data • •

S T Q Q S S D

Sistema Acadêmico em C

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <unistd.h>

struct Cadastro {
    char Aluno [50];
    char Curso [20];
    char CPF [19];
    char Matricula [20];
    char Turno [11];
};

struct Cadastro* CreateCadastro(const char* Aluno, const
char* Curso, const char* CPF, const char* Matricula,
const char* Turno) {
    if(NewCadastro) {
        strcpy(NewCadastro->Aluno, Aluno);
        strcpy(NewCadastro->Curso, Curso);
        strcpy(NewCadastro -> CPF, CPF);
        strcpy(NewCadastro -> Matricula, Matricula);
        strcpy(NewCadastro -> Turno, Turno);
        NewCadastro -> Next = NULL;
    }
    Return NewCadastro;

Void AddCadastro(struct Cadastro** List, const char*
Aluno, const char* Curso, const char* CPF, const
char * Matricula, const char* Turno) {
    struct Cadastro* NewCadastro = CreateCadastro(Aluno, Curso
```

data • •

S T Q Q S S D

```
CPF, MATRICULA, TURNO);
  IF (NewCadastro) {
     NewCadastro -> Next = *List;
     *List = NewCadastro;
  }
}

Void ExibirCadastro(const struct Cadastro* List) {
   const struct Cadastro* Current = List;
   int Count = 0; // Variável para contar cadastros

   while (Current != NULL) {
   printf("\n\nNome do Aluno: %s\n", Current -> Aluno);
   printf("Curso: %s\n", Current -> Curso);
   printf("CPF do Aluno: %s\n", Current -> CPF);
   printf("Matrícula: %s\n", Current -> MATRICULA);
   printf("Turno: %s\n\n", Current -> Turno);
   Current = Current -> Next;
   Count++;
   }
    printf("Total de Alunos Matriculados: %d\n\n\n", Count); // Exibe a Quantidade de Alunos }

   Void DeleteCadastro(struct Cadastro** List, const char* CPF) {
      struct Cadastro* Current = *List;
      struct Cadastro* Previous = NULL;

   while (Current != NULL) {
      if (strcmp(Current -> CPF, CPF) == 0) {
      if (Previous == NULL) {
         *List = Current -> Next;
} else {
```

```
PREVIOUS -> NEXT = CURRENT -> NEXT;
}
FREE(CURRENT);

RETURN;

}

PREVIOUS = CURRENT;
CURRENT = CURRENT -> NEXT;

}

PRINTF("ALUNO NÃO ENCONTRADO.\n");

}
VOID UPDATECADASTRO(STRUCT CADASTRO* LIST, CONST CHAR*
CPF, CONST CHAR* NEWCURSO) {
    WHILE (LIST != NULL) {
        IF (STRCMP(LIST -> CPF, CPF) == 0) {
            STRCPY(LIST -> CURSO, NEWCURSO);

            RETURN;
    }
    LIST = LIST -> NEXT;
    }
    PRINTF("ALUNO NÃO ENCONTRADO.\n");
}
INT MAIN() {
    STRUCT CADASTRO * CADASTROLIST = NULL;
    CHAR ALUNO[50];
    CHAR CURSO[20];
    CHAR CPF[19];
```

data • •

S T Q Q S S D

```
CHAR MATRICULA [20];
CHAR TURNO [11];
INT CHOICE;
INT SCHOICE;
INT ZCHOICE;
INT SEGUNDOS = 1;

WHILE (1) {
    PRINTF("GERENCIAMENTO DE ALUNOS DA INSTITUICAO @NAOBO
:/N");
    PRINTF("1. CADASTRAR ALUNO \N");
    PRINTF("2. EXIBIR LISTA DE ALUNOS \N");
    PRINTF("3. EXCLUIR ALUNO DA BASE \N");
    PRINTF("4. MODIFICAR CURSO DO ALUNO \N");
    PRINTF("5. SAIR\N");
    PRINTF("ESCOLHA UMA OPCAO:");
    SCANF("%D", &CHOICE);
        GETCHAR(); // CONSUMIR O CARACTERE DE NOVA
LINHA SWITCH (CHOICE) {
    CASE 1:
        SYSTEM("CLS");

        PRINTF("* NOVO ALUNO: \N\N");

        PRINTF("NOME ");
        FGETS(ALUNO, SIZEOF(ALUNO), STDIN);
        ALUNO[STRCSPN(ALUNO, "\N")] = 0; // REMOVA O
CARACTERE DE NOVA LINHA

