

**INSTITUTO FEDERAL DE EDUCAÇÃO CIÊNCIA E TECNOLOGIA DE MINAS  
GERAIS – CAMPUS SÃO JOÃO EVANGELISTA  
BACHARELADO EM SISTEMAS DE INFORMAÇÃO**

**GABRIEL KÁICON BATISTA HILÁRIO**

**TRABALHO PRÁTICO I**

**SÃO JOÃO EVANGELISTA  
SETEMBRO - 2022**

GABRIEL KÁICON BATISTA HILÁRIO

SISTEMA DE CADASTRO DE FUNCIONÁRIOS E DE PROJETOS DENTRO DE  
FUNCIONÁRIOS

**SÃO JOÃO EVANGELISTA**  
**SETEMBRO - 2022**

## SUMÁRIO

1. INTRODUÇÃO	5
1.1. Objetivo Geral.....	5
1.2. Objetivos Específicos.....	5
1.3. Justificativa .....	5
2. DESENVOLVIMENTO	7
2.1. Conceitos Aplicados.....	7
2.1.1. Tipos Abstratos de Dados .....	7
2.1.2. Lista Encadeada .....	8
2.1.3. Lista Sequencial.....	9
2.1.4. Ponteiros .....	9
2.1.5. Arquivos.....	9
2.2. Implementação.....	10
3. CONCLUSÃO	12
4. REFERÊNCIAS	14
5. APÊNDICES	15
5.1. APÊNDICE A – Funções das TADs no arquivo.cpp.....	15
5.2. APÊNDICE B – Sistema.hpp.....	17
5.3. APÊNDICE C – Manipulação de arquivos.....	18
5.4. APÊNDICE D – Menu .....	19
5.5. APÊNDICE E – Função de Inserção de funcionário.....	20
5.6. APÊNDICE F – Função de retorno de funcionário, para inserção .....	20
5.7. APÊNDICE G – Inclusão de Projeto .....	21
5.8. APÊNDICE H – Pesquisa funcionário por ID .....	21
5.9. APÊNDICE I – Exclui Projetos.....	22
5.10. APÊNDICE J – Exclui funcionários sem Projeto .....	22
5.11. APÊNDICE K – Impressão de Funcionário .....	23

5.12. APÊNDICE L – Cálculo de Salário Bruto.....	23
--	----

## 1. INTRODUÇÃO

Este trabalho prático foi documentado para que seja avaliado em conjunto com os códigos na linguagem C/C++, exigido pelo docente Eduardo Augusto da Costa Trindade, dentro da disciplina de Algoritmos e Estruturas de Dados I, ministrada pelo mesmo. Porém a documentação tem cunho expositivo, onde é descrito as funcionalidades do programa, com testes, e desenvolvimento de novas linhas de raciocínio lógico para realização do trabalho prático.

### 1.1. Objetivo Geral

Este trabalho tem como objetivo geral apresentar na prática os conhecimentos adquiridos nas aulas de Algoritmos e Estruturas de Dados I, a respeito de Listas com Arranjo, Listas Encadeadas, Tipos Abstratos de Dados (TADs), Manipulação de Arquivos e Ponteiros, utilizando a linguagem de C++ para escrita dos códigos.

### 1.2. Objetivos Específicos

Esse trabalho tem como objetivos específicos:

- Apresentar conhecimentos em lista encadeada;
- Apresentar conhecimentos em lista com arranjo;
- Apresentar conhecimentos em manipulação de arquivos;
- Aplicar conhecimentos adquiridos em um minissistema, em uma visão empresarial.

### 1.3. Justificativa

Ao iniciar os estudos de Algoritmos e Estruturas de Dados I, vemos os seguintes conteúdos:

- Ponteiros
- TADs
- Manipulação de Arquivos
- Listas:
  - Lista com Arranjo
  - Lista Encadeada

Vemos em ponteiros, a manipulação de valores da variável por meio do endereço de memória, utilizando ponteiros. Vemos em arquivos, os comandos básicos de leitura e gravação de dados em um arquivo por meio de objetos da biblioteca *fstream*, o *ifstream* para leitura e o *ofstream* para gravação.

Posteriormente vemos um conteúdo mais amplo de listas, que consiste na inserção de itens em uma lista, uma estrutura sendo inserida dentro de outra, semelhante a POO (Programação Orientada a Objetos). Inicialmente são apresentados dois tipos de lista:

- A lista com arranjo, também chamada de lista sequencial que é com alocação estática, ou seja, possui um limite pré-definido, podendo ter um número limitado de itens.
- A lista encadeada ou lista com ponteiro que é a com alocação dinâmica, ou seja, não possui um limite pré-definido podendo ter um número infinito de itens.

A princípio ambas são aplicados de forma individual, porém no trabalho, é exigido que usemos as duas juntas, antes devemos cadastrar funcionários em uma lista encadeada, e devemos criar uma lista sequencial dentro de cada funcionário, para cadastrar projetos para esse funcionário que está na lista encadeada.

Os conteúdos foram aplicados para realização do trabalho, criando um minissistema que nos permite cadastrar e excluir funcionários, adicionar e retirar projetos ao funcionário, e calcular e imprimir o contracheque.

Tendo isso tudo em vista, o trabalho foi exigido para que seja possível desenvolver o raciocínio lógico quanto a aplicação dessas estruturas em conjunto.

## 2. DESENVOLVIMENTO

Nesta seção do documento é apresentado, os conceitos aprendidos e o desenvolvimento do trabalho em si, na linguagem C++.

### 2.1. Conceitos Aplicados

Explicação sucinta dos conceitos de Lista Encadeada, Lista sequencial, arquivos e Tipos Abstratos de Dados(TADs).

