



Relatório Projeto Final

Gabriel Vinícius Rocha Barboza, GU3042715

Introdução

Este documento fornece uma análise detalhada do código no projeto "lista-de-contatos", um sistema de gerenciamento de contatos implementado em linguagem C para o projeto final da disciplina Estrutura de Dados I no Instituto Federal de São Paulo - Câmpus Guarulhos.

A primeira decisão tomada foi a utilização de uma estrutura de lista sequencial dinâmica para armazenar informações sobre os clientes. Isso porque é um dos tipos de lista dinâmicas mais simples e que poderia atender perfeitamente os requisitos do negócio.

A segunda grande decisão foi seguir uma abordagem de desenvolvimento orientado a testes. Trecho de código eram desenvolvidos modularizados para atender a requisitos específicos, submetendo cada incremento a testes imediatos. Esse método permitiu validar cada funcionalidade à medida que era implementada, garantindo a estabilidade e a precisão do sistema.

A estrutura do código assemelha-se às estruturas ensinadas em aula, embora apresente algumas modularizações adicionais. Um exemplo disso é a criação do método "lista_existe()", o qual foi introduzido no código para reduzir a repetição de verificações de "lista != null" em vários outros métodos. O objetivo disso foi deixar os códigos mais intuitivos para ser mais fácil de algum programador o ler quando for fazer alguma manutenção ou criar uma nova funcionalidade.

Estrutura da Lista

O código define um struct `CLIENTE` para armazenar as informações do cliente (código, nome, empresa, departamento, telefone, celular e email). Além disso, há uma estrutura `CONTATO`, que possui dois elementos: um cliente e um ponteiro para o próximo contato

na lista. Já a lista de contatos é representada pela estrutura `LISTA`, que contém ponteiros para o próximo contato.

Persistência de Dados

- A função `restore_contatos(lista)` é chamada no início do programa para carregar os contatos salvos em uma sessão anterior, armazenados em um arquivo binário. (`backup_lista_de_contatos.bin`).
- A função `backup_contatos(lista)` é chamada no final do programa, antes de fechá-lo, para salvar os contatos atuais da lista no mesmo arquivo binário.
- Para conseguir salvar os dados da lista no arquivo, eu pensei em criar um vetor alocado dinamicamente baseado no tamanho da lista (ou seja, no número de contatos cadastrados na lista). Em seguida, eu fiz um loop while na lista para conseguir colocar cada contato num índice do vetor. Então eu fiz uma verificação para saber se o `fwrite` estava realmente cadastrando todos os contatos da lista. E por fim dei um `free()` no vetor, para liberar a memória alocada por ele, e também dei um `fclose()` no arquivo para não correr o risco de corromper nada.
- Já para conseguir trazer esses dados de volta quando o usuário abre o programa, eu comecei abrindo o arquivo e iniciando um loop while que roda enquanto o arquivo não atingir o caractere EOF (End Of File), que é um caractere especial que sinaliza o fim do arquivo. Dentro desse loop, eu fui utilizando a função `fseek()` para mover o ponteiro pelo arquivo usando uma contagem (i) para conseguir ler um cliente por vez. No fim do loop eu sempre pego o próximo caractere depois de inserir o cliente na lista com `fgetc()` para saber se esse caractere é o EOF que deve sinalizar o fim do arquivo e sair do while. Eu termino o código com um `fclose()` para garantir a integridade do programa e dos dados.

Menu Principal

O programa apresenta um menu interativo apresentado em um loop do-while até que o usuário escolha sair (opção 0). Ele permite ao usuário realizar diversas operações requisitadas pelo negócio:

```
// Menu em Loop
char choice; // Guarda o primeiro character do que o usuario digitar para decidir no switch o que fazer com o que o usuario digitou
do
{
    system("cls");
    printf("~~~~~ Lista de Contatos ACME S.A. - %s ~~~~~\n", VERSION);
    printf("1 - Inserir de novo contato\n");
    printf("2 - Gerar relatorio total\n");
    printf("3 - Gerar relatorio individual\n");
    printf("4 - Gerar relatorio por nome\n");
    printf("5 - Editar contato\n");
    printf("6 - Excluir contato\n");
    printf("0 - Sair do programa\n");
    printf("\nInforme o numero da acao que deseja: ");
    fflush(stdin); // Para garantir que somente o que o usuario digitar sera guardado
    scanf("%c", &choice);
    system("cls");

    switch(choice)
    {
        case '1':
```

Operações no Menu

As operações no menu são responsáveis por fazer toda a interação do usuário com as funções do header `lista_de_contatos.h` que executam os códigos encapsulados no `lista_de_contatos.c` para gerenciar a lista de contatos alocada na memória.

