UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

DEPARTAMENTO DE INFORMÁTICA

ESTRUTURA DE DADOS I

Afonso Salvador de Magalhães

Thamya Vieira Hashimoto Donadia

EDCare: Monitoramento de Idosos

Trabalho I

Vitória, ES

2022

Introdução

O EDCare consiste em um sistema de monitoramento de idosos, em que estes, possuem uma rede de amigos e de cuidadores. O projeto manipula três tipos de listas encadeadas diferentes: lista de amigos, idosos e cuidadores. A partir do arquivo “apoio.txt” e “cuidadores.txt”, o programa estrutura a lista com todos os idosos e cuidadores, bem como as relações de amizade e de cuidado de cada paciente.

Desse modo, realiza-se a leitura dos dados sensoriais do idoso e do cuidador, de forma a analisar as necessidades de cada paciente, as quais são escritas nos arquivos de saída. As funcionalidades principais do programa foram definidas no TAD “rede”, as quais realizam todo o gerenciamento do programa e, portanto, são chamadas no “edcare.c”, que contém a função *main*. Em conclusão, esse TAD se encontra como ponte entre o acesso às informações a serem analisadas e as ferramentas que foram utilizadas para tal análise.

Implementação

A priori, para a estruturação do sistema, foram definidos 5 tipos de dados diferentes: struct idoso, struct cuidador, struct listaidoso, struct listaamigo e struct listacuidador, os quais são especificados a seguir:

1. *struct idoso*

O tipo “Idoso” armazena as informações pessoais e sensoriais do idoso, o arquivo de leitura e de saída, bem como a sua lista de amigos e de cuidadores. Para esse tipo de dado, foram definidas funcionalidades básicas, tais como inicialização, get’s, set’s e destruição do tipo.

2. *struct cuidador*

O tipo “Cuidador” armazena as informações pessoais e sensoriais do cuidador do paciente, assim como seu arquivo de leitura de sensores. Para esse tipo de dado, foram definidas funcionalidades básicas, tais como inicialização, get’s, set’s e destruição do tipo.

3. *struct listaidoso*

O tipo “ListaIdoso” consiste em uma lista duplamente encadeada com sentinela, a qual armazena todos os idosos do EDCare. Assim, para esse tipo de dado, foram implementadas funcionalidades elementares de listas, como inicialização, inserção e retirada de idosos e destruição da estrutura.

4. *struct listacuidador*

O tipo “ListaCuidador” consiste em uma lista duplamente encadeada com sentinela, que contém todos os cuidadores do EDCare. Assim, para esse tipo de dado, foram implementadas funcionalidades elementares de listas, como inicialização, inserção e destruição da estrutura. Não há uma função de retirada de um cuidador específico da lista já que, dentro do funcionamento do programa, não houve a necessidade de retirar apenas um cuidador da lista. Ademais, foi definida uma função que retorna o ponteiro do cuidador mais próximo de um idoso, baseada nas leituras sensoriadas de ambos.

5. *struct listaamigo*

O tipo “ListaAmigo” consiste em uma lista duplamente encadeada com sentinela, que está armazenada em cada idoso do sistema, e contém todos os amigos de cada idoso. Assim, para esse tipo de dado, foram implementadas funcionalidades elementares de listas, como inicialização, inserção e retirada de amigos (idosos) e destruição da estrutura. Ademais, foi definida uma função que retorna o ponteiro do amigo mais próximo de um idoso, desde que o idoso esteja em condições de ser notificado (vivo), baseada nas leituras sensoriadas de ambos.

Vale ressaltar que a escolha pela lista duplamente encadeada com sentinela foi pautada na priorização, por parte da dupla, em organização, segurança estrutural, bem como na facilidade de navegação pelos elementos da lista. Desse modo, a arquitetura do EDCare ficou definida da seguinte forma:

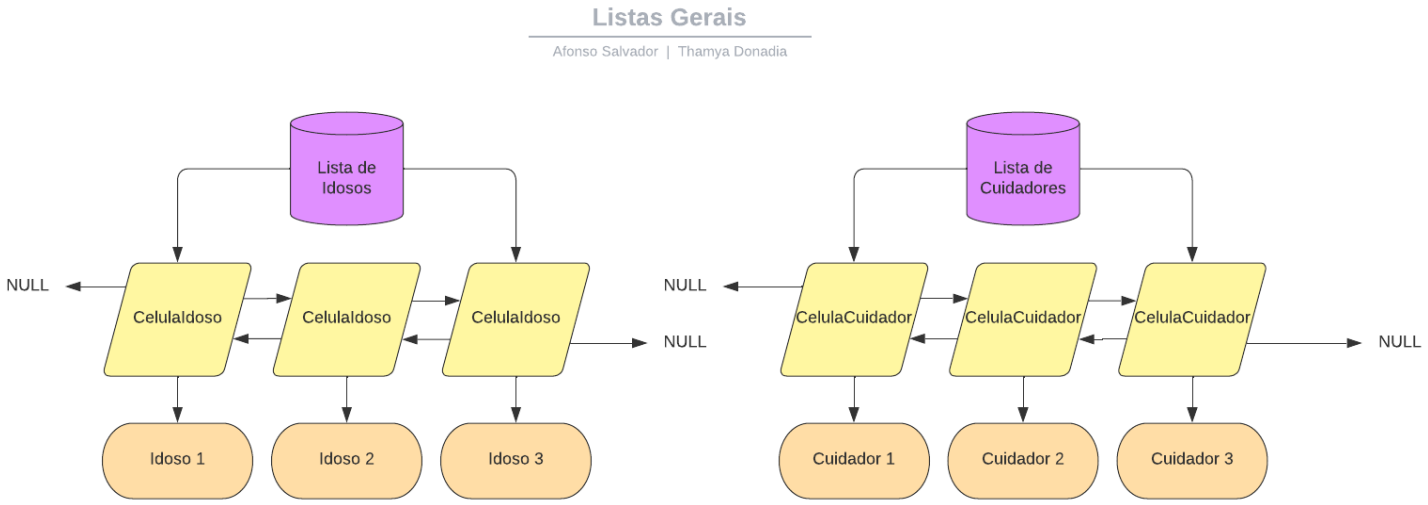

Figura 1 - Estrutura das listas de idosos e de cuidadores (“ListaIdoso” e “ListaCuidador”)

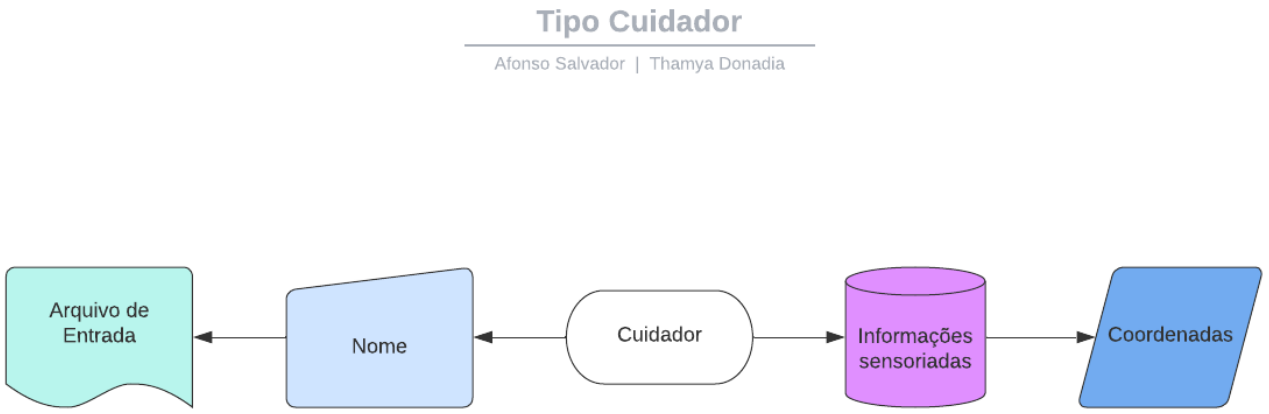

Figura 2 - Estrutura do tipo cuidador (“Cuidador”)