#### 2.1.1. Tipos Abstratos de Dados

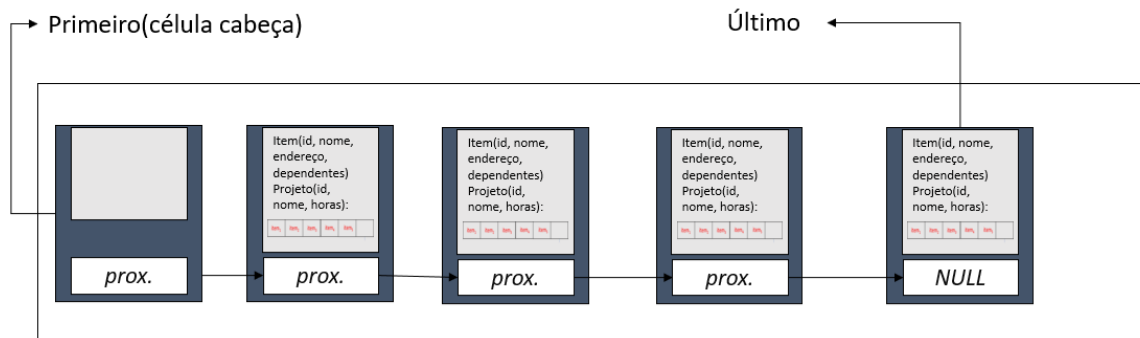
As estruturas utilizadas como registro para criação de objetos, e as funções de manipulação dessas estruturas, ambas compõem uma TAD. Declaramos esses modelos como *structs*, elas são representações de qualquer coisa no mundo real, sendo ela lógica, abstrata ou física, como por exemplo uma pessoa, que é algo físico, ou um filme digital, que é algo lógico/abstrato, veja o exemplo na figura 1. Cada um tem suas características específicas, uma pessoa nome, sexo, idade, CPF, altura, dentre outras, e um filme título, linguagem, elenco, personagens, duração, categoria, ano de lançamento, dentre outros, e tudo isso pode ser definida dentro de uma struct para cada um deles. Resumindo uma Struct é uma espécie de variável modelo para cadastrar diferentes itens, dentro de um software escrito em C/C++. Acompanhado das structs temos as funções para manipulação dos dados dessa lista, e desses itens, que será visto no próximo tópico.

Figura 1 – Struct exemplo, sem ligação com o trabalho

```
6  typedef struct Horario{
7      int hora;
8      int min;
9  };
10
11 typedef struct Data{
12     int dia;
13     int mes;
14     int ano;
15 };
16
17 typedef struct Compromisso{
18     char descricao[100];
19     Data dt;
20     Horario hr;
21 };
```

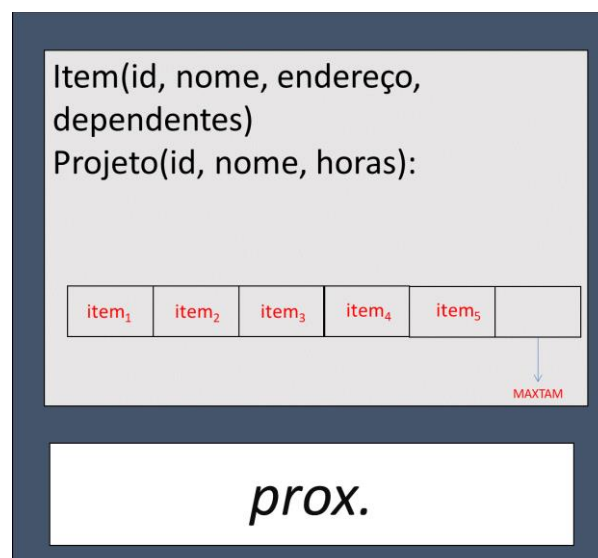
### 2.1.2. Lista Encadeada

Figura 2 – Lista Encadeada com Sequencial



Na figura 2, vemos um desenho esquemático de como seria uma lista encadeada. Cada espaço (em cinza azulado) da lista, tem um apontador, que aponta para a célula seguinte, e uma célula (em cinza claro), essa célula tem o Funcionário (lá está como item) que é uma struct e o projeto, a nossa lista sequencial, que é uma struct, com isso temos, uma struct dentro de um campo de outra struct, comom vemos na figura 3. O primeiro, recebe a célula como NULL, pois ele deve apontar para o primeiro item, e o último, tem o apontador NULL, pois não há uma célula seguinte. Com isso, podemos inserir no inicio ou no fim da lista, e também após um item.

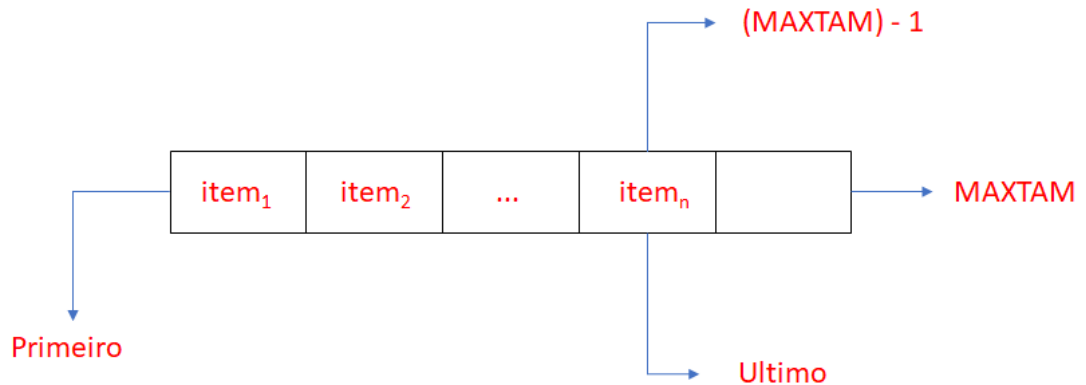
Figura 3 – Célula da Lista





### 2.1.3. Lista Sequencial

Figura 4 – Lista Sequencial



Na figura 4, vemos um desenho esquemático de como seria uma lista sequencial, e o limite dela, é o MAXTAM, uma variável didática muito comum de ser usada, e sua função é delimitar o tamanho da lista. Ela é um vetor, porém a tipagem de dados a ser inserida podem ser as structs, podendo armazenar mais de um tipo de variável dentro de uma posição. Os elementos são inseridos dentro do índice do vetor no ultimo, que seria o apontador para ultima posição até o momento, e assim sucessivamente, até que a Lista alcance o tamanho máximo, que seria o valor definido para o MAXTAM.

### 2.1.4. Ponteiros

Ponteiros são variáveis especiais, que podem armazenar um endereço de memória, e manipular o conteúdo desses endereços de memória. Tem aplicações como manipulação de vetores, passagem de parâmetro por referência, desenvolvimento de estruturas de dados complexos. Resumindo, um tipo de variável que nos permite exibir uma struct, ou o conteúdo de um vetor ou variável, apenas pelo endereço de memória dele.