Eu optei por receber o que o usuário digita para escolher uma opção do menu como char, para poder usar um switch para determinar o que acontece baseado na escolha do usuário (quais funções são chamadas? com quais parâmetros?). Isso garante que ele tenha que digitar exatamente o número da opção que deseja e não algo com número equivalente na tabela ASCII.

- **Inserir Novo Contato (case '1'):** Utiliza a função `inserir_contato()` para adicionar um novo contato na lista. Os dados do contato são coletados usando `coletar_cliente()`.
- **Gerar Relatório Total (case '2'):** Utiliza a função `listar_contatos()` para exibir informações de todos os contatos.
- **Gerar Relatório Individual (case '3'):** Utiliza a função `listar_contatos_codigo()` para exibir informações de um contato específico, cujo código é coletado usando `coletar_codigo()`.
- **Gerar Relatório por Nome (case '4'):** Coleta um nome usando `coletar_nome()` e utiliza a função `listar_contatos_nome()` para exibir informações dos contatos que possuem esse nome.
- **Editar Contato (case '5'):** Utiliza a função `editar_contato_processo()` para editar informações de um contato existente.

- **Excluir Contato (case '6'):** Utiliza a função `remover_contato_processo()` para excluir um contato da lista.
- **Sair do Programa (case '0'):** Finaliza o loop do-while e encerra o programa. Antes de sair, realiza o backup dos contatos.

Principais Testes

O desenvolvimento do programa foi feito orientado a testes, e para atingir isso, implementei uma funcionalidade do menu por vez (o que, por consequência, fazia eu implementar funções que seriam usadas posteriormente até em outras opções do menu), executando o programa com a tecla F9 do teclado e interagindo com o menu.

