);

printf("\n\n");

return 0;

default: printf("\nOpÃ§Ã£o invÃ¡lida");

}

}

}

/\* FunÃ§Ã£o que cadastra os filmes \*/

void cadastra\_filmes(void)

{

int i;

/\* Entrada de dados \*/

printf("\n\nCadastro - Entre com a quantidade de filmes: ");

scanf("%d",&N);

getchar();

/\* Aloca memÃ³ria \*/

filme = (Efilme \*) malloc( N\*sizeof(Efilme) );

if( filme == NULL ) /\* testa se a memÃ³ria foi alocada \*/

{

printf("ERRO - MemÃ³ria insuficiente");

exit(1);

}

/\* Cadastro dos filmes \*/

for(i=0; i < N; i++)

{

printf("\nFilme %d ", i+1);

printf("\nNome: ");

gets(filme[i].nome);

filme[i].status = LIVRE;

}

}

/\* FunÃ§Ã£o que faz a locaÃ§Ã£o dos filmes \*/

void loca\_filmes(void)

{

int i;

char nome[TAM];

printf("\nLocaÃ§Ã£o - Entre com o nome do filme: ");

gets(nome);

for(i=0; i < N; i++)

{

if( strcmp(nome,filme[i].nome) == 0 )

{

if( filme[i].status == LIVRE )

{

printf("Filme disponÃ­vel");

printf("\nEntre com o nome do cliente: ");

gets(filme[i].nome\_cliente);

filme[i].status = LOCADO;

break;

}

else

{

printf("Filme jÃ¡ estÃ¡ locado");

break;

}

}

else if(i == N-1)

{

printf("Filme nÃ£o cadastrado");

}

}

}

/\* FunÃ§Ã£o que libera filmes \*/

void libera\_filmes(void)

{

int i;

char nome[TAM];

printf("\nLiberaÃ§Ã£o - Entre com o nome do filme: ");

gets(nome);

for(i=0; i < N; i++)

{

if( strcmp(nome,filme[i].nome) == 0 )

{

if( filme[i].status == LOCADO )

{

printf("Filme liberado");

strcpy(filme[i].nome\_cliente," ");

filme[i].status = LIVRE;

break;

}

else

{

printf("Filme jÃ¡ estava livre");

break;

}

}

else if(i == N-1)

{

printf("Filme nÃ£o cadastrado");

}

}

}

/\* FunÃ§Ã£o que consulta todos os filmes \*/

void consulta\_filmes(void)

{

int i;

printf("\nConsulta - Todo cadastro: ");

/\* Consulta filmes \*/

for(i=0; i < N; i++)

{

printf("\n\nFilme %d ", i+1);

printf("\nNome : ");

puts(filme[i].nome);

if( filme[i].status == LIVRE )

printf("Status : LIVRE");

else

{

printf("Status : LOCADO");

printf("\nCliente: ");

puts(filme[i].nome\_cliente);

}

}

}

//4

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define TAM 20

typedef struct Ecarro{

char modelo[TAM]; /\* nome do carro \*/

char marca[TAM];

char cor[TAM];

} Ecarro;

int main() {

int i, op;

int N; /\* quantidade de carros \*/

int cont;

Ecarro \*carro; /\* ponteiro para os carros \*/

char nome[TAM];

/\* Limpa a tela \*/

system("clear");

/\* Entrada de dados \*/

printf("\n\*\*\*\*\* Cadastro e consulta de carros \*\*\*\*\*\n");

printf("\nCadastro - Entre com a quantidade de carros: ");

scanf("%d",&N);

getchar();

/\* Aloca memÃ³ria \*/

carro = (Ecarro \*) malloc( N\*sizeof(Ecarro) );

if( carro == NULL ) /\* testa se a memÃ³ria foi alocada \*/

{

printf("ERRO - MemÃ³ria insuficiente");

exit(1);

}

/\* Cadastro dos carros \*/

for(i=0; i < N; i++)

{

printf("\nCarro %d ", i+1);

printf("\nModelo: ");

gets(carro[i].modelo);

printf("Marca : ");

gets(carro[i].marca);

printf("Cor : ");

gets(carro[i].cor);

}

/\* Consulta \*/

while(1)

{

printf("\nEscolha uma da opÃ§Ãµes abaixo: ");

printf("\n(1) - Listar carros pelo modelo");

printf("\n(2) - Listar carros pela marca");

printf("\n(3) - Listar carros pela cor");

printf("\n(4) - Sair");

printf("\nOpÃ§Ã£o: ");

scanf("%d",&op);

getchar();

switch (op) {

case 1: printf("\nEntre com o modelo: ");

gets(nome);

cont = 0;

for(i=0; i< N; i++)

{

if( strcmp(nome,carro[i].modelo) == 0 )

{

printf("\nCarro %d ", i+1);

printf("\nMarca : ");

puts(carro[i].marca);

printf("Cor : ");

puts(carro[i].cor);

cont = cont + 1;

}

}

printf("Quantidade de carros desse modelo: %d\n",cont);

break;

case 2: printf("\nEntre com a marca: ");

gets(nome);

cont = 0;

for(i=0; i< N; i++)

{

if( strcmp(nome,carro[i].marca) == 0 )

{

printf("\nCarro %d ", i+1);

printf("\nModelo: ");

puts(carro[i].modelo);

printf("Cor : ");

puts(carro[i].cor);

cont = cont + 1;

}

}

printf("Quantidade de carros dessa marca: %d\n",cont);

break;

case 3: printf("\nEntre com a cor: ");

gets(nome);

cont = 0;

for(i=0; i< N; i++)

{

if( strcmp(nome,carro[i].cor) == 0 )

{

printf("\nCarro %d ", i+1);

printf("\nModelo: ");

puts(carro[i].modelo);

printf("Marca : ");

puts(carro[i].marca);

cont = cont + 1;

}

}

printf("\nQuantidade de carros dessa cor: %d\n",cont);

break;

case 4: free(carro);

printf("\n\n");

return 0;

default: printf("\nOpÃ§Ã£o invÃ¡lida\n");

}

}

}