Algoritmos e Estruturas de Dados I

Prof. Lúcio Mauro Pereira

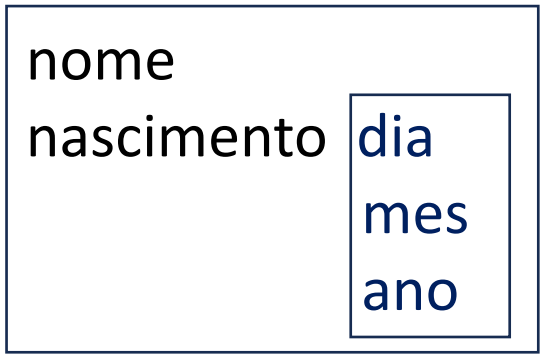
04/11/2024

Atividade solicitada:

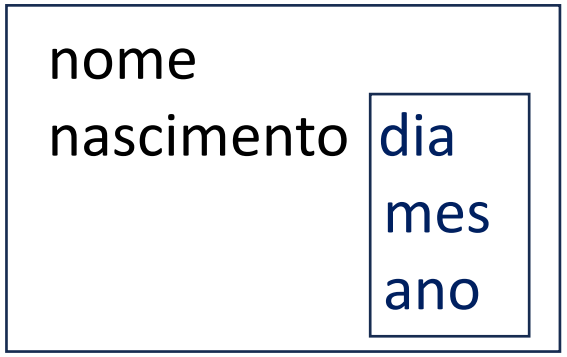
Tipo Abstrato de Dados Pessoa

Tipo Abstrato de Dados: Tipo Pessoa

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

typedef struct
        {
          char nome[50];
          Data nascimento;
        } Pessoa;
```



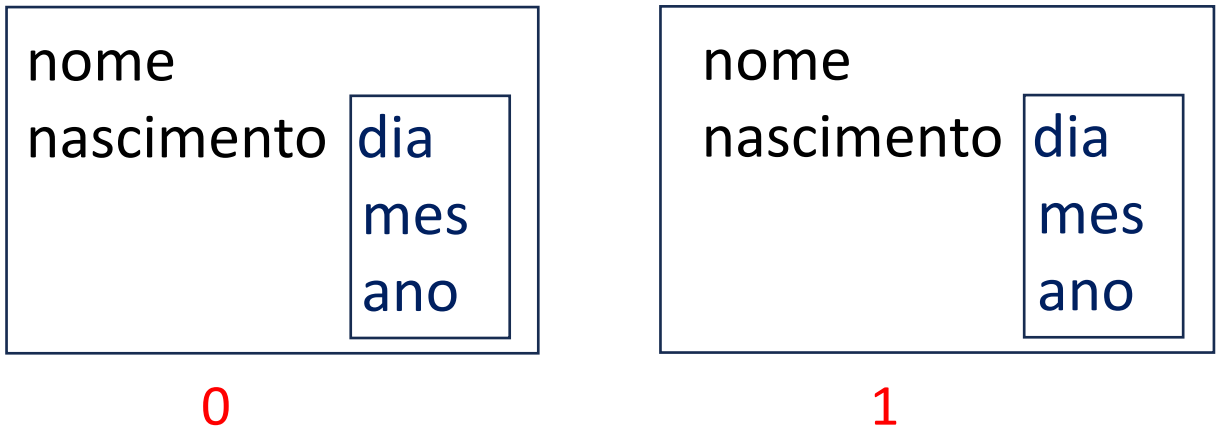
0



1

Tipo Abstrato de Dados: Tipo Pessoa

```
#include "data.h"

typedef struct
        {
          char nome[50];
          Data nascimento;
        } Pessoa;
```



Selecione (em Code::blocks)

File

New

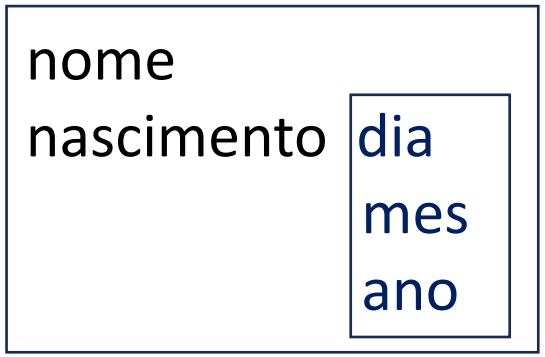
File

C/C++ header

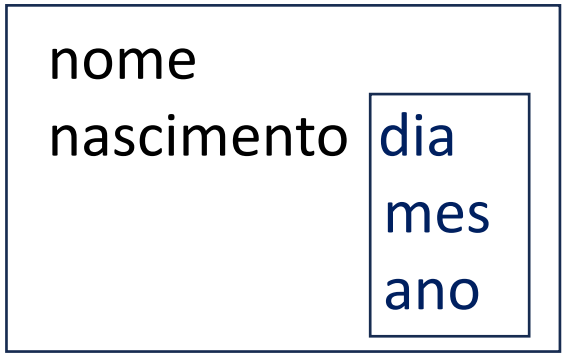
pessoa.h

Tipo Abstrato de Dados: Tipo Pessoa