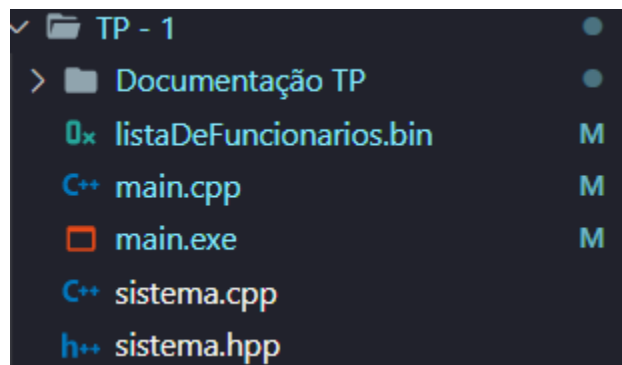
### 2.1.5. Arquivos

A parte de manipulação de arquivos é mais abstrata, não é convencional para ilustrar ela, mas podemos usar uma analogia para compreender, imagine-se estudando para uma prova, onde se usa um livro e uma folha de papel em branco, o papel é seu arquivo, e o livro é seu programa, quando você quer guardar algo importante do livro, você lê o que está nele e escreve na folha, isso seria semelhante

às funções do ofstream, que pega os dados digitados no programa e escreve no arquivo, independente da extensão deste. Imagine novamente, com um caderno e uma folha apenas, o caderno é o programa, e a folha é o arquivo com o que você leu dele, e não estava em branco quando começou tendo algo escrito na folha, você realiza a leitura e exibe isso no caderno, seria semelhante às funções do ifstream, que pega os dados do arquivo e exibe na tela.

## 2.2. Implementação

Figura 5 – Organização



O trabalho foi dividido em 5 arquivos, incluindo a main, e o arquivo bin, como apresentado na figura 4. E o arquivo sistema.cpp com seu arquivo sistema.hpp, e o \*.exe. E a pasta de documentação, com os arquivos que serão enviados.

O sistema.cpp possui uma parte para as funções básicas para manipulação das TADs(Apêndice A) da Lista Encadeada e da Lista Sequencial.

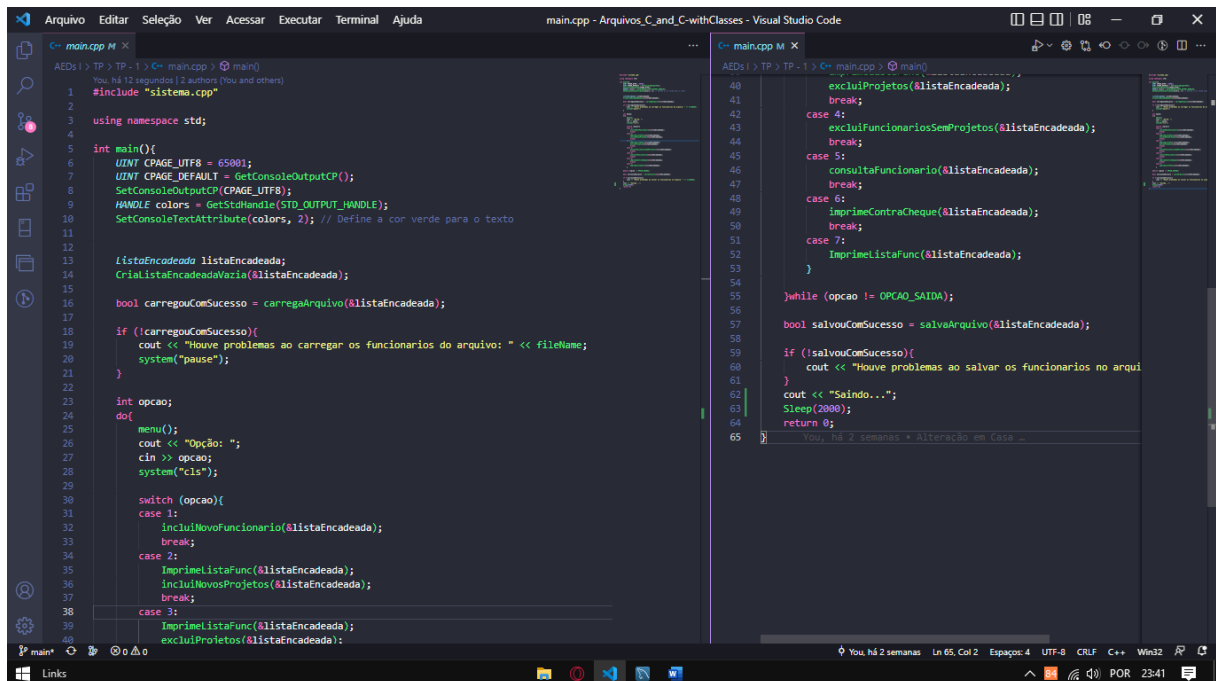
- void menu(); - Menu que é exibido na tela(Apêndice D)
- void incluiNovoFuncionario(ListaEncadeada \*lista); - função para a inclusão de funcionário (Apêndice E)
- Funcionario \*criaFuncionario(); - função com cout e cin que retorna o funcionário, para ser inserido(Apêndice F)
- void incluiNovosProjetos(ListaEncadeada \*lista); - incluir novos projetos, é para perguntar, se quer incluir projetos, e quantos projetos quer incluir, para inserir de fato o projeto com o incluiNovoProjeto(Apêndice G)

- void incluiNovoProjeto(Funcionario \*funcionario); - insere o projeto ao funcionário (Apêndice G)
- void criaProjeto(ListaSequencial \*lista, Projeto \*projeto); - pega o projeto como parâmetro, e faz alterações nele dentro da função (Apêndice G)
- Funcionario \*FuncionarioPorID(ListaEncadeada \*lista); - função para pesquisar um funcionário por id dentro da função, e retornar, um funcionário, para manipulação de outras funções (Apêndice H)
- void excluiProjetos(ListaEncadeada \*lista); - pesquisa o funcionário na lista encadeada, usa das funções da TAD, para remover o projeto do funcionário (Apêndice I)
- void excluiFuncionariosSemProjetos(ListaEncadeada \*lista); - se existir funcionários sem projetos, ou seja com a Lista Sequencial com tamanho = 0, esse funcionário é excluído (Apêndice J)
- void ImprimeListaFunc(ListaEncadeada \*lista); - Imprime a lista de funcionários (Apêndice K)
- void consultaFuncionario(ListaEncadeada \*lista); - pesquisa um funcionário pelo id dele, e exibe na tela as informações dele (Apêndice K)
- int totalHorasSemanais(ListaSequencial lista); - calcula o somatório de horas semanais, de um funcionário (Apêndice L)
- double calculaSalarioBruto(Funcionario \*funcionario); - calcula o salário bruto de acordo com base nas horas semanais (Apêndice L)

- void imprimeContraCheque(ListaEncadeada \*lista); - calcula o salário líquido, com base nos descontos no salário bruto e imprime o contracheque (Apêndice L)
- // FUNÇÕES DE MANIPULAÇÃO DE ARQUIVOS
- bool carregaArquivo(ListaEncadeada \*lista); - ele pega o que foi cadastrado na struct, e insere na última posição da lista, no momento em que o programa é iniciado (Apêndice C)
- bool salvaArquivo(ListaEncadeada \*lista); - ele pega o que foi cadastrado durante a execução e salva no arquivo bin, e deleta ele da memória (Apêndice C)

Após todas as funções, chamo as funções de carregamento e salvamento do arquivo, e as funções exigidas pelo trabalho, como na figura 6.

