UNIVERSIDADE PAULISTA - UNIP EaD

Projeto Integrado Multidiciplinar

Curso Superior em Tecnologia em
Análise e Desenvolvimento de Sistemas

NICOLAS DOS SANTOS LIRA - 2251695

SISTEMA DE CONTROLE DE PACIENTES COM COVID-19

Projeto desenvolvido em C

Ferraz de Vasconcelos, São Paulo
2022

NICOLAS DOS SANTOS LIRA - 2251695

SISTEMA DE CONTROLE DE PACIENTES COM COVID-19

Projeto desenvolvido em C

Projeto Integrado Multidisciplinar em
Análise e Desenvolvimento de Sistemas

Projeto Integrado Multidisciplinar para obtenção do título de tecnólogo em Análise e Desenvolvimento de Sistemas, apresentado à Universidade Paulista - UNIP EaD.

Orientador(a): Profa. Dra. Vanessa Lessa.

Ferraz de vasconcelos, São Paulo
2022

RESUMO

Como proposto no material disponível pela orientadora do projeto, foi desenvolvido um projeto usando a IDE Code::Block, usando da linguagem de programação C, sobre o *template* de projeto *“console application”*.

Para o desenvolvido deste, foram utilizados os conhecimentos adquiridos nas disciplinas Linguagens e técnicas de programação,e Engenharia de Software I, do 2° semestre do curso Análise e desenvolvimento de sistemas. Além destas, foi imposto o conceito de outras disciplinas do curso, e conhecimento adquirido a base de pesquisas para complemento do projeto.

Durante o desenvolvimento do mesmo, foram realizados diversos testes manuais, a fim de resolver eventuais problemas que surgiram durante a codificação do programa.

Palavras-chaves: COVID-19, Programação em C, Automação.

ABSTRACT

As proposed in the material available by the project advisor, a project was developed using the Code::Block IDE, using the C programming language, on the “console application” project template.

For the development of this, the knowledge acquired in the disciplines Languages and programming techniques, and Software Engineering I, of the 2nd semester of the Analysis and development of systems course were used. In addition to these, the concept of other disciplines of the course was imposed, and knowledge acquired based on research to complement the project.

During its development, several manual tests were carried out in order to solve any problems that arose during the coding of the program.

Key Words: COVID-19, programming in C, automation.

SUMÁRIO

1. INTRODUÇÃO

Em 31 de dezembro de 2019, a organização mundial(OMS) da saúde foi alertada sobre vários casos de doenças respiratórias na cidade de Wuhan, na China. Pouco tempo se passou, e a COVID-19, uma doença infecciosa causada pelo vírus SARS-CoV-2 se espalhou pelo mundo velozmente, fazendo com que a OMS caracterize a doença como uma pandemia.

A transmissão desta doença é feita pelo ar ou contato com secreção contaminada, e seus efeitos vão de leves a graves, sendo comum os sintomas críticos atingirem pessoas com comorbidade ou de idade elevada. Os sintomas leves da doença são semelhantes a uma gripe, sendo eles tosse, febre, cansaço e perda de paladar ou olfato, e seus sintomas mais severos sendo falta de ar, dificuldade para respirar, perda da fala, mobilidade ou confusão, e dores na região do tórax.

Os sintomas mais graves se mostraram letais em pessoas com comorbidade e em idosos, e por conta disso houve uma superlotação nos sistemas de saúde mundial, bem como milhares de mortes por conta desta doença e suas consequências.

02. ESTRUTURA DO PROGRAMA

Assim como proposto no manual do projeto Integrado multidisciplinar IV, o programa foi desenvolvido dentro da IDE Code::Block utilizando da linguagem C. Dentro da mesma, foi escolhido para elaboração deste o template de Aplicação em console, possibilitando uma aplicação leve que realize o cadastro do paciente que foi diagnosticado com o vírus, e como proposto no material, associando em um arquivo específico aqueles que tenham idade superior a 65 anos ou sejam portadores de alguma comorbidade.

Dentro do sistema há 2 funções de suma importância para seu pleno funcionamento, uma que realiza a verificação para login no sistema, que tem como parâmetros de entrada as variáveis user e a password, ambas do tipo strings. A outra função cria e escreve os dados do paciente em um arquivo, independente de sua idade e se possui comorbidade, e tem como parâmetros de entrada uma struct Paciente.

