FIAP ON

LÓGICA DE PROGRAMAÇÃO

PROJETO: AGENDA DE CONTATOS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE CÓDIGOS-FONTE

SUMÁRIO

PROJETO: AGENDA DE CONTATOS

1 INTRODUÇÃO

Neste curso, aprendemos todos os comandos simples (primitivos), estruturados e avançados de lógica de programação nas linguagens Pseudocódigo, Python e Java.

PRIMITIVOS (NA RESPECTIVA ORDEM: PSEUDOCÓDIGO, PYTHON E JAVA):

- Saída de dados (Escreva / print() / System.out.print()).
- Entrada de dados (Leia / input() / next()).
- Processamento de dados.

ESTRUTURADOS:

- Decisão (Se / if / If()).
- Seleção (Escolha / if elif / switch()).

ESTRUTURAS DE REPETIÇÃO:

- Contator (Para / for / for()).
- Pré-condicional (Enquanto-faça / while / while()).
- Pós-condicional (Faça-Enquanto / while True / do while()).

Depois de explorarmos esses oito comandos de lógica de programação, aprendemos a parte avançada nessas mesmas três linguagens:

- Vetor;
- Matriz;
- Procedimento;
- Função.

Pronto! Agora chegou a hora de fazermos um projeto real, envolvendo todos esses conteúdos aprendidos. Neste capítulo, faremos o passo a passo do projeto agenda de contatos.

2 DEFINIÇÃO

Neste projeto, não abordaremos as melhores práticas em construção de tabelas/Banco de dados (como chave primária, normalização...) para armazenar os dados, pois daremos prioridade à manipulação dos comandos que aprendemos no curso para fazer as quatro operações fundamentais: cadastro, consulta, alteração e exclusão (CRU), além de listagem dos registros.

Vale salientar que essas operações serão feitas em estruturas de memória, ou seja, não serão armazenadas magneticamente no computador

Para este projeto, utilizaremos a estrutura apresentada na Tabela “Estrutura da tabela do projeto”:

Tabela 1 – Estrutura da tabela do projeto

CAMPO	TIPO
Nome	Texto
Celular	Texto
e-mail	Texto

Fonte: Elaborado pelo autor (2022)

3 FUNCIONAMENTO DO PROJETO

Antes de apresentarmos os projetos escritos em Pseudocódigo, Python e Java, vale ressaltar que a sistemática de utilização de matriz/listas entre essas linguagens são diferentes. Citaremos as diferenças mais importantes:

- Vetor e matriz:
  - No Pseudocódigo e em Java essas estruturas são estáticas, ou seja, deve-se definir antes a quantidade de linhas e colunas que elas armazenarão.
  - No Python essas estruturas são dinâmicas, ou seja, elas começam vazias e podem ser inseridos e removidos elementos no decorrer do programa.
- Tipo do vetor e matriz:

- No pseudocódigo e em Java, uma vez definido o tipo dos elementos, todas as células só podem ser daquele tipo.
- No Python, não há tipo definido, ou seja, cada célula pode conter qualquer tipo de dado.

- Codificação:
  - No pseudocódigo e em Java, toda manipulação é feita pelos indices e de forma manual, através dos comandos que aprendemos.
  - No Python, há comandos/métodos específicos para a manipulação das listas, por exemplo, o append() e del(), apesar de também poder tratar pelo índice os elementos já existentes.

Por esse motivo, não estranhem as construções diferentes dos subalgoritmos e do programa principal (do projeto) no Pseudocódigo e no Java (que são similares) com a construção do código Python.

Antes de apresentarmos os projetos nas linguagens que aprendemos, vamos ver o funcionamento de cada item do menu:

Vejamos a tela do menu:

```
********* M E N U ********
1 - Adicionar novo contato
2 - Editar contato
3 - Pesquisar contato
4 - Lista de contatos
5 - Apagar um contato
6 - Sair
Escolha uma opção: _
```

Quadro 1 – Tela do menu do projeto
Fonte: Elaborado pelo autor (2022)

A partir desta tela, temos a lista de opções que podemos efetuar em nossa tabela (tabela, lista e matriz são sinônimos neste projeto).

Escolhendo a opção 1, (Adicionar novo contato) aparecerá:

```
---------- PREENCHA O NOVO CONTATO:
Nome.....: Edson
```

```
Celular.....: 119543-5567
E-mail.....: eds@hotmail.com
```

Quadro 2 – Cadastro do primeiro contato
Fonte: Elaborado pelo autor (2022)

Os dados em azul claro são as informações digitadas pelo usuário para cadastrar o primeiro contato na tabela Agenda.