        PRINTF("CURSO");
        FGETS(CURSO, SIZEOF(CURSO), STDIN);
        CURSO[STRCSPN(CURSO, "\N")] = 0; // REMOVA O
        CARACTERE DE NOVA LINHA.
```

```
PRINTF("CPF DO ALUNO:: ");
FGETS(CPF, SIZEOF(CPF), STDIN);
CPF[STRCSPN(CPF, "\N")] = 0; // REMOVA O CARACTERE DE NOVA LINHA

PRINTF(" NOVA MATRICULA:");
FGETS(MATRICULA, SIZEOF(MATRICULA), STDIN);
MATRICULA[STRCSPN(MATRICULA, "\N")] = 0; // REMOVA O CARACTERE DE NOVA LINHA

PRINTF("\N\NQUAL O TURNO DO ALUNO? (FORNECE UMA DAS OPCOES ABAIXO: \N\N 1: MATUTINO\N 2: VESPERTINO\N 3: NOTURNO\N -");
    SCANF("%D", &ZCHOICE);

  SWITCH (ZCHOICE) {

    CASE 1:
      STRCPY(TURNO, "MATUTINO");
      ADDCADASTRO(&CADASTROLIST, ALUNO, CURSO, CPF, MATRICULA, TURNO);

      SYSTEM("CLS");
      PRINTF("ALUNO ADICIONADO COM SUCESSO!\N AGUARDE...");

      SLEEP(5SEGUNDOS);
      SYSTEM("CLS");
      SLEEP(5SEGUNDOS);
      BREAK;
```

data • •

S T Q Q S S D

```
Case 2

    StrCpy(Turno, "Vespertino");
    AddCadastro(&CadastroList, Aluno, Curso, CPF, Matricula,
Turno);

    System ("CLS");
    Printf("Aluno Adicionado Com Sucesso!\n Aguarde...");

    Sleep(Segundos);
    System("CLS");
    Sleep (Segundos);
    Break;

Case 3
    StrCpy(Turno, "Noturno");
    AddCadastro(&CadastroList, Aluno, Curso, CPF, Matricula,
Turno);

    System ("CLS");
    Printf ("Aluno Adicionado Com Sucesso!\n Aguarde...");

    Sleep (Segundos);
    System ("CLS");
    Sleep (Segundo));
    Break;

}
BREAK;

CASE 2:
```

```
        system("cls");
        printf("*Alunos matriculados*\n\n");
        ExibirCadastros(cadastroList);
        printf("Digite 1 e enter para retornar ao menu\n");

        switch (sChoice) {
            case 1:
                system("cls");
                break;
            default:
                system("cls");
                break;

        }
        break;

        case 3
            system("cls");
            printf("*Excluir Aluno da base de Dados*:\n
\n");
            fgets(cpf, sizeof(cpf), stdin);
            cpf[strcspn(cpf, "\n")] = 0; // Remove o caractere
de nova linha

            deleteCadastro(&cadastroList, cpf);
            system("cls");
            printf("Aguarde ...");
            Sleep(segundos);
            system("cls");
            printf("Aluno excluido com sucesso!\n\n\n Retornando
ao menu ...");
            Sleep(segundos);
            system("cls");
```

data

```
        printf("Aluno excluido com sucesso! \n\n\n Retornando
ao menu ...");

        Sleep(segundos);
        system("cls");

        break;
    case 4:
        system("cls");
        printf("Alterar Curso/Serie do Aluno: \n\n");
        printf("CPF do Aluno que recebera a atualizacao: ");
        fgets(cpf, sizeof(cpf), stdin);
        cpf[strcspn(cpf, "\n")] = 0; // Remove o caractere
de nova linha

        printf("Novo Curso/Serie a ser matriculado: ");
        fgets(curso, sizeof(curso), stdin);
        curso[strcspn(curso, "\n")] = 0; // Remova o
caractere de nova linha

        UpdateCadastro(cadastroList, cpf, curso);
        system("cls");
        printf("Aguarde ...");
        Sleep(segundos);
        system("cls");
        printf("Dados do Aluno Atualizado com Sucesso! \n");
        system("cls");
        break;

    case 5
        system("cls");
        printf("Encerrando Programa ...");
        Sleep(segundos);
```

```
system("cls");
printf("Aeronave ...");
sleep(segundos);
system("cls");
// liberar a memória alocada antes de sair while
(cadastroList) {
    struct Cadastro * temp = cadastroList;
    cadastroList = cadastroList -> next;
    free(temp);
}
exit(0);
default:
    printf("Escolha inválida. Tente novamente.\n");
}
}
return 0;
}
```