Para o cadastro do paciente, foi construído 3 estruturas de listas, uma denominada *Data* que contém 3 valores todos do tipo inteiro, e permite que seja inserido e trabalhado de forma mais prática datas, a outra chamada *Endereco* e possui todos os itens solicitado no material para a criação do endereço do paciente. E a terceira é a lista Paciente, ela tem como variáveis de sua estrutura 3 strings, nome, email e comodidade, o CPF e telefone como inteiros, data de nascimento e de diagnóstico são listas do tipo Data, e por fim a lista de Endereço.

Foi escolhido o desenvolvimento de listas para armazenar os dados do paciente por conta da facilidade no transporte da mesma quando utilizada como parâmetro e chamada da mesma, além de deixar o código mais visual.

03. FUNCIONAMENTO DO PROGRAMA

O sistema desenvolvido apresenta uma lógica simples, ao ser iniciado, estabelecemos uma variável global do main de tipo inteiro nomeada i, depois disso entramos em uma laço de repetição do while, a qual finaliza quando a variável i apresentar um valor diferente de 0.

Dentro desse laço, iniciamos duas strings, a user e o password, solicitamos a entrada para estas por meio do comando gets(), e com os valores inseridos atribuímos o valor da função Login para a variável i. Seguindo dentro do código, temos uma estrutura condicional, que se o valor de i for 0 apresenta uma mensagem de erro de login, pausa o sistema por meio do comando system(“pause”), e depois de uma interação do usuário limpa a tela com o system(“cls”). Se o valor de i for diferente de 0, o programa limpa a tela e sai do laço de repetição.

Assim que saímos do laço de repetição, a tela é novamente limpa pelo comando system(“cls”), e iniciamos o cadastro do paciente que contraiu o vírus. De início é inserido um texto e logo em seguida declaramos a lista pac de estrutura Paciente. Com a lista inicializada, incluímos os respectivos valores para os campos nome, CPF, telefone, Data Nascimento, endereço, email, e data de diagnóstico, por meio de gets para entrada das strings, e scanf(“%d”, lista.variável) para a entrada de inteiros.

Com a entrada destes dados, declaramos um char e coletamos um valor do tipo solicitando ao usuário indicar se o paciente possui comorbidade com S representando sim e N o não, após a entrada do usuário com a respectiva resposta, temos uma estrutura condicional que utiliza novamente do comando strcmp para comparar a variável comorbidade com a resposta positiva, e se a resposta for S ou s, exibe um texto solicitando para que o usuário indique a comorbidade do paciente, e atribui o que foi digitado no campo da lista respectivo,e por fim atribui o valor de valueComorbidade como 1.

Saindo desta estrutura, é chamada a função anexarArquivo, que tem como entrada a lista que acabamos de preencher. Após ter o arquivo criado e preenchido,

entramos em uma estrutura condicional que verifica se a idade do paciente é maior que 65, ou se ele possui comorbidade por meio da variável *valueComorbidade*, e caso alguma condicional seja verdadeira, abre, ou cria caso ele não exista, um arquivo chamado RISCO.txt, e adiciona no seu final o nome do paciente, data de nascimento, CEP, e Telefone, depois verifica se ele possui comorbidade, e caso possua incrementa a mesma. Após a escrita, o arquivo é fechado.

Assim, finalizamos o programa limpando a tela e exibindo uma mensagem dizendo que o cadastro foi realizado com sucesso, e finalizando o mesmo.

04. LISTAS E FUNÇÕES

04.1 Lista Data.

Esta lista foi desenvolvida para auxiliar e facilitar o desenvolvimento dos campos de data dentro da lista principal. Neste temos os inteiros dia, mês e ano, que além de deixar a entrada de dados muito mais intuitiva, nos permite fazer operações com seus elementos mais facilmente. Abaixo é apresentado a estrutura da lista, e sua utilização na lista principal.

```
typedef struct{
    int dia;
    int mes;
    int ano;
}Data;
```

Imagem 01 - Lista Data.

```
Data nasc;
Data Diag;
```

Imagem 02 - Utilização da lista Datas.