Para testarmos o funcionamento de todos os itens, vamos inserir mais dois contatos para uma melhor experiência e visualização de todas as funcionalidades:

```
---------- PREENCHA O NOVO CONTATO:
Nome.....: Maria
Celular.....: 1193322-3456
E-mail.....: maria@gmail.com
```

Quadro 3 – Cadastro do segundo contato
Fonte: Elaborado pelo autor (2022)

O contato 2 foi cadastrado no quadro acima; o contato 3, no quadro abaixo:

```
---------- PREENCHA O NOVO CONTATO:
Nome.....: Adriano
Celular.....: 1198786-8584
E-mail.....: adriano@ig.com.br
```

Quadro 4 – Cadastro do terceiro contato
Fonte: Elaborado pelo autor (2022)

Pronto! Os contatos: Edson, Maria e Adriano estão cadastrados.

Vamos ver se os contatos foram devidamente cadastrados? No menu, digite 4 (Lista de contatos) e veja a execução:

```
******** M E N U ********
1 - Adicionar novo contato
2 - Editar contato
3 - Pesquisar contato
4 - Lista de contatos
5 - Apagar um contato
6 - Sair
Escolha uma opção:4

---------- CONTATOS DA AGENDA:
Nome.....: Edson
```

```
Celular.....: 119543-5567
E-mail......: eds@hotmail.com
----------------------------------------
Nome........: Maria
Celular.....: 1193322-3456
E-mail......: maria@gmail.com
----------------------------------------
Nome........: Adriano
Celular.....: 1198786-8584
E-mail......: adriano@ig.com.br
----------------------------------------

---------- FIM DA AGENDA:
```

Quadro 5 – Listagem dos contatos
Fonte: Elaborado pelo autor (2022)

Todos os contatos foram listados.

Agora, nós pesquisaremos um contato (item 3).

Vamos primeiramente pesquisar o “Adriano”; como ele está cadastrado na tabela então aparecerão os dados dele.

Em seguida, pesquisaremos o “José”, que não está cadastrado na agenda, e será dada a mensagem “Contato não cadastrado!”, vejam esta sequência:

```
********* M E N U ********
1 - Adicionar novo contato
2 - Editar contato
3 - Pesquisar contato
4 - Lista de contatos
5 - Apagar um contato
6 - Sair
Escolha uma opção:3

---------- PESQUISE O CONTATO:
Digite o nome.........:Adriano
Nome........: Adriano
Celular.....: 1198786-8584
E-mail......: adriano@ig.com.br

********* M E N U ********
```

```
1 - Adicionar novo contato
2 - Editar contato
3 - Pesquisar contato
4 - Lista de contatos
5 - Apagar um contato
6 - Sair
Escolha uma opção:3

---------- PESQUISE O CONTATO:
Digite o nome.....:José
Contato não cadastrado!
```

Quadro 6 – Pesquisa dos contatos
Fonte: Elaborado pelo autor (2022)

Agora vejamos como funciona a edição de contatos.

Escolha a opção 2 (Editar contato) do menu: nessa opção, foi solicitada a digitação de um nome. “Edson” foi o escolhido por ser um nome existente.

Primeiramente foi exibido o registro para que o usuário visualize o(s) campo(s) que ele precisa alterar. Depois foram abertos os campos: celular e e-mail para digitação (Repare que não é permitido que seja alterada a chave da pesquisa. Nome, neste caso). Registro alterado!

Numa segunda edição, tentamos editar o contato “Marcelo”, mas, como ele não está cadastrado na tabela, apareceu a mensagem “Contato não cadastrado!”.

Depois escolhemos a opção 4 para listarmos os contatos e verificarmos se realmente o Edson foi editado. Veja este processo:

```
********* M E N U ********
1 - Adicionar novo contato
2 - Editar contato
3 - Pesquisar contato
4 - Lista de contatos
5 - Apagar um contato
6 - Sair
Escolha uma opção:2

---------- EDITANDO (PESQUISE O CONTATO):
Digite o nome.....:Edson
Nome.....: Edson
Celular.....: 119543-5567
```

```
E-mail......: eds@hotmail.com
---------- EDITE O CONTATO:
Nome.........: Edson
Celular......: 119392-9195
E-mail.......: eds.oliveira@gmail.com

********* M E N U ********
1 - Adicionar novo contato
2 - Editar contato
3 - Pesquisar contato
4 - Lista de contatos
5 - Apagar um contato
6 - Sair
Escolha uma opção:2

---------- EDITANDO (PESQUISE O CONTATO):
Digite o nome.........:Marcelo
Contato não cadastrado!

********* M E N U ********
1 - Adicionar novo contato
2 - Editar contato
3 - Pesquisar contato
4 - Lista de contatos
5 - Apagar um contato
6 - Sair
Escolha uma opção:4

---------- CONTATOS DA AGENDA:
Nome.........: Edson
Celular.....: 119392-9195
E-mail......: eds.oliveira@gmail.com
----------------------------------------
Nome.........: Maria
Celular.....: 1193322-3456
E-mail......: maria@gmail.com
----------------------------------------
Nome.........: Adriano
Celular.....: 1198786-8584
E-mail......: adriano@ig.com.br
----------------------------------------

---------- FIM DA AGENDA:
```

