



Estrutura de Dados

Agenda

```
1- #include <stdio.h>
2- #include <stdlib.h>
3- #include <string.h>
4- #include <stdbool.h>
5-
6- #define MAX_TAREFAS 50
7-
8- typedef int TIPOCHAVE;
9- typedef Char dataRef[11];
10-
11- typedef struct {
12-     TIPOCHAVE chave;
13-     dataRef dataCompromisso[20];
14-     char descricaoCompromisso[300];
15- } Agenda;
16-
17- bool realidasData(const char *data) {
18-     int anos, meses, dias;
19-     if (sscanf(data, "%d-%d-%d", &anos, &meses, &dias) != 3) {
20-         return false;
21-     }
22-
23-     if (meses <= 0 || meses > 12 || dias <= 0 || dias > 31) {
24-         return false;
25-     }
26-
27-     return true;
```

```

28- 3
29-
30- bool validarHora (const char* hora) {
31-     int horas, minutos, segundos;
32-     if (sscanf (hora, "%d:%d:%d", &horas, &minutos, &segundos) != 3) {
33-         return false;
34-     }
35-
36-     if (horas < 0 || horas > 23 || minutos < 0 || minutos > 59 || segundos < 0 || segundos > 59) {
37-         return false;
38-     }
39-
40-     return true;
41- }
42-
43- void Criartimestamp (char* timestamp) {
44-     char data[11], hora[9];
45-
46-     do {
47-         printf ("Digite a data do seu novo compromisso (aaaa-mm-dd): ");
48-         scanf ("%s", data);
49-     } while (!validaData (data));
50-
51-     do {
52-         printf ("Digite o horario do seu novo compromisso (hh:mm:ss): ");
53-         scanf ("%s", hora);
54-     } while (!validaHora (hora));
55-
56-     sprintf (timestamp, "%s %s", data, hora);
57- }

```

```

58-
59- void alterarCompromisso(agenda * lista, char * compromisso) {
60-     printf("Digite a descrição do seu novo compromisso:");
61-     fflush(stdin);
62-     fgets(compromisso, 300, stdin);
63-     compromisso[strcspn(compromisso, "\n")] = '\0';
64-     if (strlen(compromisso) == 0) {
65-         printf("Descrição vazia. Digite novamente.\n");
66-     }
67- }
68-
69- void exibirElementos(Agenda * lista, int tamanho) {
70-     for (int i = 0; i < tamanho; i++) {
71-         printf("Index: %d, Data: %s, Descrição: %s\n", lista[i].chave, lista[i].data, lista[i].compromisso,
72-               lista[i].descricaoCompromisso);
73-     }
74-
75- void adicionarTarefa(Agenda * lista, int * tamanho, int * chave, char * novaData,
76-                       char * novaDescricao) {
77-     if (*tamanho >= MAX_TAREFAS) {
78-         printf("A lista de tarefas está cheia. Não é possível adicionar mais tarefas.\n");
79-         return;
80-     }
81-     lista[*tamanho].chave = *chave;
82-     (*chave)++;
83-     strcpy(lista[*tamanho].dataCompromisso, novaData);
84-     strcpy(lista[*tamanho].descricaoCompromisso, novaDescricao);
85-     (*tamanho)++;
86-
87- }

```

88-

```
89- void editarCompromisso(Agenda lista[], int chance, const char *descricao, const char *data){  
90-     for (int i = 0; i < MAX_TAREFAS; i++) {  
91-         if (lista[i].chance == chance) {  
92-             strcpy(lista[i].dataCompromisso, data, sizeof(lista[i].dataCompromisso));  
93-             strcpy(lista[i].descricaoCompromisso, descricao, sizeof(lista[i].descricaoCompromisso));  
94-             break; // Se encontrou a chance, pode sair do loop  
95-         }  
96-     }  
97- }  
98-  
99-  
100- void removerCompromisso(Agenda lista[], int *tamanho, int chance){  
101-     int encontrado = 0;  
102-     for (int i = 0; i < *tamanho; i++) {  
103-         if (lista[i].chance == chance) {  
104-             encontrado = 1;  
105-             for (int j = i; j < *tamanho - 1; j++) {  
106-                 lista[j] = lista[j + 1];  
107-             }  
108-             lista[*tamanho - 1].chance = -1;  
109-             (*tamanho) --;  
110-             break;  
111-         }  
112-     }  
113-     if (!encontrado) {  
114-         printf("Chance não encontrada. Nenhum compromisso foi removido.\n");  
115-     }  
116- }  
117-  
118- }
```



```

119- Void consultarElemento (Agenda lista[], int chance) {
120-     for (int i=0; i < MAX_TAREFAS; i++) {
121-         if (lista[i].Chance == chance) {
122-             printf("Chave: %d, Data: %s, Descrição: %s\n", lista[i].Chave,
123-                   lista[i].dataCompromisso, lista[i].descricaoCompromisso);
124-             return;
125-         }
126-     }
127-     printf("Chave não encontrada. Nenhum compromisso foi encontrado.\n");
128-
129-
130-     Int comparar_agenda (const social *a, const social *b) {
131-         Agenda *compromisso_a = (Agenda *)a;
132-         Agenda *compromisso_b = (Agenda *)b;
133-
134-         return strcmp(compromisso_a->dataCompromisso, compromisso_b->
135-                         dataCompromisso);
136-
137-     Int comprimentoAgenda (Agenda lista[], int tamanho) {
138-         int contador = 0;
139-         for (int i=0; i < tamanho; i++) {
140-             if (lista[i].Chance != -1) {
141-                 contador++;
142-             }
143-         }
144-         return contador;
145-     }
146-