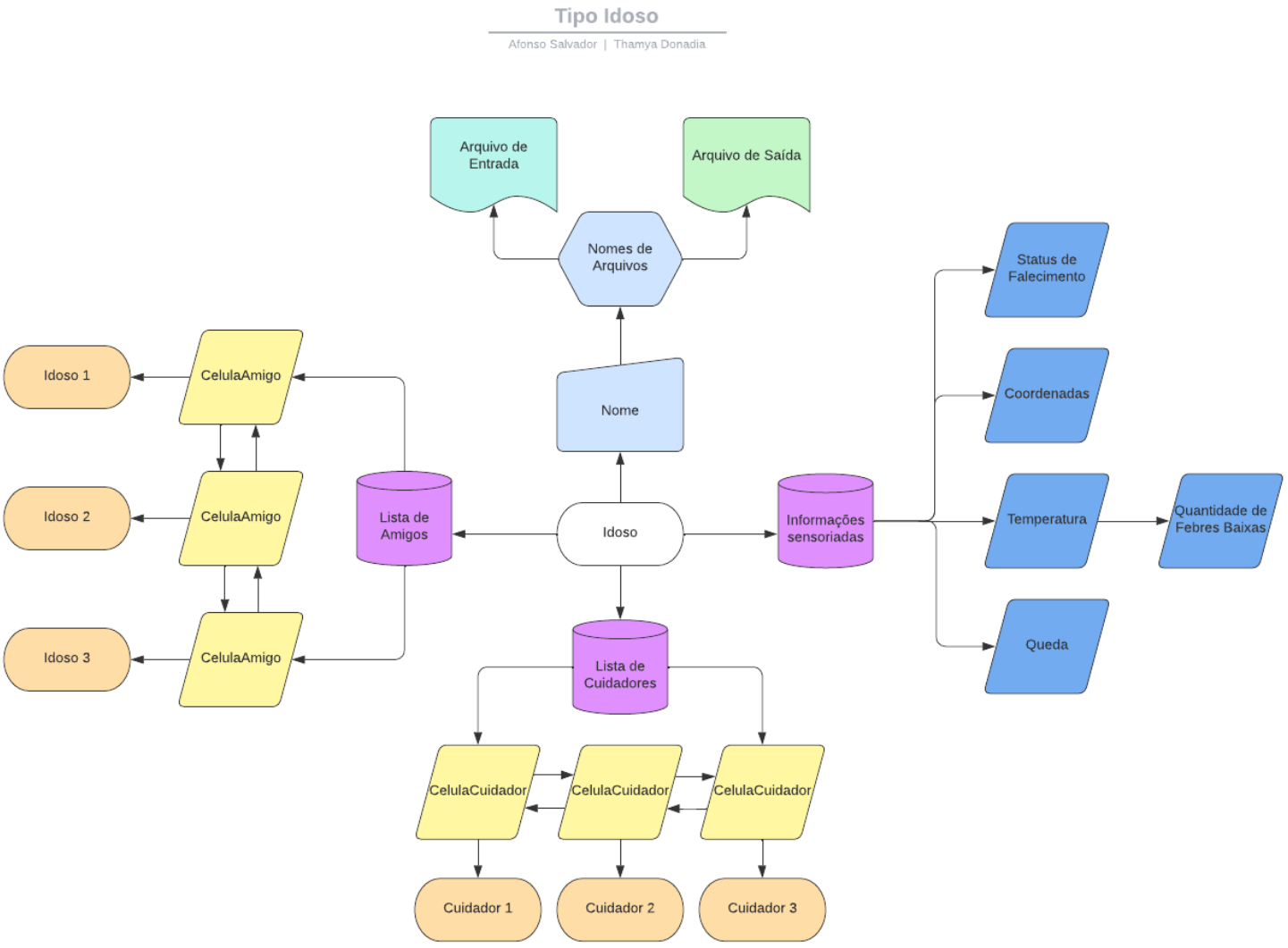

Figura 3 - Estrutura do tipo idoso (“Idoso”)

O sistema foi projetado para que as funções principais, tais como a atualização das informações sensoriais dos idosos e cuidadores, a partir da leitura dos arquivos de entrada, o monitoramento dos idosos e o encerramento do sistema, fossem definidas dentro dos arquivos “rede.h” e “rede.c”. Desse modo, foi possível manter o código modularizado e as estruturas de dados protegidas, uma vez que no "edcare.c" , o qual contém a *main*, apenas são chamadas as funções que gerenciam o EDCare.