Quadro 7 – Edição dos contatos
Fonte: Elaborado pelo autor (2022)

Vamos, agora, para a operação mais delicada: a operação de exclusão. A sua mecânica é semelhante à mecânica de Edição: primeiro digitamos o nome, se for encontrado, lista-se o registro (senão, mostra que não existe) e pergunta-se se o usuário realmente quer excluí-lo, tendo que digitar S, para sim, ou N, para não.

Faremos três testes.

- No primeiro, digitaremos “Marilene” – não existe esse nome cadastrado na agenda – e será exibida a mensagem “Contato não cadastrado”.
- No segundo teste, digitaremos “Maria” – esse contato será encontrado na agenda – então será exibido o registro e questionado se realmente quer excluir; nesse caso, o usuário optou por excluir.
- No terceiro teste, o usuário digitou “Edson” e apareceu o registro. Ao ser indagado sobre excluir ou não, o usuário optou por não excluir.

Para verificar se as ações surtiram efeito, escolhemos listar os contatos. Veja como ficou todo esse processo:

```
********* M E N U ********
1 - Adicionar novo contato
2 - Editar contato
3 - Pesquisar contato
4 - Lista de contatos
5 - Apagar um contato
6 - Sair
Escolha uma opção:5

---------- EXCLUINDO (PESQUISE O CONTATO):
Digite o nome.....:Marilene
Contato não encontrado!

********* M E N U ********
1 - Adicionar novo contato
2 - Editar contato
3 - Pesquisar contato
4 - Lista de contatos
5 - Apagar um contato
```

```
6 - Sair
Escolha uma opção:5

---------- EXCLUINDO (PESQUISE O CONTATO):
Digite o nome.....:Maria
Nome.....: Maria
Celular.....: 1193322-3456
E-mail.....: maria@gmail.com
Confirma a exclusão do contato?
[S]im ou [N]ão?
s
Contato Excluído

******** M E N U ********
1 - Adicionar novo contato
2 - Editar contato
3 - Pesquisar contato
4 - Lista de contatos
5 - Apagar um contato
6 - Sair
Escolha uma opção:5

---------- EXCLUINDO (PESQUISE O CONTATO):
Digite o nome.....:Edson
Nome.....: Edson
Celular.....: 119392-9195
E-mail.....: eds.oliveira@gmail.com
Confirma a exclusão do contato?
[S]im ou [N]ão?
n
Exclusão cancelada!

******** M E N U ********
1 - Adicionar novo contato
2 - Editar contato
3 - Pesquisar contato
4 - Lista de contatos
5 - Apagar um contato
6 - Sair
Escolha uma opção:4

---------- CONTATOS DA AGENDA:
Nome.....: Edson
```

```
Celular.....: 119392-9195
E-mail......: eds.oliveira@gmail.com
----------------------------------------
Nome........: Adriano
Celular.....: 1198786-8584
E-mail......: adriano@ig.com.br
----------------------------------------

---------- FIM DA AGENDA:
```

Quadro 8 – Exclusão dos contatos
Fonte: Elaborado pelo autor (2022)

Por fim, o usuário escolheu o item 6 (Sair), e a aplicação se encerrou depois de exibir a mensagem “OBRIGADO POR UTILIZAR A NOSSA AGENDA :)”.

```
******** M E N U ********
1 - Adicionar novo contato
2 - Editar contato
3 - Pesquisar contato
4 - Lista de contatos
5 - Apagar um contato
6 - Sair
Escolha uma opção:6

OBRIGADO POR UTILIZAR A NOSSA AGENDA :)
```

Quadro 9 – Finalizando a aplicação
Fonte: Elaborado pelo autor (2022)

Vejamos agora os códigos-fonte deste projeto, nas linguagens de programação que vimos.

O código-fonte está comentado, documentando a assinatura de cada Subalgoritmo e os passos mais importantes.

Confiram!