Figura 6 – Main



### 3. CONCLUSÃO

Ao longo do trabalho encontrei dificuldades para interpretar o que estava sendo pedido, fui a monitoria para sanar minhas dúvidas, conversei com poucos

colegas de classe, para me auxiliarem na interpretação e na escrita do código do trabalho, fiz uso dos slides disponibilizados pelo professor para sanar dúvidas a respeito das TADs (Tipos Abstratos de Dados), criação de funções de manipulação de arquivos, e também utilizei as informações da biblioteca, sites e fóruns.

Coloquei o trabalho em modularização separando-o em 5 arquivos, 2 arquivos \*.cpp, o sistema.cpp, com as funções e a main.cpp com a execução das funções, 1 arquivo \*.hpp, o sistema.hpp, onde tinha as structs e o cabeçalho das funções, 1 arquivo \*.bin, listaDeFuncionarios.bin onde os funcionários eram salvos, e 1 arquivo \*.exe, a main.exe, que seria o executável do código, onde as funções compilavam.

Creio eu que meu desenvolvimento apesar das dificuldades de interpretação, foram ótimos, aprendi coisas novas como funções que retornam um item, manipular arquivos, usar duas Estruturas de Dados para compor 1 só. Achei bastante semelhante ao que fiz no Ensino Médio Técnico Integrado, onde fizemos um sistema para atender à uma pequena empresa. Acho que consegui atingir meus objetivos, e explicar bem o que cada função faz, apresentar conhecimentos em lista encadeada, lista sequencial, manipulação de arquivos, e aplicar isso em um software, que atendesse o mínimo que uma empresa precisaria. Estou feliz com o resultado, e ele atendeu às minhas expectativas além do que eu esperava.

#### 4. REFERÊNCIAS

COSTA TRINDADE. Eduardo Augusto. Algoritmos e Estrutura de Dados - Arquivos. 2022. Apresentação PDF. Disponível em: [https://ead.ifmg.edu.br/saojoaoevangelista/pluginfile.php/146487/mod\\_resource/content/1/Aula%203%20-%20Arquivos.pdf](https://ead.ifmg.edu.br/saojoaoevangelista/pluginfile.php/146487/mod_resource/content/1/Aula%203%20-%20Arquivos.pdf) . Acesso em: 25 de setembro de 2022.

COSTA TRINDADE. Eduardo Augusto. Algoritmos e Estrutura de Dados - Listas. 2022. Apresentação PDF. Disponível em: [https://ead.ifmg.edu.br/saojoaoevangelista/pluginfile.php/146499/mod\\_resource/content/1/Aula%206%20-%20Listas.pdf](https://ead.ifmg.edu.br/saojoaoevangelista/pluginfile.php/146499/mod_resource/content/1/Aula%206%20-%20Listas.pdf) . Acesso em: 24 de setembro de 2022.

COSTA TRINDADE. Eduardo Augusto. Algoritmos e Estrutura de Dados - Listas utilizando Ponteiro. 2022. Apresentação PDF. Disponível em: [https://ead.ifmg.edu.br/saojoaoevangelista/pluginfile.php/146500/mod\\_resource/content/1/Aula%207%20-%20Listas%20Encadeadas.pdf](https://ead.ifmg.edu.br/saojoaoevangelista/pluginfile.php/146500/mod_resource/content/1/Aula%207%20-%20Listas%20Encadeadas.pdf) . Acesso em: 26 de setembro de 2022.

COSTA TRINDADE. Eduardo Augusto. Algoritmos e Estrutura de Dados – Ponteiros. 2022. Apresentação PDF. Disponível em: . Acesso em : 28 de setembro de 2022.

Programar em C++/Entrada e saída de dados 2. **WIKI LIVROS**, 16/04/2020.Disponível em: [https://pt.wikibooks.org/wiki/Programar\\_em\\_C%2B%2B/Entrada\\_e\\_saída\\_de\\_dados\\_2#:~:text=biblioteca%20padrão%20fstreamEditar&text=Esta%20biblioteca%20define%203%20novos,\"in%20from%20a%20file\"%20](https://pt.wikibooks.org/wiki/Programar_em_C%2B%2B/Entrada_e_saída_de_dados_2#:~:text=biblioteca%20padrão%20fstreamEditar&text=Esta%20biblioteca%20define%203%20novos,\). Acesso em: 25 de setembro de 2022.