Um dos primeiros testes que fiz, foi para testar se a função de inserir cliente estava funcionando, ao tentar inserir um cliente com código fixo ao acessar a opção 1 do menu, como demonstram as imagens abaixo:

```
switch(choice)
{
case '1':
    cliente.cod = 100;
    printf("\n\Contato cadastrado: %d\n\n", cliente_cadastrado(lista, cliente));
    if(inserir_contato(lista, cliente))
    {
        printf("Contato inserido com sucesso!");
    }
    else
    {
        printf("Falha ao cadastrar contato!");
    }
    break;
}
```

```
main.c [lista-de-contatos] - Code::Blocks 20.03

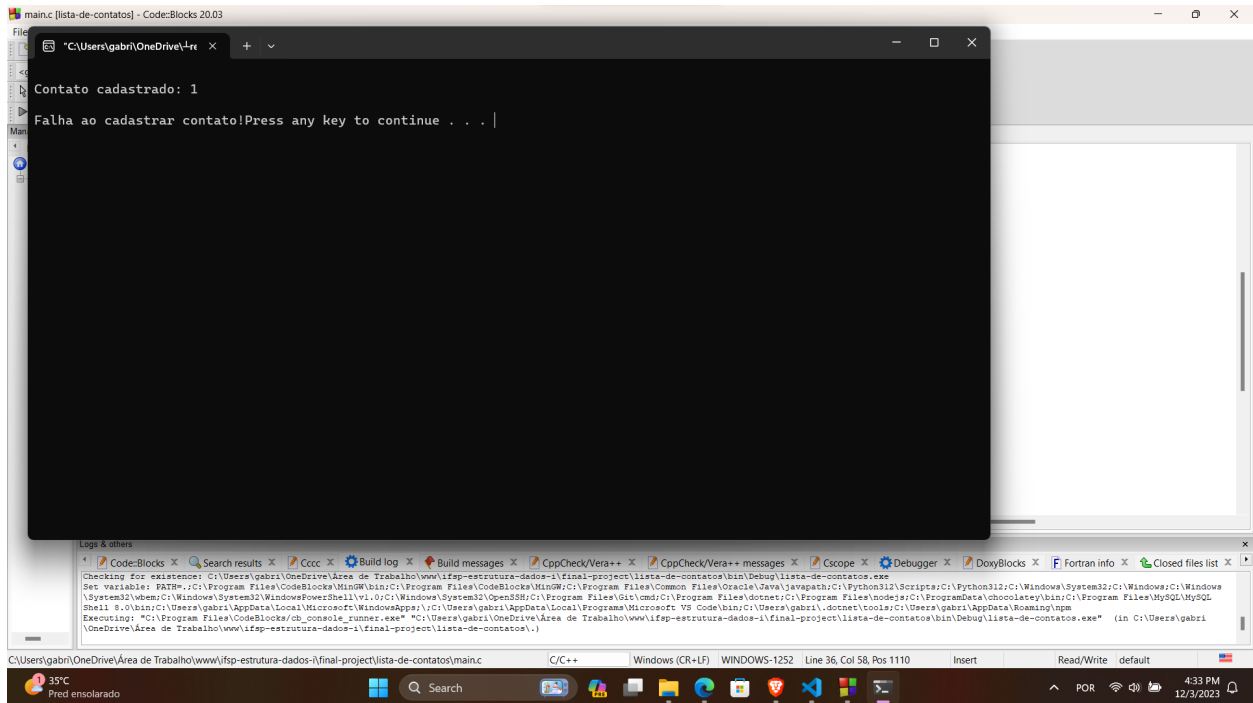
Lista de Contatos ACME S.A. - v1.0

1 - Inserir de novo contato
2 - Gerar relatório total
3 - Gerar relatório individual
4 - Gerar relatório por nome
5 - Editar contato
6 - Excluir contato
0 - Sair do programa

Informe o numero da acao que deseja: 1|
```

```
main.c [lista-de-contatos] - Code::Blocks 20.03

Contato cadastrado: 0
Contato inserido com sucesso! Press any key to continue . . . |
```



Outro teste que fiz, foi inserir aquele cliente fixo do teste anterior para testar se a função de `listar_contatos` funcionava como demonstra abaixo:

```
break;
case '2':
    listar_contatos(lista);
    break;
```

```
"C:\Users\gabri\OneDrive\tr... X + v
----- Lista de Contatos ACME S.A. - v1.0 -----
1 - Inserir de novo contato
2 - Gerar relatorio total
3 - Gerar relatorio individual
4 - Gerar relatorio por nome
5 - Editar contato
6 - Excluir contato
0 - Sair do programa

Informe o numero da acao que deseja: 2|
```

```
"C:\Users\gabri\OneDrive\tr... X + v
----- Relatorio Completo -----

Nada para mostrar, lista esta vazia!Press any key to continue . . . |
```

```
"C:\Users\gabri\OneDrive\Trabalho\Projetos\Projeto Final\src\bin\Debug\net6.0\Projeto Final.exe" X + v - □ X
Relatorio Completo
Codigo.....: 100
Nome.....:
Empresa.....:
Departamento.:
Telefone.....:
Celular.....:
Email.....: Press any key to continue . . . |
```

Outro teste essencial, foi executar as opções do menu para inserir 3 contatos e verificar se a função de busca_nome estava funcionando corretamente, já que ela busca o que o usuário digitou nos nomes cadastrados na lista e pode retornar vários contatos:

lista-de-contatos.c [lista-de-contatos] - Code::Blocks 20.03

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

<global> buscar_nome(LISTA *li, char nome[], LISTA *contatos_encontrados): int

```
232 int buscar_nome(LISTA *li, char nome[], LISTA *contatos_encontrados)
233 {
234     char lower_nome[60], lower_selected_nome[60];
235     char copy_nome[60];
236     strcpy(copy_nome, nome);
237     strlwr(copy_nome);
238     strcpy(lower_nome, copy_nome);
239     if(!lista_vazia(li))
240     {
241         CONTATO *c = *li;
242         while(c != NULL)
243         {
244             strcpy(copy_nome, c->dados.nome);
245             strlwr(copy_nome);
246             strcpy(lower_selected_nome, copy_nome);
247             if(strstr(lower_selected_nome, lower_nome) != NULL)
248             { //strstr devolve valor diferente de 0, ou seja, true, caso encontre o nome a ser buscado no contato atual selecionado
249                 printf("\n%s esta contida em %s", lower_nome, lower_selected_nome);
250                 system("pause");
251                 inserir_contato(contatos_encontrados, c->dados);
252             }
253             c = c->prox_contato;
254         }
255     }
256 }
```

Logs & others

Code::Blocks Search results Cccc Build log Build messages CppCheck/Vera++ CppCheck/Vera++ messages Cscope Debugger DoxyBlocks Fortran info Closed files list

File Line Message

C:\Users\gabr... 341 warning: implicit declaration of function 'tolower' [-Wimplicit-function-decl...

C:\Users\gabr... 341 warning: implicit declaration of function 'tolower' [-Wimplicit-function-decl...

Build finished: 0 error(s), 1 warning(s) (0 minute(s), 0 second(s))

C:\Users\gabr\OneDrive\Área de Trabalho\www\ifsp-estrutura-dados-3\final-project\lista-de-contatos\lista-de-contatos.c C/C++ Windows (CR+LF) WINDOWS-1252 Line 255, Col 38, Pos 5695 Insert Read/Write default

23°C Limpo

"C:\Users\gabr\OneDrive\Área de Trabalho\www\ifsp-estrutura-dados-3\final-project\lista-de-contatos\lista-de-contatos.c" + -