4 PROJETO NO PSEUDOCÓDIGO

```
// ######## S U B A L G O R I T M O S ############################

/*
 * Descrição....: Este procedimento limpa a matriz
 * Nome.........: limpar_matriz(matriz[][]: Texto)
 * Tipo.........: void
*/
        Procedimento limparMatriz(mm[][]: Texto)
        Var
                l, c: inteiro
        início
                // Insere vazio em todas as células da matriz
                para l de 0 até 10 inc 1 faça
                        para c de 0 até 10 inc 1 faça
                                mm[l][c] = ""
                        fim_para
                fim_para
        Fim

/*
 * Descrição....: Este procedimento cadastra um novo contato
 * Nome.........: novo(matriz[][]: Texto, l: inteiro)
 * Tipo.........: void
*/

        Procedimento novo(mm[][]: Texto, l: inteiro)
        Início
                // Pede ao usuário a edição dos campos da linha por parâmetro
                Escreva "---------- PREENCHA O NOVO CONTATO: "
                Escreva "Nome.........: "
                Leia mm[l][0]
                Escreva "Celular......: "
                Leia mm[l][1]
                Escreva "E-mail.......: "
                Leia mm[l][2]
        Fim

/*
 * Descrição....: Este procedimento edita um contato
 * Nome.........: editarContato(matriz: Texto, l: inteiro)
```

```
 * Tipo.........: void
*/
        Procedimento editarContato(mm[][]: Texto, l: inteiro)
        Início
                // Permite ao usuário editar o contato encontrado
                // Não permite que o nome seja editado por ser a chave
                Escreva "---------- EDITE O CONTATO: "
                Escreva "Nome.........: ", mm[l][0]
                Escreva "Celular......: "
                Leia mm[l][1]
                Escreva "E-mail.......: "
                Leia mm[l][2]
        Fim

/*
 * Descrição....: Esta função retorna a próxima linha em branco da matriz
 * Nome.........: linhaProximoContato(matriz[][]: Texto)
 * Tipo.........: void
*/
        Função linhaProximoContato(mm[][]: Texto): Inteiro
        Var
                l: inteiro
        Início
                para l de 0 até 10 inc 1 faça
                        Se mm[l][0] == "" então
                                // caso tenha encontrado, retorne o número da linha
                                // próxima linha vazia
                                retorne l
                        Fim_se
                Fim_para
                // -1 representa a matriz estar cheia
                Retorne -1
        Fim

/*
 * Descrição....: Este procedimento exibe um contato
 * Nome.........: exibirContato(matriz[][]: Texto, l: inteiro)
 * Tipo.........: void
*/
        Procedimento exibirContato(mm[][]: Texto, l: inteiro)
        Início
                // Exibe o registro da linha passada por parâmetro
                Escreva "Nome........: ", mm[l][0]
```

```
            Escreva "Celular.....: ", mm[l][1]
            Escreva "E-mail......: ", mm[l][2]
        Fim

/*
 * Descrição....: Este procedimento lista todos os contatos
 * Nome.........: listarAgenda(matriz[][]: Texto)
 * Tipo.........: void
*/
        Procedimento listarAgenda(mm[][]: Texto)
        Var
            l: inteiro
        Início
            Escreva "------------ CONTATOS DA AGENDA: "
            para l de 0 até 10 inc 1 faça
                // Enquanto tiver registro, exibe-o
                Se mm[l][0] != "" Então
                    exibirContato(mm, l)
                    Escreva "----------------------------------------"
                Fim_se
            Fim_para
            Escreva "\n------------ FIM DA AGENDA: "
        Fim

/*
 * Descrição....: Esta função pesquisa o contato e retorna a linha
 * Nome.........: pesquisarContato(matriz[][]: Texto, nome: Texto)
 * Tipo.........: inteiro (retorna -1 se não encontrou)
*/
        Função pesquisarContato(mm[][]: Texto, n: Texto): Inteiro
        Var
            l: inteiro
        Início
            // caso encontre o contato, retorna a linha onde ele está
            para l de 0 até 10 inc 1 faça
                Se mm[l][0] == n Então
                    retorne l
                Fim_se
            // Se não encontrou o contato, retorna -1
            retorne -1
        Fim

/*
```

```
 * Descrição....: Este procedimento exclui a linha passada por parâmetro
 * Nome.....: ExcluiLinha(String matriz, l: inteiro)
 * Tipo.....: void
*/
        Procedimento excluiLinha(mm[][]: Texto, l: inteiro)