```

```

147- void limpargenda (Agenda * lista, int * tamanho) {
148-     for (int i = 0; i < *tamanho; i++) {
149-         lista[i].chave = -1;
150-     }
151-     *tamanho = 0;
152-     printf ("pressione qualquer tecla para continuar.");
153- }
154-
155- void limpastela () {
156-     printf ("pressione qualquer tecla para continuar.");
157-     flush (stdin);
158-     getch();
159-     system ("cls");
160- }
161-
162- int main () {
163-     Agenda lista [MAX_TAREFAS];
164-     int chave = 1; // inicializa a chave sequencial com 1
165-     int tamanho = 0;
166-     int escolha;
167-     int indice;
168-     int acao;
169-     int app = 1;
170-     int comprimento_agenda;
171-     char data [11];
172-     char novaData [11];
173-     char descricao [50];
174-     char novaDescricao [50];
175-

```



176- **While** ($APP == 1$) {

177- printf ("\nMenu:\n");
178- printf ("1. Verificar Compromissos\n");
179- printf ("2. Buscar Compromissos\n");
180- printf ("3. Inserir Novo Compromisso\n");
181- printf ("4. Mostrar Número de compromissos\n");
182- printf ("5. Editar/Remover Compromissos\n");
183- printf ("6. Reiniciar Agenda\n");
184- printf ("0. Sair\n");
185- printf ("Escolha uma opção:");
186- scanf ("%d", & escolha);
187-

188- **Switch** (escolha) {

189- **Case 1:**

190- **if** (tamanho == 0) {

191- printf ("Você não possui compromissos na sua agenda.\n");

192- **else** {

193- exibirElementos (listas, tamanho);

194- }

195- limparTela ();

196- break;

197- **Case 2:**

198- **if** (tamanho == 0) {

199- printf ("Você não possui compromissos na sua agenda.\n");

200- **else** {

```

201-     printf ("Digite a chance a ser consultado: ");
202-     scanf ("%d", & indice);
203-     printf ("Elemento %d: \n", indice);
204-     consultarElemento (lista, indice);
205- }
206-     limpazetela ();
207-     break;
208- Case 3:
209-     alterarDescricaoCompromisso (novaDescricao);
210-     CriarTimestamp (novaData);
211-     adicionarTarefa (lista, & tamanho, & chance, novaData, novaDescricao);
212-     qsort (lista, tamanho, sizeof (Agenda), Comparar_agenda);
213-     limpazetela ();
214-     break;
215- Case 4:
216-     Comprimento_agenda = comprimentoAgenda (data, tamanho);
217-     printf ("Voce tem %d compromissos em sua agenda.\n", Comprimento_agenda);
218-     limpazetela ();
219-     break;
220- Case 5:
221-     exibirElementos (listas, tamanho);
222-     printf ("Digite o chance do compromisso: ");
223-     scanf ("%d", & chance);
224-     printf ("\n1. Editar\n");
225-     printf ("2. Remover\n");
226-     scanf ("%d", & acao);
227-     Switch (acao
228-

```

```

229-
230-           alterarDescricaoComPromissso (novaDescricao);
231-           CriarTimestamp (novaData);
232-           editarComPromissso (lista, chave, novaDescricao, novaData);
233-           qSort (lista, tamAnho, sizeOf (Agenda), Comparar_Agenda);
234-           printf ("ComPromissso %s Editado.\n");
235-           break;
236-       Case 2:
237-           removerComPromissso (lista, & tamAnho, chave);
238-           qSort (lista, TamAnho, SizeOf (Agenda), Comparar_Agenda);
239-           printf ("ComPromissso Removido.\n");
240-           break;
241-       }
242-       LimparTela ();
243-       break;
244-   Case 6:
245-   if (tamAnho == 0) {
246-       printf ("A lista está vazia.\n");
247-   } else {
248-       LimparAgenda (lista, & tamAnho);
249-   }
250-   LimparTela ();
251-   break;
252- Case 0:
253-     printf ("Encerrando o programa.\n");
254-     app = 0;
255-     break;
256- default:
257-     printf ("opção inválida. Tente novamente.\n");

```



258- limportela();
259- }
260- }
261-
262- return 0;
263- }