## 5. APÊNDICES

### 5.1. APÊNDICE A – Funções das TADs no arquivo.cpp

```
Arquivo Editar Seleção Ver Acessar Executar Terminal Ajuda sistema.cpp - Arquivos_C_and_C-withClasses - Visual Studio Code
sistema.cpp x
AEDs | > TP > TP - 1 > C++ sistema.cpp > criaProjeto(ListaSequencial *, Projeto *)
29
30 //////////////////////////////////////////////////LISTA SEQUENCIAL////////////////////////////////////
31
32 void CriaListaSequencialVaria(ListaSequencial *lista){
33     if (!lista->listaSequencialCriada){
34         lista->listaSequencialCriada = true; //lista existe agora, e possui valor true
35         lista->tamanho = 0; //o tamanho da lista(vetor) é 5, o tamanho serve para verificar se está cheia ou não
36     }
37 }
38
39 bool ListaSequencialEstaVazia(ListaSequencial *lista){
40     return lista->tamanho == 0; //verifica se ela está vazia
41 }
42
43 bool ListaSequencialEstaCheia(ListaSequencial *lista){
44     return lista->tamanho == MAX_TAM; //verifica se ela está cheia
45 }
46
47 bool InsereProjeto(ListaSequencial *lista, Projeto projeto){
48     //se a lista não foi criada ou está cheia
49     if (!lista->listaSequencialCriada || ListaSequencialEstaCheia(lista))return false;
50
51     lista->item[lista->tamanho] = projeto; //insere o projeto na última posição(tamanho atual)
52     lista->tamanho++; //e aumenta o tamanho para que possa ser realizada uma nova inserção.
53     return true;
54 }
55
56 void ImprimeProjeto(Projeto projeto){ //impressão básica
57     cout << "Id: " << projeto.id << endl;
58     cout << "Nome: " << projeto.nome << endl;
59     cout << "Horas Trabalhadas: " << projeto.horas << endl;
60     cout << "\n";
61 }
62
63 void ImprimeListaSequencial(ListaSequencial *lista){
64     for (int i = 0; i < lista->tamanho; i++){ //percorre todos os projetos
65         Projeto projeto = lista->item[i];
66         cout << i+1 << " Projeto\n";
67         ImprimeProjeto(projeto);
68         cout << "\n";
69     }
70 }
71
72 Projeto *PesquisaProjetoPorID(ListaSequencial *lista, int id){
73     for (int i = 0; i < lista->tamanho; i++){
74         if (id == lista->item[i].id){
75             return &lista->item[i]; // Retorna o endereço na memória do Projeto
76         }
77     }
78     return NULL; // se não for encontrado retorna NULL, pois o projeto não existe.
79 }
80
81 bool RemoveProjetoPorId(ListaSequencial *lista, int id){
82     if (ListaSequencialEstaVazia(lista)) return false;
83
84     for (int i = id; i < lista->tamanho; i++)lista->item[i] = lista->item[i + 1];
85     lista->tamanho--; //lista diminui de tamanho
86     return true;
87 }
88
89 bool RemoveProjeto(ListaSequencial *lista, Projeto projeto){ //valida se o índice foi achado para só assim executar a exclusão
90     if (ListaSequencialEstaVazia(lista)){
91         cout << "Erro: Lista está vazia" << endl;
92         return false;
93     }
94     // You, há 21 horas • Conclusão do código do TP
95
96     int indice = IndiceDeProjeto(lista, projeto); //o índice encontrado é armazenado na variável "indice"
97     if (indice == -1)return false;
98     return RemoveProjetoPorId(lista, indice); //índice utilizado para remover de fato o projeto
99 }
100
101 int IndiceDeProjeto(ListaSequencial *lista, Projeto projeto){ //procura o índice
102     for (int i = 0; i < lista->tamanho; i++){
103         if (lista->item[i].id == projeto.id) return i; //retorna o índice em que o projeto está
104     }
105     return -1; //não foi encontrado
106 }
107
108 int TamanhoListaSequencial(ListaSequencial *lista){
109     return lista->tamanho;
110 }
111
112 }
```

```
Arquivo Editar Seleção Ver Acessar Executar Terminal Ajuda sistema.cpp - Arquivos_C_and_C-withClasses - Visual Studio Code
sistema.cpp x
AEDs I > TP > TP - 1 > C++ sistema.cpp > ...
113 //////////////////////////////////////////////////LISTA ENCADEADA////////////////////////////////////
114
115 void CriaListaEncadeadaVazia(ListaEncadeada *Lista){
116     if (!Lista->listaEncadeadaCriada){//se a lista não foi criada
117         Lista->primeiro = new Celula; //no apontador primeiro recebe uma nova Celula
118         Lista->ultimo = Lista->primeiro; //o primeiro apontador recebe o último, pois trata-se de uma lista vazia
119         Lista->ultimo->prox = NULL; //o apontador prox do apontador ultimo, recebe null
120         Lista->listaEncadeadaCriada = true; //e a lista foi criada
121     }
122     else{
123         cout << "Lista já foi criada" << endl;
124         Sleep(1000);
125     }
126 }
127
128 bool ListaEncadeadaEstaVazia(ListaEncadeada *Lista){
129     return (Lista->primeiro == Lista->ultimo); //se o primeiro é igual ao último significa que não há nada entre os dois
130 }
131
132 void InsereListaEncadeadaUltimo(ListaEncadeada *Lista, Funcionario *funcionario){//insere no último
133     Lista->ultimo->prox = new Celula; //o apontador prox do apontador ultimo, recebe uma nova célula
134     Lista->ultimo = Lista->ultimo->prox; //o apontador ultimo, recebe o apontador da célula dentro dele já alterado
135     Lista->ultimo->item = funcionario; //aloca o funcionario na célula
136     Lista->ultimo->prox = NULL; //o apontador prox do apontador ultimo, recebe null, após as trocas
137     Lista->tamanho++; //lista aumenta de tamanho
138 }
139 //You, há 21 horas • conclusão do código do TP
140 void ImprimeFuncionario(Funcionario *funcionario){
141     cout << "ID: " << funcionario->id << endl;
142     cout << "Nome: " << funcionario->nome << endl;
143     cout << "Endereço: " << funcionario->endereco << endl;
144     cout << "Número de Dependentes: " << funcionario->dependentes << endl;
145     cout << "\nLista de Projetos: " << endl;
146     ImprimeListaSequencial(&funcionario->projetos); //lista de projetos dele
147 }
148
149 Funcionario *PesquisaFuncionario(ListaEncadeada *Lista, int id){
150     Apontador aux = Lista->primeiro->prox;
151     while (aux != NULL){
152         if (aux->item->id == id) return aux->item;
153         aux = aux->prox; //percorrer o while
154     }
155     return NULL; // Retorna NULL caso o funcionário procurado não esteja na lista
156 }
157
158 void RemoveFuncionario(ListaEncadeada *Lista, Funcionario funcionario){
159     if (!ListaEncadeadaEstaVazia(Lista)){
160         Apontador aux, anterior, x;
161         x = Lista->primeiro;
162         while (x != NULL){
163             if (x->prox->item->id == funcionario.id){
164                 anterior = x; //salva o anterior a x, para ser passada para a variável auxiliar(aux)
165             }
166             x = x->prox; //incremento do laço para encontrar o funcionario desejado
167         }
168         aux = anterior->prox; //aux recebe o próximo valor após o anterior(último valor)
169         anterior->prox = aux->prox; //anterior->prox recebe o último
170         delete aux; // desloca o auxiliar da memória
171         Lista->tamanho--; //diminui da lista
172         AtualizaUltimo(Lista); //atualiza após a remoção
173     }
174 }
175
176 void AtualizaUltimo(ListaEncadeada *Lista){
177     Apontador aux, anterior;
178     aux = Lista->primeiro;
179     anterior = aux;
180     while (aux != NULL){
181         anterior = aux; //salva o anterior ao aux->prox na variável anterior
182         aux = aux->prox; //incrementa o laço
183     }
184     Lista->ultimo = anterior;
185 }
186
187 int TamanhoListaEncadeada(ListaEncadeada *Lista){
188     return Lista->tamanho;
189 }
```