04.2 Lista Endereco

Aqui temos uma lista que foi desenvolvida para deixar o código mais claro.

```
typedef struct{
    char estado[50];
    char cidade[50];
    char bairro[50];
    char rua[50];
    int numero;
    int CEP;
}Endereco;
```

Imagem 03 - Lista Endereco.

```
Endereco loga;
```

Imagem 04 - Utilização da lista Endereco.

04.3 Lista Paciente

Esta é a lista que usamos para alocar os dados coletados na memória temporariamente. Foi escolhida a utilização de lista ao invés de diversas variáveis dentro da main, por conta da facilidade de passar a lista como parâmetro de entrada em função, bem como sua inserção interativa.

```
typedef struct paciente{
    char nome[50];
    int CPF;
    int telefone;
    char email[50];
    char Comorbidade[50];
    Endereco loga;
    Data nasc;
    Data Diag;
}Paciente;
```

Imagem 05 - Lista Paciente.

```
Paciente pac;
```

Imagem 06 - Utilização da lista Paciente.

04.4 Função Login

Esta função realiza a validação do Login do usuário do sistema, tendo como parâmetros de entrada duas strings, login e senha. Dentro desta, é inicializado duas strings, valideteLog e validatePass, que recebem seus respectivos valores por meio do comando strcpy.

Com as variáveis inicializadas e seus respectivos valores inseridos, seguimos com uma estrutura condicional, a qual compara os valores de login e validateLog, e senha com validatePass por meio do comando strcmp, que retorna um valor inteiro dependendo dos valores da variável.

Caso os valores de ambas as condições forem iguais a 0, retorna o valor 1 e fecha o programa, se o valor de alguma das duas condições for diferente a 0, retorna o valor 0 e sai da função.

```
int Login(char login[50], char senha[50]){
    char validateLog[50];
    char validatePass[50];
    strcpy(validateLog, "User Teste");
    strcpy(validatePass, "10pass20");
    if(strcmp(login, validateLog) == 0 && strcmp(senha, validatePass) == 0)
        return 1;
    return 0;
}
```

Imagem 07 - Função Login.

```
i = Login(login, password);
```

Imagem 08 - Chamada função Login.

04.5 Função AnexarArquivo

Esta função recebe como parâmetro de entrada a lista Paciente, e tem como retorno um inteiro. Ela inicia com a inicialização de uma string denominada arquivo, depois instância o ponteiro f como arquivo, após isso ela atribui o nome do paciente para dentro da variável arquivo, e depois concatenar o conteúdo com “.txt”.

Após trabalhar com a variável arquivo, cria um arquivo com o conteúdo da variável trabalhada acima,e imprime no mesmo todos os dados do paciente, depois confere se os dados foram incrementados por meio de uma estrutura condicional, que fecha arquivo independente da condicional, porém variando o valor de retorno entre 1 para condição verdadeira e 0 para falsa.

```
37  int anexarArquivo(Paciente iPac){
38      char arquivo[100];
39      FILE *f;
40      strcat(arquivo, iPac.nome);
41      strcat(arquivo, ".txt");
42      f = fopen(arquivo, "w");
43      fprintf(f, "*************************************************************");
44          fprintf(f, "\nNome: %s.", iPac.nome);
45          fprintf(f, "\nCPF: %d.", iPac.CPF);
46          fprintf(f, "\nTelefone: %d.", iPac.telefone);
47          fprintf(f, "\nData Nascimento: %d/%d/%d.", iPac.nasc.dia, iPac.nasc.mes, iPac.nasc.ano);
48          fprintf(f, "\nEndereco: %s' %d - %s' %s - %s ' %d.", iPac.loga.rua, iPac.loga.numero, iPac.loga.bairro, iPac.loga.cidade, iPac.loga.estado, iPac
    .loga.CEP);
49          fprintf(f, "\nE-mail: %s.", iPac.email);
50          fprintf(f,"\nComorbidade: %s.", iPac.Comorbidade);
51          fprintf(f, "\nData Diagnostico: %d/%d/%d.", iPac.Diag.dia, iPac.Diag.mes, iPac.Diag.ano);
52      fprintf(f, "\n*************************************************************");
53      if(f == NULL){
54          fclose(f);
55          return 0;
56      }
57      fclose(f);
58      return 1;
59  }
```

Imagem 09 - Função anexarArquivo.