Relatorio Completo

```
Codigo.....: 1
Nome.....: Gabriel Barboza
Empresa.....: 1
Departamento.: 1
Telefone.....: 1
Celular.....: 1
Email.....: 1

Codigo.....: 2
Nome.....: Vinicius Gabriel
Empresa.....: 2
Departamento.: 2
Telefone.....: 2
Celular.....: 2
Email.....: 2

Codigo.....: 3
Nome.....: Cintia Sabino
Empresa.....: 3
Departamento.: 3
Telefone.....: 3
Celular.....: 3
Email.....: 3

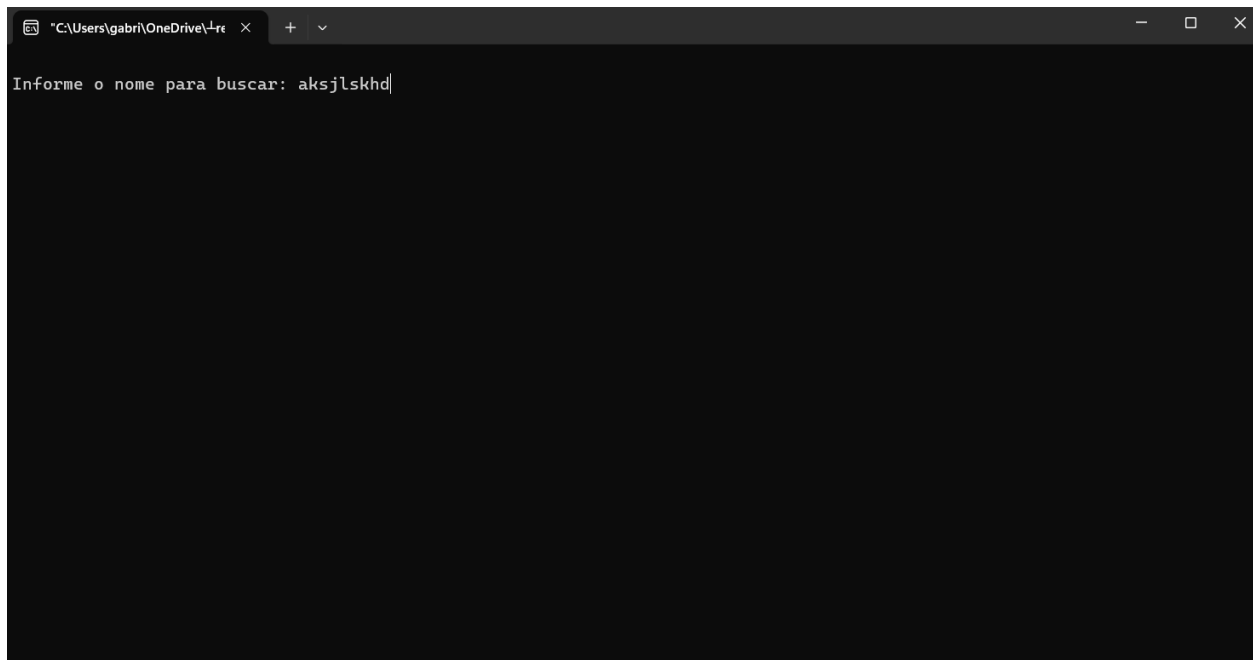
Press any key to continue . . . |
```

```
"C:\Users\gabri\OneDrive\tr... x + v
Informe o nome para buscar: Gabriel|
```

```
"C:\Users\gabri\OneDrive\tr... x + v
===== Relatorio =====
Codigo.....: 1
Nome.....: Gabriel
Empresa.....: 1
Departamento.: 1
Telefone.....: 1
Celular.....: 1
Email.....: 1

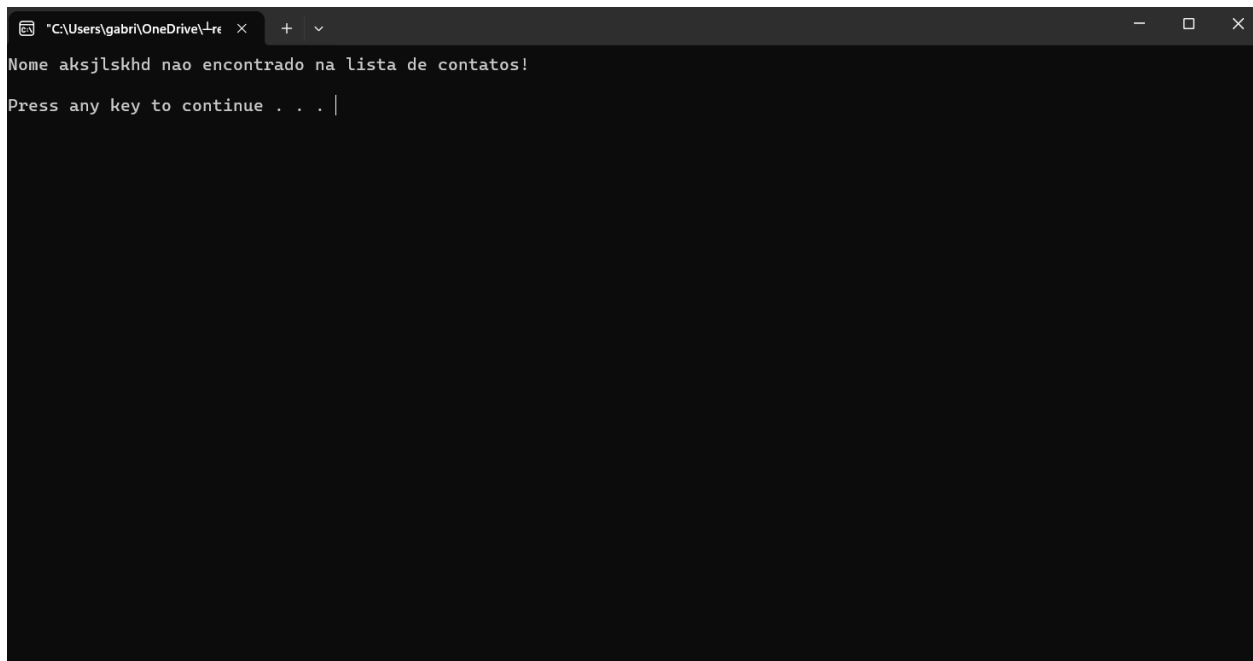
Codigo.....: 2
Nome.....: Vinicius Gabriel
Empresa.....: 2
Departamento.: 2
Telefone.....: 2
Celular.....: 2
Email.....: 2

Press any key to continue . . . |
```



A screenshot of a terminal window with a dark background. The title bar shows the file path "C:\Users\gabri\OneDrive\...". The terminal displays the text "Informe o nome para buscar: aksjlskhd" with a cursor at the end of the input.