## 5.2. APÊNDICE B – Sistema.hpp



The image shows a code editor with a dark theme. The top menu bar includes 'Arquivo', 'Editar', 'Seleção', 'Ver', 'Acessar', 'Executar', 'Terminal', and 'Ajuda'. The title bar of the editor window says 'h++ sistema.hpp M X'. The code is written in C++ and defines various data structures and constants for a system. It includes headers for `<iostream>`, `<fstream>`, and `<windows.h>`. Constants defined include `fileName` as 'listaDeFuncionarios.bin', `OPCAO_SAIDA` as 8, and `MAX_TAM` as 5. The code defines a `Projeto` struct with fields for `id`, `nome`, and `horas`. It also defines a `ListaSequencial` struct with an array of `Projeto` items, a `tamanho` field, and a `listaSequencialCriada` boolean. A `Funcionario` struct is defined with fields for `id`, `nome`, `endereco`, `dependentes`, and a pointer to a `ListaSequencial` of projects. A `Celula` struct is defined with a `Funcionario` item and a pointer to the next `Celula`. Finally, a `ListaEncadeada` struct is defined with pointers for the first and last nodes, a `tamanho` field, and a `listaEncadeadaCriada` boolean.

```
1  #ifndef SISTEMA_H
2  #define SISTEMA_H
3
4  #include <iostream>
5  #include <fstream>
6  #include <windows.h>
7  #define fileName "listaDeFuncionarios.bin"
8  #define OPCA_O_SAIDA 8
9  #define MAX_TAM 5 //tamanho máximo de projetos
10
11 using namespace std;
12
13 // LISTA SEQUENCIAL
14 You, há 21 horas | 1 author (You)
15 typedef struct Projeto{
16     int id; // código do projeto
17     char nome[30]; // nome do projeto
18     int horas; // número de horas trabalhadas no projeto
19 };
20
21 You, há 21 horas | 1 author (You)
22 typedef struct ListaSequencial{
23     Projeto item[MAX_TAM]; //vetor de tamanho máximo de projetos
24     int tamanho; //tamanho(que sempre é menor ou igual a 5)
25     bool listaSequencialCriada = false; //variavel booleana para criação de listas
26 };
27
28 // LISTA ENCADEADA
29 You, há 21 horas | 1 author (You)
30 typedef struct Funcionario{
31     int id; //id do funcionário
32     char nome[100]; //nome do funcionário
33     char endereco[40]; //endereço do funcionário
34     int dependentes; //número de dependentes do funcionário
35     ListaSequencial projetos; //Lista de Projetos alocados em um campo de funcionário
36 };
37
38 typedef struct Celula *Apontador; // "construtor" da célula
39
40 You, há 21 horas | 1 author (You)
41 typedef struct Celula{
42     Funcionario *item; //funcionario que é colocado dentro da célula
43     Apontador prox; //Apontador para a próxima célula
44 };
45
46 You, há 21 horas | 1 author (You)
47 typedef struct ListaEncadeada{ //Lista de encadeada
48     Apontador primeiro; //apontador da primeira posição
49     Apontador ultimo; //apontador da última posição
50     int tamanho = 0; //tamanho da lista
51     bool listaEncadeadaCriada = false; //variavel booleana para indicar se a lista foi criada ou não
52 };
53
```

```

50 // FUNÇÕES DA LISTA SEQUENCIAL
51 void CriaListaSequencialVazia(ListaSequencial *lista);
52 bool ListaSequencialEstaVazia(ListaSequencial *lista);
53 bool ListaSequencialEstaCheia(ListaSequencial *lista);
54 bool InsereProjeto(ListaSequencial *lista, Projeto projeto);
55 void ImprimeProjeto(Projeto projeto);
56 void ImprimeListaSequencial(ListaSequencial *lista);
57 Projeto *PesquisaProjetoPorID(ListaSequencial *lista, int id);
58 bool RemoveProjetoPorID(ListaSequencial *lista, int id);
59 bool RemoveProjeto(ListaSequencial *lista, Projeto projeto);
60 int IndiceDeProjeto(ListaSequencial *lista, Projeto projeto);
61 int TamanhoListaSequencial(ListaSequencial *lista);
62
63 // FUNÇÕES DA LISTA ENCADEADA
64 void CriaListaEncadeadaVazia(ListaEncadeada *lista);
65 bool ListaEncadeadaEstaVazia(ListaEncadeada *lista);
66 void InsereListaEncadeadaUltimo(ListaEncadeada *lista, Funcionario *funcionario);
67 void ImprimeFuncionario(Funcionario *funcionario);
68 Funcionario *PesquisaFuncionario(ListaEncadeada *lista, int id);
69 void RemoveFuncionario(ListaEncadeada *lista, Funcionario funcionario);
70 void AtualizaUltimo(ListaEncadeada *lista);
71 int TamanhoListaEncadeada(ListaEncadeada *lista);
72
73 // FUNÇÕES DO SISTEMA
74 void menu();
75 void incluiNovoFuncionario(ListaEncadeada *lista);
76 Funcionario *criaFuncionario();
77 void incluiNovosProjetos(ListaEncadeada *lista);
78 Funcionario *FuncionarioPorID(ListaEncadeada *lista);
79 void incluiNovoProjeto(Funcionario *funcionario);
80 void criaProjeto(ListaSequencial *lista, Projeto *projeto);
81 void excluiProjetos(ListaEncadeada *lista);
82 void excluiFuncionariosSemProjetos(ListaEncadeada *lista);
83 void ImprimeListaFunc(ListaEncadeada *lista);
84 void consultaFuncionario(ListaEncadeada *lista);
85 int totalHorasSemanais(ListaSequencial lista);
86 double calculaSalarioBruto(Funcionario *funcionario);
87 void imprimeContraCheque(ListaEncadeada *lista);
88
89
90 // FUNÇÕES DE MANIPULAÇÃO DE ARQUIVOS
91 bool carregaArquivo(ListaEncadeada *lista);
92 bool salvaArquivo(ListaEncadeada *lista);
93