No arquivo “edcare.c”, como apresentado na figura abaixo, é possível observar as funções, pertencentes ao TAD “rede”, que gerenciam o sistema:

```c
int main(int argc, char const *argv[]){
    int numLeituras;

    if(argc<=1){
        printf("ERRO: NAO FOI INFORMADO O NUMERO DE LEITURAS");
        return 1;
    }

    //Coleta número de leituras
    sscanf(argv[1], "%d", &numLeituras);

    //Trata o arquvio Apoio.txt
    ListaIdoso* listaGeralIdosos = lerRedeApoio();
    //Trata o arquivo Cuidadores.txt
    ListaCuidadores* listaGeralCuidadores = lerRedeCuidado(listaGeralIdosos);

    //Trata a coleta de informações e os resultados de tais inforamções
    monitoramentoGeral(listaGeralIdosos, listaGeralCuidadores, numLeituras);

    //Destroi listas gerais e seus elementos
    finalizaEdCare(listaGeralIdosos, listaGeralCuidadores);

    return 0;
}
```

Figura 4 - Funcionamento geral do EDCare (edcare.c)

Dessa forma, o sistema tem a seguinte ordem de funcionamento:

1. Leitura e tratamento do número de leituras de cada arquivo, fornecido pelo parâmetro “char const* argv[]”;
2. Leitura do arquivo “apoio.txt” e criação da lista de geral de pacientes, bem como da rede de apoio de cada idoso, isto é, as relações de amizade entre cada um;
3. Leitura do arquivo “cuidadores.txt” e criação da lista geral de cuidadores do sistema, assim como da rede de cuidado dos idosos, isto é, a estruturação dos cuidadores de cada idoso;
4. Monitoramento geral dos idosos, a partir da leitura dos arquivos de cada idoso e cada cuidador (uma linha por vez de todos). Assim, foram utilizadas funções para interpretação dos dados sensoriais dos idosos e dos cuidadores, com o intuito de determinar as medidas a serem tomadas em caso de queda, febre alta e febre baixa, as quais foram armazenadas no

arquivo de saída de cada paciente. Ao final, a função finaliza todos os arquivos utilizados;

5. Finalização do sistema por meio da destruição das listas de idosos e cuidadores, bem como dos elementos que compoẽm as listas, de forma a liberar toda a memória que foi alocada para o funcionamento do programa.

Em conclusão, o EDCare foi estruturado como mostra o fluxograma abaixo:

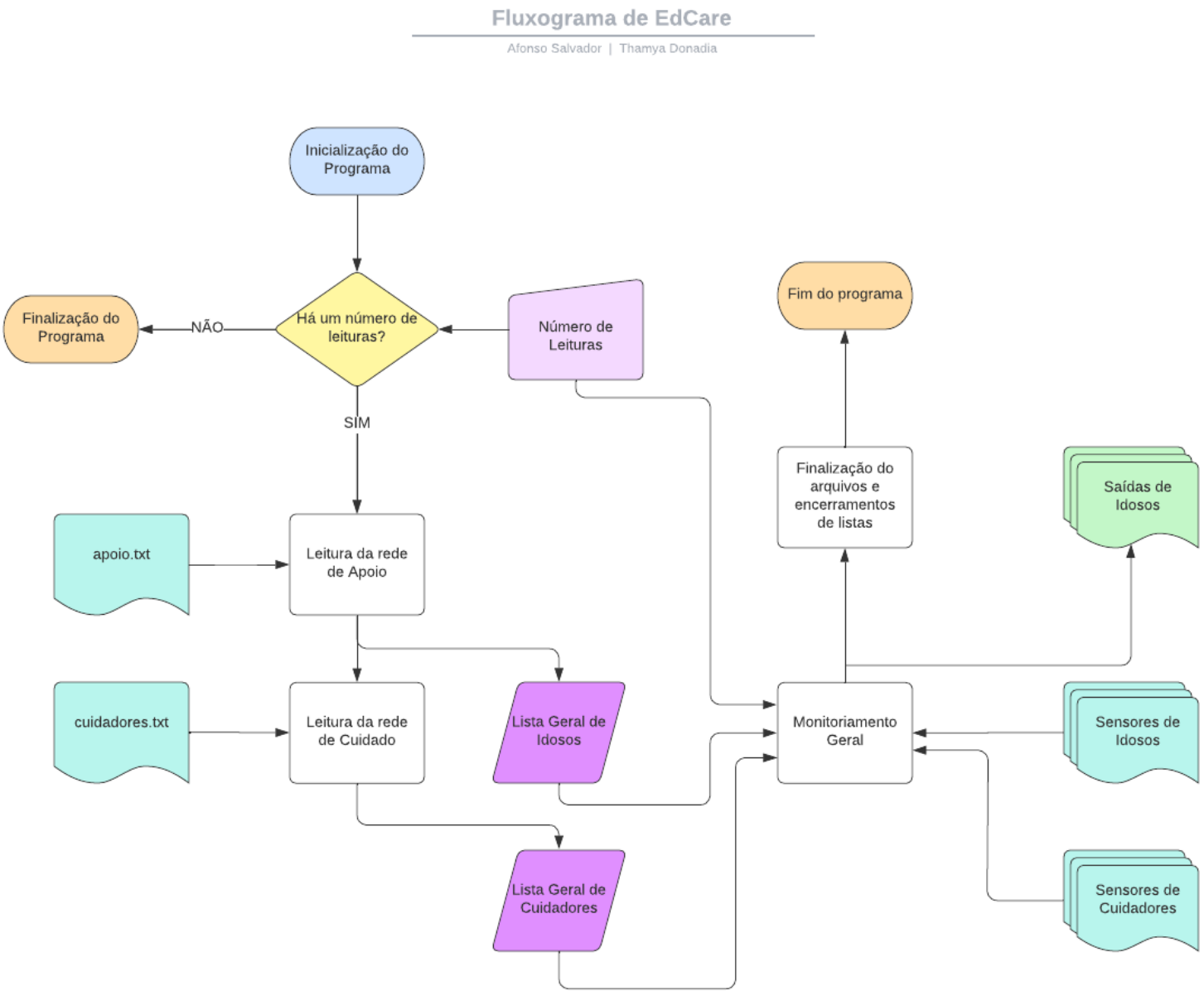

Figura 5 - Fluxograma que apresenta o funcionamento geral do EDCare

Vale ressaltar que, no tocante ao monitoramento dos idosos, o programa funciona por rodadas, isto é, por linha de arquivo. Desse modo, ocorre a leitura linha a linha de cada arquivo, e, após cada rodada, a atualização das informações dos idosos e cuidadores, informadas pelos sensores. A partir da informação de cada rodada, é feito o monitoramento do idoso e, por conseguinte, determinadas as ações a serem tomadas para o cuidado do paciente.

Além disso, ainda na análise das informações dos idosos, em função dos outputs dos casos de teste, foi definido que um amigo só está indisponível para ser chamado, caso ele tenha falecido. Por fim, outra decisão tomada pela dupla, foi o tratamento do *whitespace* ao final de algumas linhas dos casos de teste.

Conclusão

A elaboração do sistema foi pautada em uma organização prévia, em que foram definidas as estruturas e os tipos de dados que seriam utilizados. Ademais, em uma visão macro, foi estabelecida a arquitetura geral do projeto, isto é, quais eram as funcionalidades exigidas e como elas seriam implementadas. Desse modo, a partir dessa documentação inicial, o processo de escrever o código foi guiado, e, portanto, fluiu mais rapidamente.

No decorrer do desenvolvimento do trabalho, foram encontradas algumas dificuldades no tocante à implementação do tipo “idoso”, visto que o compilador apontava que existiam tipos que estavam sendo utilizados sem estarem previamente definidos. Isso aconteceu em virtude das listas de amigos e de cuidadores que foram implementadas dentro de cada estrutura do idoso, isto é, o TAD estava sendo incluído antes da definição do tipo. Como medida de contorno da situação, a dupla definiu os tipos de dados antes de incluir os arquivos “.h”.

Bibliografia

NULL-TERMINATED byte strings: String manipulation. *In*: C Programming Language. [*S. l.*]. Disponível em: https://devdocs.io/c/string/byte. Acesso em: 16 jun. 2022.

STRTOK, strtok_s. *In*: C Programming Language. [*S. l.*]. Disponível em: https://devdocs.io/c/string/byte/strtok. Acesso em: 16 jun. 2022.

C / File input/output. *In*: C Programming Language. [*S. l.*]. Disponível em: https://devdocs.io/c-file-input-output/. Acesso em: 16 jun. 2022.