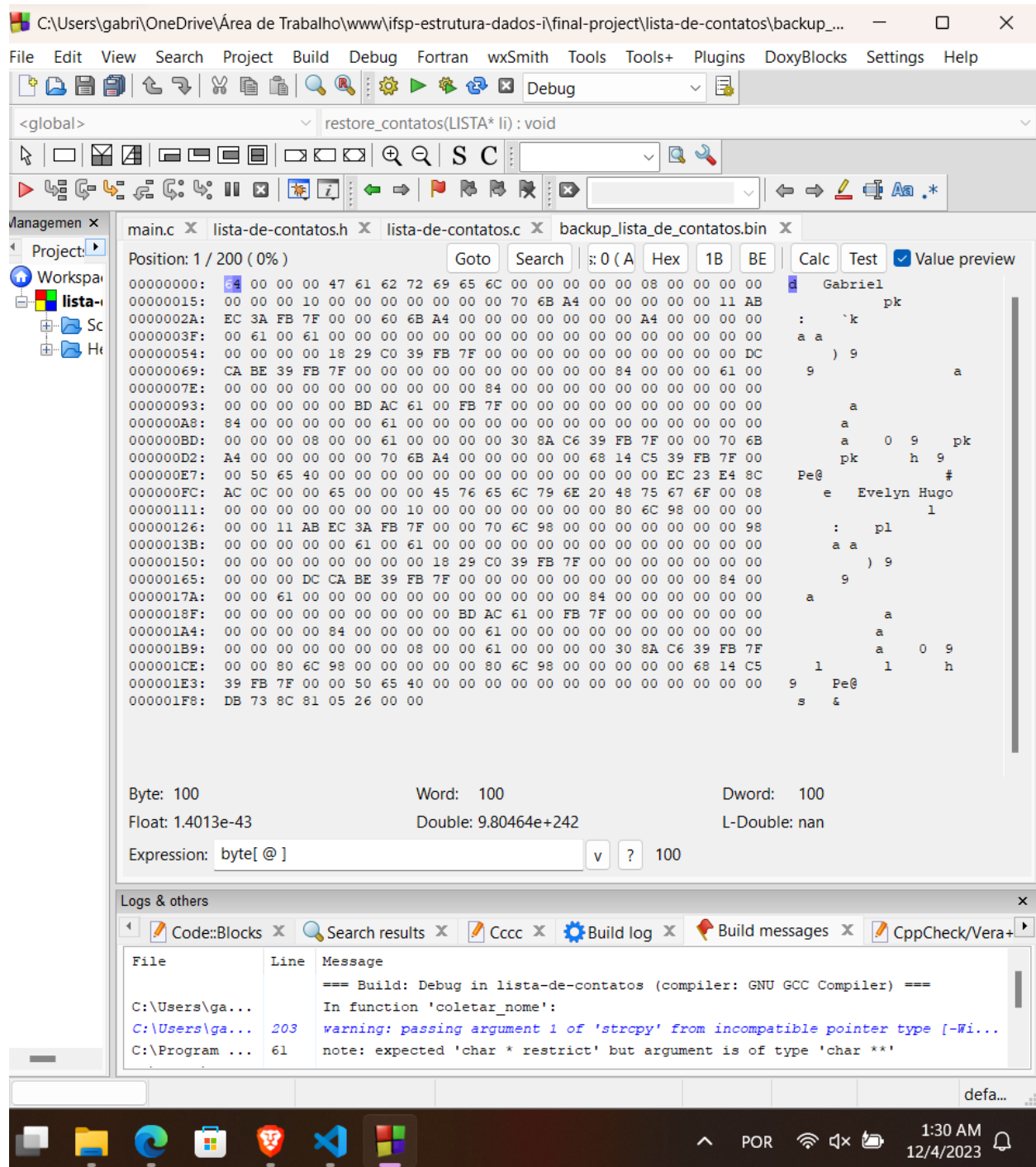
```
"C:\Users\gabri\OneDrive\... x + v  
Informe o nome para buscar: aksjlskhd|
```