```

### 5.3. APÊNDICE C – Manipulação de arquivos

```

411 //////////////////////////////////// MANIPULAÇÃO DE ARQUIVOS ////////////////////////////////////
412 bool carregaArquivo(ListaEncadeada *lista){
413     ifstream arquivo;
414     arquivo.open(fileName, ifstream::binary); // Abre o arquivo em modo binario, ou cria se não existir
415
416     if (arquivo.fail()){ // Se falhar
417         arquivo.clear(); // Fecha o arquivo
418         return false;
419     }
420     while (arquivo.peek() != ifstream::traits_type::eof()){ // Enquanto o arquivo não for vazio
421         Funcionario *funcionario = new Funcionario;
422         arquivo.read((char *)funcionario, sizeof(Funcionario)); // Lê a Struct dos funcionarios completos
423         InsereListaEncadeadaUltimo(lista, funcionario); // Insere o projeto que pegou do arquivo na lista encadeada
424     }
425     arquivo.clear();
426     return true;
427 }
428
429 bool salvaArquivo(ListaEncadeada *lista){
430     ofstream arquivo;
431     arquivo.open(fileName, ofstream::trunc | ofstream::binary); // Abre arquivo em modo trunc pra deixa-lo vazio e em modo binario
432
433     if (arquivo.fail()){
434         arquivo.clear();
435         return false;
436     }
437
438     Apontador aux = lista->primeiro->prox;
439     while (aux != NULL){
440         Funcionario *funcionario = aux->item;
441         arquivo.write((char *)funcionario, sizeof(Funcionario)); // Escreve no arquivo a Struct do funcionario completa
442         delete funcionario;
443         aux = aux->prox;
444     }
445     return true;
446 }

```

#### 5.4. APÊNDICE D – Menu

```
You, há 21 horas | 1 author (You)
1  #include "sistema.hpp"
2
3  void menu(){
4      system("cls");
5      cout << "*****" << endl;
6      cout << "*" << endl;
7      cout << "      Departamento Pessoal" << endl;
8      cout << "*" << endl;
9      cout << "*****" << endl;
10     cout << "*" << endl;
11     cout << "  1 - INCLUIR NOVO FUNCIONÁRIO" << endl;
12     cout << "*" << endl;
13     cout << "  2 - INCLUIR NOVOS PROJETOS" << endl;
14     cout << "*" << endl;
15     cout << "  3 - EXCLUIR PROJETOS" << endl;
16     cout << "*" << endl;
17     cout << "  4 - EXCLUIR FUNCIONÁRIOS SEM PROJETOS" << endl;
18     cout << "*" << endl;
19     cout << "  5 - CONSULTAR FUNCIONÁRIO" << endl;
20     cout << "*" << endl;
21     cout << "  6 - IMPRIMIR CONTRA-CHEQUE" << endl;
22     cout << "*" << endl;
23     cout << "  7 - LISTA DE FUNCIONÁRIOS" << endl;
24     cout << "*" << endl;
25     cout << "  8 - SAIR" << endl;
26     cout << "*" << endl;
27     cout << "*****" << endl;
28 }
```



```
C:\WINDOWS\system32\cmd.exe
*****
*
*      Departamento Pessoal
*
*****
*
*  1 - INCLUIR NOVO FUNCIONÁRIO
*
*  2 - INCLUIR NOVOS PROJETOS
*
*  3 - EXCLUIR PROJETOS
*
*  4 - EXCLUIR FUNCIONÁRIOS SEM PROJETOS
*
*  5 - CONSULTAR FUNCIONÁRIO
*
*  6 - IMPRIMIR CONTRA-CHEQUE
*
*  7 - LISTA DE FUNCIONÁRIOS
*
*  8 - SAIR
*
*****
Opção:
```

## 5.5. APÊNDICE E – Função de Inserção de funcionário

```
192 void incluiNovoFuncionario(ListaEncadeada *lista){
193     Funcionario *funcionario = criaFuncionario();
194
195     if (PesquisaFuncionario(lista, funcionario->id) == NULL){
196         InserirListaEncadeadaUltimo(lista, funcionario); //insere no fim da lista, isso vem lá da listaEncadeada.cpp
197         cout << "\nFuncionário incluído com Sucesso!\n" << endl;
198         int opcao;
199         do{
200             cout << "Deseja adicionar projetos ao funcionário?" << endl;
201             cout << "1 - Sim" << endl;
202             cout << "2 - Não" << endl;
203             cin >> opcao;
204             if (opcao == 1){
205                 system("cls");
206                 int quantidade;
207                 cout << "Quantidade de projetos que deseja adicionar: ";
208                 cin >> quantidade;
209                 for (int i = 0; i < quantidade; i++) incluiNovoProjeto(funcionario);
210             }
211             system("cls");
212         } while (opcao != 1 && opcao != 2);
213     }
214     else{ //caso exista um funcionario com o mesmo ID, que quero inserir
215         cout << "\nJá existe funcionário com ID: " << funcionario->id << endl;
216         cout << "Adicione novamente com outro ID" << endl;
217         delete funcionario; // desloca o funcionario da memória, já que ele não será usado
218         Sleep(1000);
219     }
220 }
```

## 5.6. APÊNDICE F – Função de retorno de funcionário, para inserção

```
222 Funcionario *criaFuncionario(){
223     Funcionario *funcionario = new Funcionario;
224     cout << "Cadastro Funcionário" << endl;
225     cout << "ID: ";
226     cin >> funcionario->id;
227     cin.ignore();
228     cout << "Nome: ";
229     cin.getline(funcionario->nome, 100);
230     cout << "Endereço: ";
231     cin.getline(funcionario->endereco, 40);
232     cout << "Número de Dependentes: ";
233     cin >> funcionario->dependentes;
234     CriarListaSequencialVazia(&funcionario->projetos);
235     return funcionario;
236 }
```

## 5.7. APÊNDICE G – Inclusão de Projeto

```
237 void incluiNovosProjetos(ListaEncadeada *Lista){
238     if (!ListaEncadeadaEstaVazia(Lista)){
239         Funcionario *funcionario = FuncionarioPorID(Lista);
240
241         if (funcionario != NULL){ // Só adiciona se o funcionario existir
242             int opcao = 1;
243             do{
244                 if (opcao == 1) incluiNovoProjeto(funcionario);
245                 system("cls");
246                 cout << "Deseja adicionar outro projeto para esse funcionário?\n";
247                 cout << "1-Sim" << endl;
248                 cout << "2-Não" << endl;
249                 cin >> opcao;
250                 system("cls");
251             }while (opcao != 2);
252         }
253         else{
254             cout << "\nNão existe funcionário com esse id";
255             Sleep(1000);
256         }
257     }
258     else{
259         cout << "Erro: Funcionário não cadastrado!" << endl;
260         Sleep(1000);
261     }
262 }
263
264 void criaProjeto(ListaSequencial *Lista, Projeto *projeto){
265     cout << "Cadastro de Novo Projeto" << endl;
266     cout << "ID: ";
267     cin >> projeto->id;
268     cin.ignore();
269     cout << "Nome: ";
270     cin.getline(projeto->nome, 30);
271     cout << "Horas Trabalhadas: ";
272     cin >> projeto->horas;
273     cin.ignore();
274 }
275
276 void incluiNovoProjeto(Funcionario *funcionario){
277     Projeto projeto;
278     criaProjeto(&funcionario->projetos, &projeto);
279     if (PesquisaProjetoPorID(&funcionario->projetos, projeto.id) == NULL){ // TRUE se o projeto não está na lista, FALSE do contrário
280         bool inseridoComSucesso = InsereProjeto(&funcionario->projetos, projeto);
281         if (inseridoComSucesso) cout << "\nProjeto inserido com sucesso" << endl;
282         else cout << "\nEsse funcionário atingiu o máximo de projetos possíveis";
283     }
284     else cout << "\nJá existe Projeto com ID: " << projeto.id << " nesse funcionário.\nAdicione novamente com outro ID" << endl;
285     Sleep(1000);
286 }
287 }
```