05. MANUAL DE INSTRUÇÕES

Para se iniciar o programa, é necessário clicar no atalho presente dentro da pasta junto a este documento. Assim que iniciar o programa é necessário realizar o login, digitando “User Teste” como login, e “10pass20” para a senha.

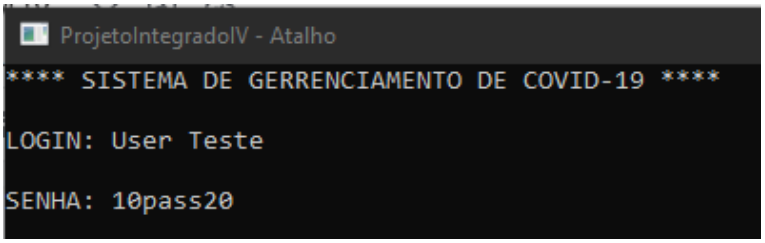

Imagem 10 - Login no sistema

Assim que efetuado o login no sistema, é iniciada a aplicação interativa do questionário de cadastro de paciente que foi diagnosticado com covid-19, digitando a informação solicitada, sem utilizar caracteres especiais e acentuação. Abaixo temos um exemplo de preenchimento da primeira etapa do questionário.

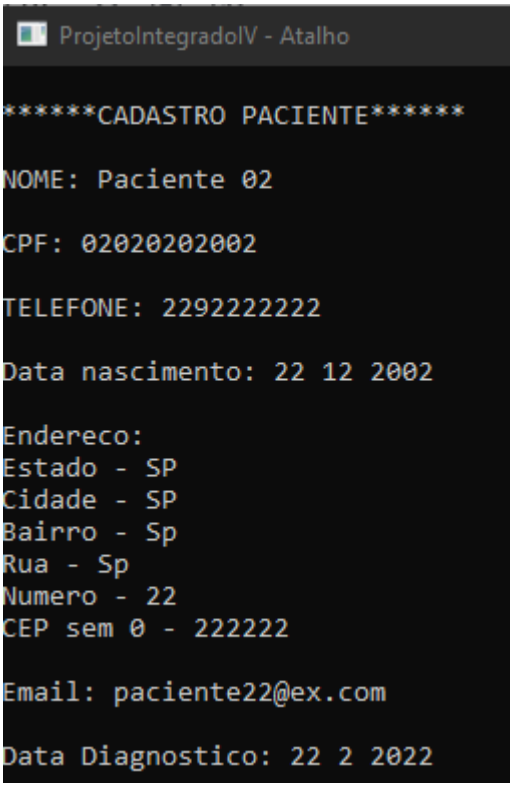

Imagem 11 - Exemplo da primeira etapa.

Em todos os campos se faz necessário após o preenchimento pressionar a tecla Enter no teclado, para inserir o dado na variável e dar continuidade ao programa. Nesta primeira etapa existem 3 casos especiais, sendo eles o preenchimento da data de nascimento, CEP, e data de diagnóstico. Para o preenchimento das datas, se adiciona os inteiros e separa o dia, mês e ano pelo espaço. Já no CEP, é solicitado apenas os números após o 0.

Já na segunda etapa do cadastro, temos uma questão que pergunta se o paciente possui alguma comorbidade, e é necessário entrar com S ou N, representando sim e não para o sistema. Caso a entrada seja S ou s, o mesmo irá perguntar qual a comorbidade do paciente, e assim que preenchido e pressionar Enter, irá para a tela confirmando que o paciente foi cadastrado. Caso a entrada seja N ou n, o programa irá exibir a mensagem confirmando o cadastro.

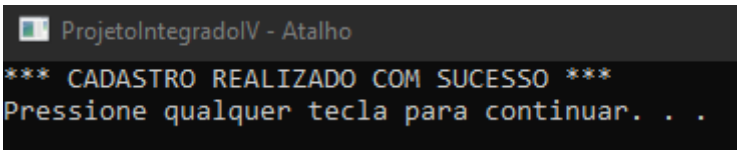

Imagem 12 - Confirmação de cadastro.