A screenshot of a terminal window with a dark background. The title bar shows the file path "C:\Users\gabri\OneDrive\...". The terminal displays two lines of text: "Nome aksjlskhd nao encontrado na lista de contatos!" followed by "Press any key to continue . . . |" with a cursor at the end.

```
"C:\Users\gabri\OneDrive\... x + v  
Nome aksjlskhd nao encontrado na lista de contatos!  
Press any key to continue . . . |
```

Após criar a função de backup, abri o arquivo binário pelo Code Blocks para visualizar se o salvamento estava ocorrendo como eu desejava:



Principal Bug Enfrentado

```

int buscar_codigo(LISTA *li, int cod, LISTA *contato)
{
    if(!lista_vazia(li)) // se lista nao estiver vazia
    {
        CONTATO *c = *li;
        while(c != NULL && c->dados.cod != cod)
        {
            c = c->prox_contato;
        }

        if(c != NULL) // se tiver encontrado um contato com mesmo codigo
        {
            *contato = c; // retorna ponteiro para contato encontrado no parametro *contato
            return 1;
        }
    }
    return 0;
}

```

Esse código retornava toda a lista de contatos, porque eu enviava c inteiro para o ponteiro *contato, e o c tem um c que aponta para outro c, que aponta para outro... até chegar em NULL.

Então quando eu usava esse código em outras funções como remover ou editar, ele excluía todos os contatos após esse na lista.

Código atual, corrigido:

```

264 int buscar_codigo(LISTA *li, int cod, CLIENTE *contato)
265 {
266     if(!lista_vazia(li)) // se lista nao estiver vazia
267     {
268         CONTATO *c = *li;
269         while(c != NULL && c->dados.cod != cod)
270         {
271             c = c->prox_contato;
272         }
273
274         if(c != NULL) // se tiver encontrado um contato com mesmo codigo
275         {
276             *contato = c->dados; // guarda o CLIENTE encontrado na variável apontada por *contato
277             return 1;
278         }
279     }
280     return 0;
281 }

```

Eu corriji o bug devolvendo para o ponteiro *contato somente a estrutura CLIENTE (CONTATO → dados). Tive que trocar o parâmetro para CLIENTE *contato (no lugar de LISTA*) e criar um CLIENTE nas funções que chamar a buscar_codigo() para passar o endereço deles usando &, como demonstra abaixo:

```

    CLIENTE c;
    int cod = coletar_codigo(), success;

    success = buscar_codigo(li, cod, &c);
    if(success)

```

Outro bug que enfrentei foi, quando eu apagava todos os contatos, ao fechar o programa todos eram recuperados. Isso porque eu estava fazendo um if no método backup_contatos() que fazia nada ser salvo quando a lista não tinha nenhum contato, e isso o impedia de apagar todos que estava no arquivo.

```

if(total_contatos > 0)
{
    // Criando variavel do tipo arquivo
    FILE *f;

    // Tentando abrir arquivo usando modo de escrita binaria (wb)
    f = fopen("backup_lista_de_contatos.bin", "wb");
    if(f == NULL)
    { // Se nao conseguir abrir o arquivo, emitir erro e fechar programa
        printf("Erro na abertura de 'backup_lista_de_contatos.bin'!\n\n");
        system("pause");
        exit(1); // fechar programa com código de erro
    }
}

```

Conclusão

Este projeto implementa um sistema de gerenciamento de contatos robusto usando lista sequencial dinâmica em linguagem C. O menu chama funções que oferecem um CRUD para administrar os contatos na lista e backup/restauração de contatos, proporcionando um sistema completo, útil e de uso fácil para o usuário.