        Início
                // exclui o contato a partir da linha passada por parâmetro
                mm[l][0] = ""
                mm[l][1] = ""
                mm[l][2] = ""
                Escreva "Contato Excluído"
        Fim

/*
 * Descrição....: Este procedimento pesquisa e exclui um contato
 * Nome.....: apagarContato(String matriz, String nome)
 * Tipo.....: void
*/
    Procedimento apagarContato(mm[][]: Texto, n: Texto)
    Var
        Achou: Lógica
        opcao: Texto
        linha: Inteiro
    Início
        // inicia a variável, achou como false
        achou = Falso
        // alimenta a linha com o valor da pesquisa (-1 não encontrou)
        linha = pesquisarContato(mm, n)
        Se linha != -1 então
            // Exibe o contato a partir da linha
            exibirContato(mm,linha)
            // Confirma com o usuário se ele quer excluir ou nao o contato
            Escreva "Confirma a exclusão do contato?\n[S]im ou [N]ão?"
            Leia opcao
            Se opcao == "s" ou opcao == "S" Então
                // se a escolha for Sim, exclui o contato
                excluiLinha(mm, linha)
            else
                // Cancela a exclusão
                Escreva "Exclusão cancelada!"
            Fim_se
        senão
            // contato não encontrado
```

```
            Escreva "Contato não encontrado!"
        Fim_se

    Fim

/*
 * Descrição....: Este procedimento lista as opções do Menu
 * Nome.....: apagarContato(String matriz, String nome)
 * Tipo.....: void
*/

        Procedimento exibeMenu()
        Início
            Escreva "********* M E N U ********"
            Escreva "1 - Adicionar novo contato"
            Escreva "2 - Editar contato"
            Escreva "3 - Pesquisar contato"
            Escreva "4 - Lista de contatos"
            Escreva "5 - Apagar um contato"
            Escreva "6 - Sair"
        Fim

// ######## P R O G R A M A   P R I N C I P A L ######################
        Programa Agenda_de_contato
        Var
            // variáveis e objetos públicos
            agenda[10][3]: Texto
            opcao, linha: inteiro
            nome: Texto
        Início
            // ao iniciar a aplicação, sempre é bom limparmos a matriz para
            // que a "sujeira" do Buffer não influencie nos resultados
            limparMatriz(agenda)

            faça
                // Exibição do menu
                exibeMenu()

                // Colhe a opção escolhida pelo usuário
                Escreva "Escolha uma opção:"
                Leia opcao

                // Seleciona a opção escolhida
```

```
Escolha(opcao)

    // Caso a escolha seja "adicionar novo contato"
    caso 1:
        // Retorna a próxima linha disponível antes de
        // incluir o novo contato
        novo(agenda, linhaProximoContato(agenda))


    // Caso a escolha seja "Editar contato"
    caso 2:
        Escreva "--------- EDITANDO (PESQUISE O CONTATO): "
        Escreva "Digite o nome.....:"
        nome = entrada.next(
        linha = pesquisarContato(agenda, nome
        Se (linha == -1) então
            // informa se não encontrou o contato
            Escreva "Contato não cadastrado!\n"
        Senão
            // se encontrou, exibe o contato e o edita
            exibirContato(agenda, linha
            editarContato(agenda, linha
        Fim_se


    // Efetua a pesquisa do contato
    caso 3:
        // pede o nome
        Escreva "---------- PESQUISE O CONTATO: "
        Escreva "Digite o nome.....:"
        nome = entrada.next(
        // Retorna a linha da pesquisa
      linha = pesquisarContato(agenda, nome
        Se (linha == -1)
            // Se o contato não existe
            Escreva "Contato não cadastrado!\n"
        Senão
            // Se o contato existe, exibí-lo
            exibirContato(agenda, linha
        Fim_se


    // Lista todos os Contatos da agenda
```

```
                caso 4:
                    listarAgenda(agenda

                // Exclui um registro digitado pelo usuário
                caso 5:

                    Escreva "------- EXCLUINDO (PESQUISE O CONTATO): "
                    Escreva "Digite o nome.....:"
                    nome = entrada.next(
                    apagarContato(agenda, nome

                caso 6:
                    Escreva "OBRIGADO POR UTILIZAR A NOSSA
AGENDA :)"

            Fim_Escolha
        Enquanto opcao != 6
    Fim
```