## 5.8. APÊNDICE H – Pesquisa funcionário por ID

```
289 Funcionario *FuncionarioPorID(ListaEncadeada *Lista){
290     int idFuncionario;
291     cout << "ID do Funcionário: ";
292     cin >> idFuncionario;
293     Funcionario *funcionario = PesquisaFuncionario(Lista, idFuncionario);
294     return funcionario;
295 }
296 }
```

## 5.9. APÊNDICE I – Exclui Projetos

```
297 void excluiProjetos(ListaEncadeada *lista){
298     if (!ListaEncadeadaEstaVazia(lista)){
299         Funcionario *funcionario = FuncionarioPorID(lista); // Retorna ou o funcionário ou NULL
300
301         if (funcionario != NULL){
302             int idProjeto;
303             cout << "ID do Projeto: ";
304             cin >> idProjeto;
305
306             Projeto *projeto = PesquisaProjetoPorID(&funcionario->projetos, idProjeto);
307             if (projeto == NULL) cout << "\nProjeto com id igual à " << idProjeto << " não encontrado" << endl;
308             else{
309                 bool removeuComSucesso = RemoveProjeto(&funcionario->projetos, *projeto);
310                 if (removeuComSucesso) cout << "\nProjeto com id: " << idProjeto << " foi removido com sucesso" << endl;
311                 else cout << "\nErro: Não foi possível excluir projeto com id: " << idProjeto << endl;
312             }
313         }
314         else cout << "\nNão existe funcionário com esse Id" << endl;
315     }
316     else cout << "Erro: Nenhum Funcionário cadastrado na lista" << endl;
317     Sleep(2000);
318 }
319 }
```

## 5.10. APÊNDICE J – Exclui funcionários sem Projeto

```
320 void excluiFuncionariosSemProjetos(ListaEncadeada *lista){
321     if (!ListaEncadeadaEstaVazia(lista)){
322         Apontador aux = lista->primeiro->prox;
323         Apontador anterior = aux;
324         while (aux != NULL && PesquisaFuncionario(lista, aux->item->id) != NULL){
325             if (TamanhoListaSequencial(&aux->item->projetos) == 0){
326                 RemoveFuncionario(lista, *aux->item);
327                 aux = anterior;
328             }
329             anterior = aux;
330             aux = anterior->prox;
331         }
332         cout << "Todos os funcionário sem projetos foram apagados." << endl;
333         Sleep(2000);
334     }
335     else cout << "Erro: Nenhum Funcionário cadastrado" << endl;
336     Sleep(2000);
337 }
```

## 5.11. APÊNDICE K – Impressão de Funcionário

```
338
339 void ImprimeListaFunc(ListaEncadeada *lista){
340     if(!ListaEncadeadaEstaVazia(lista)){ //verifica se o endereço da lista é true
341         cout << "Lista Vazia";
342         Sleep(1000);
343         return ;
344     }
345     Apontador aux;
346     aux = lista->primeiro->prox; //recebe o apontador de próximo, depois do primeiro
347     while(aux != NULL){ //enquanto não for nulo, ele imprime
348         ImprimeFuncionario(aux->item);
349         aux = aux->prox; //aux sendo 'incrementado'
350     }
351     Sleep(5000);
352 }
353
354 void consultaFuncionario(ListaEncadeada *lista){
355     if (!ListaEncadeadaEstaVazia(lista)){
356         Funcionario *funcionario = FuncionarioPorID(lista);
357
358         if (funcionario != NULL) ImprimeFuncionario(funcionario);
359         else cout << "\nNão existe funcionário com esse Id";
360     }
361     else cout << "Erro: Nenhum Funcionário cadastrado" << endl;
362     Sleep(5000);
363 }
364
```

## 5.12. APÊNDICE L – Cálculo de Salário Bruto

```
364
365 int totalHorasSemanais(ListaSequencial lista){
366     int totalHoras = 0;
367     for (int i = 0; i < lista.tamanho; i++){
368         Projeto projeto = lista.item[i];
369         totalHoras += projeto.horas;
370     }
371     return totalHoras;
372 }
373
374 double calculaSalarioBruto(Funcionario *funcionario){
375     double salarioBruto = 0;
376     int totalHoras = totalHorasSemanais(funcionario->projetos);
377     salarioBruto = (totalHoras * 45) + (35 * funcionario->dependentes);
378     return salarioBruto;
379 }
380
```

```
381 void imprimeContraCheque(ListaEncadeada *lista){
382     if (!ListaEncadeadaEstaVazia(lista)){
383         Apontador aux;
384         aux = lista->primeiro->prox;
385
386         while (aux != NULL){
387             Funcionario *funcionario = aux->item;
388             int totalHoras;
389             double descontoINSS, salarioBruto, descontoIR, salarioLiquido;
390             salarioBruto = calculaSalarioBruto(funcionario);
391             descontoINSS = (8.5 / 100.0) * salarioBruto;
392             descontoIR = (15.0 / 100.0) * salarioBruto;
393             salarioLiquido = salarioBruto - descontoINSS - descontoIR;
394             totalHoras = totalHorasSemanais(funcionario->projetos);
395             //impressão
396             cout << "Id: " << funcionario->id;
397             cout << "\nNome: " << funcionario->nome;
398             cout << "\nTotal de Horas Semanais: " << totalHoras;
399             cout << "\nSalário Bruto: " << salarioBruto;
400             cout << "\nDesconto do INSS: " << descontoINSS;
401             cout << "\nDesconto de Imposto de renda: " << descontoIR;
402             cout << "\nSalário líquido: " << salarioLiquido;
403             cout << "\n";
404             aux = aux->prox;
405         }
406     }
407     else cout << "Erro: Nenhum Funcionário cadastrado" << endl;
408     Sleep(2000);
409 }
410
```