Código-fonte 1 – Projeto escrito no pseudocódigo
Fonte: Elaborado pelo autor (2022)

5 PROJETO EM PYTHON

```python
agenda = []  # Criando uma lista vazia.


# DEFINIÇÃO DAS FUNÇÕES

# Descrição: Este procedimento cria um novo contato na agenda.
# Nome: novo()
# Tipo: procedimento
def novo():
    global agenda  # Definindo variável como Global.
    nome = p_nome()
    celular = input("Celular....: ")
    email   = input("E-mail.....: ")
    agenda.append([nome, celular, email])  # Adicionando os dados na agenda
    print("\n-------------------------"
          "\nRegistro gravado com Sucesso!!!\n"
          "-------------------------")
```

```
# Descrição: Este procedimento lê um nome.
# Nome: p_nome()
# Tipo: procedimento
def p_nome():
    return(input("Nome.....: "))


# Descrição: Este procedimento lista um registro
# Nome: listar_dados(nome, celular, email)
# Tipo: procedimento
def listar_dados(nome, celular, email):
    print("Nome: %s\nCelular: %s\nEmail: %s" % (nome, celular, email))
    print("--------------------------------")

# Descrição: Este procedimento lista todos os registros da matriz.
# Nome: listar()
# Tipo: procedimento
def listar():  # Função para mostrar lista de contatos.
    print("\nCONTATOS DA AGENDA ################\n")
    for e in agenda:
        listar_dados(e[0], e[1], e[2])
    print("\nFIM DA AGENDA ####################\n")

# Descrição: Esta função pesquisa um contato pelo nome.
# Nome: pesquisa(nome): int
# Tipo: função
def pesquisa(nome):  # Função para pesquisar contatos.
    name = nome.lower()
    for d, e in enumerate(agenda):  # percorre toda a matriz.
        if e[0].lower() == name:  # procura o nome desejado
            return d  # retorna o índice do nome encontrado
    return None  # retorna vazio se não encontrar

# Descrição: Este procedimento exibe o registro ou mensagem de insucesso
# Nome: pesquisar():
# Tipo: procedimento
def pesquisar():
    # pesquisa o nome
    p = pesquisa(p_nome())  # Entrada de dados.
    if p != None:
        print("Registro encontrado!")
        # atualiza as variáveis se encontrou
```

```
        nome = agenda[p][0]
        celular = agenda[p][1]
        email = agenda[p][2]
        # mostra o registro
        listar_dados(nome, celular, email)
    else:
        # exibe a mensagem de insucesso na procura do registro
        print("\nNome não encontrado!!!")

# Descrição: Este procedimento apaga um contato
# Nome: apagar():
# Tipo: procedimento
def apagar():
    global agenda
    nome = p_nome()
    # retorna o índice do nome ou vazio
    p = pesquisa(nome)
    if p != None:  # Se encontrou o contato
        del agenda[p]  # exclui o contato
        print("\n----------------------------"
              "\nRegistro APAGADO com Sucesso!!!\n"
              "----------------------------")
    else:
        # não encontrou o registro para excluir
        print("Nome não encontrado.")

# Descrição: Este procedimento edita um contato
# Nome: editar():
# Tipo: procedimento
def editar():
    p = pesquisa(p_nome())  # Entrada de dados.
    # Se encontrou o registro
    if p != None:
        # mostra o nome e pede a edição dos demais
        nome = agenda[p][0]
        print("Nome.....: ", nome)
        celular = input("Celular....:")
        email = input("E-mail.....:")
        agenda[p] = [nome, celular, email]  # Armazenando os novos dados.
        print("\n----------------------------"
              "\nRegistro EDITADO com Sucesso!!!\n"
              "----------------------------")
    else:
```

```
        print("Nome não encontrado.")  # Executa caso a condição seja falsa.


# Descrição: Esta função valida SE O ITEM DIGITADO FOI VALIDO
# Nome: validar(pergunta, inicia, fim): int
# Tipo: função
def validar(pergunta, inicio, fim):  # Função para validar números inteiros.
    while True:  # Criando um loop infinito.
        try:  # Criando um acordo/condição.
            valor = int(input(pergunta))  # Entrada de dados.
            if inicio <= valor <= fim:  # Determinando uma condição.
                return (valor)  # Executa caso for verdadeira.
            else:
                return (0)
        except ValueError:  # Executa caso for falsa.
            print("Valor inválido, favor digitar entre %d e %d" % (inicio, fim))


# Descrição: Esta função retorna o item do menu ou 0 para invalido
# Nome: menu(pergunta, inicia, fim): int
# Tipo: função
def menu():  # Função que exibe o menu de opções.
    print("""
    1 - Adicionar novo contato
    2 - Editar um contato
    3 - Pesquisar contato
    4 - Lista de contatos
    5 - Apagar um contato
    6 - Sair
""")

    return validar("Escolha uma opção: ", 1, 6)

# PROGRAMA PRINCIPAL
while True:  # Criando um loop infinito.
    opcao = menu()
    if opcao == 0:
        print("Opcao Inválida!")
    elif opcao == 6:
        break
    elif opcao == 1:
        novo()
    elif opcao == 2:
```

```
        editar()
    elif opcao == 3:
        pesquisar()
    elif opcao == 4:
        listar()
    elif opcao == 5:
        apagar()
```

Código-fonte 2 – Projeto escrito no Python
Fonte: Elaborado pelo autor (2022)

6 PROJETO EM JAVA

```
import java.util.Scanner;
public class First
{
    // ######## S U B A L G O R I T M O S ##########################

    // variáveis e objetos públicos
    public static Scanner entrada = new Scanner(System.in);
    public static String agenda[][] = new String[10][3];

    /*
     * Descrição....: Este procedimento limpa a matriz
     * Nome.....: limpar_matriz(String matriz)
     * Tipo.....: void
    */
    public static void limparMatriz(String mm[][])
    {
        // Insere vazio em todas as células da matriz
        for(int l = 0; l < 10; l++)
        {
            for(int c = 0; c < 3; c++)
            {
                mm[l][c] = "";
            }
        }
    }

    /*
     * Descrição....: Este procedimento cadastra um novo contato
     * Nome.....: novo(String matriz, int linha)
     * Tipo.....: void
```

```
*/

public static void novo(String mm[][], int l)
{
      // Pede ao usuário a edição dos campos da linha passada por parâmetro
      System.out.println("------------ PREENCHA O NOVO CONTATO: ");
      System.out.print("Nome.....: ");
      mm[l][0] = entrada.next();
      System.out.print("Celular.....: ");
      mm[l][1] = entrada.next();
      System.out.print("E-mail.....: ");
      mm[l][2] = entrada.next();
}

/*
 * Descrição....: Este procedimento edita um contato
 * Nome.....: editarContato(String matriz, int linha)
 * Tipo.....: void
*/
public static void editarContato(String mm[][], int l)
{
      // Permite ao usuário editar o contato encontrado
      // Não permite que o nome seja editado por ser a chave de pesquisa
      System.out.println("------------ EDITE O CONTATO: ");
      System.out.print("Nome.....: " + mm[l][0] +"\n");
      System.out.print("Celular.....: ");
      mm[l][1] = entrada.next();
      System.out.print("E-mail.....: ");
      mm[l][2] = entrada.next();
}

/*
 * Descrição....: Esta função retorna a próxima linha em branco da matriz
 * Nome.....: linhaProximoContato(String matriz, int linha)
 * Tipo.....: void
*/
public static int linhaProximoContato(String mm[][])
{
      for(int l = 0; l < 10; l++)
            if (mm[l][0].equals(""))
            {
                  // caso tenha encontrado, retorne o número da linha
                  // próxima linha vazia
```

```
                return l;
            }
        // -1 representa a matriz estar cheia
        return -1;
    }

    /*
     * Descrição....: Este procedimento exibe um contato
     * Nome.....: exibirContato(String matriz, int linha)
     * Tipo.....: void
    */
    public static void exibirContato(String mm[][], int linha)
    {
        // Exibe o registro da linha passada por parâmetro
        System.out.println("Nome.....: " + mm[linha][0]);
        System.out.println("Celular.....: " + mm[linha][1]);
        System.out.println("E-mail.....: " + mm[linha][2]);
    }

    /*
     * Descrição....: Este procedimento lista todos os contatos cadastrados
     * Nome.....: listarAgenda(String matriz)
     * Tipo.....: void
    */
    public static void listarAgenda(String mm[][])
    {
        System.out.println("------------ CONTATOS DA AGENDA: ");
        for(int l = 0; l < 10; l++)
        {
            // Enquanto tiver registro, exibe-o
            if (mm[l][0] != "") {
                exibirContato(mm, l);
                System.out.println("----------------------------------------");
            }
        }
        System.out.println("\n------------ FIM DA AGENDA: ");
    }

    /*
     * Descrição....: Esta função pesquisa o contato e retorna a linha
     * Nome.....: pesquisarContato(String matriz, string nome)
     * Tipo.....: int (retorna -1 se não encontrou)
    */
```

```
public static int pesquisarContato(String mm[][], String n)
{
    // caso encontre o contato, retorna a linha onde ele está
    for(int l = 0; l < 10; l++)
        if (mm[l][0].equals(n)) {
            return l;
        }
    // Se não encontrou o contato retorna -1
    return -1;
}

/*
 * Descrição....: Este procedimento exclui a linha passada por parâmetro
 * Nome.....: ExcluiLinha(String matriz, int linha)
 * Tipo.....: void
*/
public static void excluiLinha(String mm[][], int l)
{
    // exclui o contato a partir da linha passada por parâmetro
    mm[l][0] = "";
    mm[l][1] = "";
    mm[l][2] = "";
    System.out.println("Contato Excluído");
}

/*
 * Descrição....: Este procedimento pesquisa e exclui um contato
 * Nome.....: apagarContato(String matriz, String nome)
 * Tipo.....: void
*/
public static void apagarContato(String mm[][], String n)
{
    // inicia a variável que achou como false
    boolean achou = false;
    // alimenta a linha com o valor da pesquisa (-1 não encontrou)
    int linha = pesquisarContato(mm, n);
    String opcao;
    if(linha != -1)
    {
        // Exibe o contato a partir da linha
        exibirContato(mm,linha);
        // Confirma com o usuário se ele quer excluir ou nao o contato
        System.out.println("Confirma a exclusão do contato?\n[S]im ou [N]ão?");
```

```
            opcao = entrada.next();
            if(opcao.equals("s") || opcao.equals("S"))
            {
                // se a escolha for Sim, exclui o contato
                excluiLinha(mm, linha);
            }
            else
            {
                // Cancela a exclusão
                System.out.println("Exclusão cancelada!");
            }
        }
        else
        {
            // Contato não encontrado
            System.out.println("Contato não encontrado!");
        }

    }

    public static void exibeMenu()
    {
        System.out.println("********* M E N U ********");
        System.out.println("1 - Adicionar novo contato");
        System.out.println("2 - Editar contato");
        System.out.println("3 - Pesquisar contato");
        System.out.println("4 - Lista de contatos");
        System.out.println("5 - Apagar um contato");
        System.out.println("6 - Sair");
    }

    // ######## P R O G R A M A   P R I N C I P A L ##############################
    public static void main(String[] args)
    {
        int opcao, linha;
        String nome;

        // ao iniciar a aplicação, sempre é bom limparmos a matriz para que a
"sujeira"
        // do Buffer não influencie nos resultados
        limparMatriz(agenda);

        do
```

```
        {
            // Exibição do menu
            exibeMenu();

            // Colhe a opção escolhida pelo usuário
            System.out.print("Escolha uma opção:");
            opcao = entrada.nextInt();
            System.out.println();

            // Seleciona a opção escolhida
            switch(opcao)
            {
                // Caso a escolha seja "adicionar novo contato"
                case 1:
                    novo(agenda, linhaProximoContato(agenda));
                    break;

                // Caso a escolha seja "Editar contato"
                case 2:
                    System.out.println("------------ EDITANDO (PESQUISE O
CONTATO): ");
                    System.out.print("Digite o nome.....:");
                    nome = entrada.next();
                    linha = pesquisarContato(agenda, nome);
                    if (linha == -1)
                    {
                        // informa se não encontrou o contato
                        System.out.print("Contato não Cadastrado!\n");
                    }
                    else
                    {
                        // se encontrou, exibe o contato e o edite
                        exibirContato(agenda, linha);
                        editarContato(agenda, linha);
                    }
                    break;

                // Efetua a pesquisa do contato
                case 3:
                    // pede o nome
                    System.out.println("------------ PESQUISE O CONTATO: ");
                    System.out.print("Digite o nome.....:");
                    nome = entrada.next();
```

```
                        // Retorna a linha da pesquisa
                        linha = pesquisarContato(agenda, nome);
                        if (linha == -1)
                        {
                            // Se o contato não existe
                            System.out.print("Contato não Cadastrado!\n");
                        }
                        else
                        {
                            // Se o contato existe, exibí-lo
                            exibirContato(agenda, linha);
                        }
                        break;

                    // Lista todos os Contatos da agenda
                    case 4:
                        listarAgenda(agenda);
                       break;

                    // Exclui um registro digitado pelo usuário
                    case 5:

                        System.out.println("------------ EXCLUINDO (PESQUISE O
CONTATO): ");
                        System.out.print("Digite o nome.....:");
                        nome = entrada.next();
                        apagarContato(agenda, nome);
                        break;

                    case 6:
                        System.out.print("OBRIGADO POR UTILIZAR A NOSSA
AGENDA :)");

                }
                System.out.println();
            }while(opcao != 6);
        }
}
```

Código-fonte 3 – Projeto escrito em Java
Fonte: Elaborado pelo autor (2022)

Com este projeto, finalizamos o curso de Lógica de Programação. Vimos as instruções fundamentais de programação, os conceitos avançados de programação e a aplicação da Lógica de Programação em um projeto real.

Lembrem-se, além de estudar tem que praticar para atingir o sucesso! Estudem e pratiquem.