```
#ifndef PESSOA_H_INCLUDED
#define PESSOA_H_INCLUDED

#include "data.h"

typedef struct
        {
          char nome[50];
          Data nascimento;
        } Pessoa;


#endif // PESSOA_H_INCLUDED
```



0

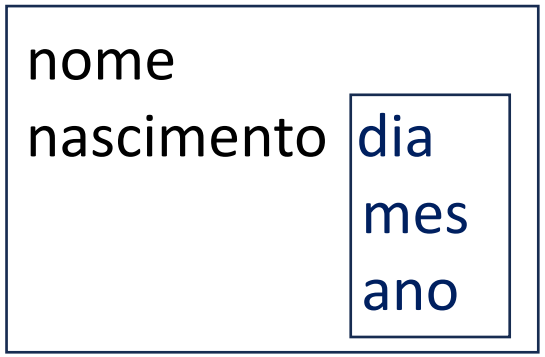


1

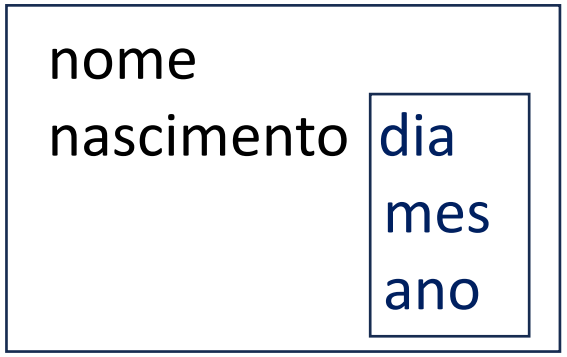
Tipo Pessoa: inserindo operações

Tipo Abstrato de Dados: Tipo Pessoa

```
#ifndef PESSOA_H_INCLUDED
#define PESSOA_H_INCLUDED

#include "data.h"
typedef struct {
        char nome[50];
        Data nascimento;
      } Pessoa;

void escrevaPessoa(Pessoa PESSOAS[], int i){
   escrevaEstPessoa( PESSOAS[i]);
}

#endif // PESSOA_H_INCLUDED
```



0

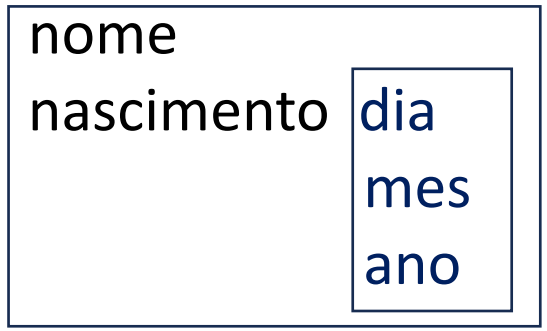


1

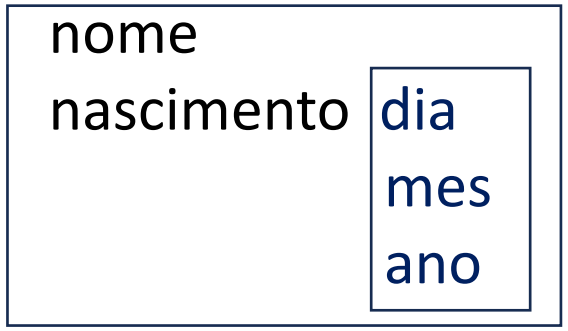
Tipo Abstrato de Dados: Tipo Pessoa

```
#ifndef PESSOA_H_INCLUDED
#define PESSOA_H_INCLUDED

#include "data.h"
typedef struct {
        char nome[50];
        Data nascimento;
      } Pessoa;


void escrevaPessoa(Pessoa PESSOAS[], int i){
   escrevaEstPessoa( PESSOAS[i]);
}
```

```
void escrevaEstPessoa(Pessoa PESSOA){
   printf("\nNome: %s“, PESSOA.nome );
   printf("\nData de Nascimento: %d/%d/%d “,
     PESSOA.nascimento.dia,
     PESSOA.nascimento.mes,
     PESSOA.nascimento.ano
   );
}
#endif // PESSOA_H_INCLUDED
```



0



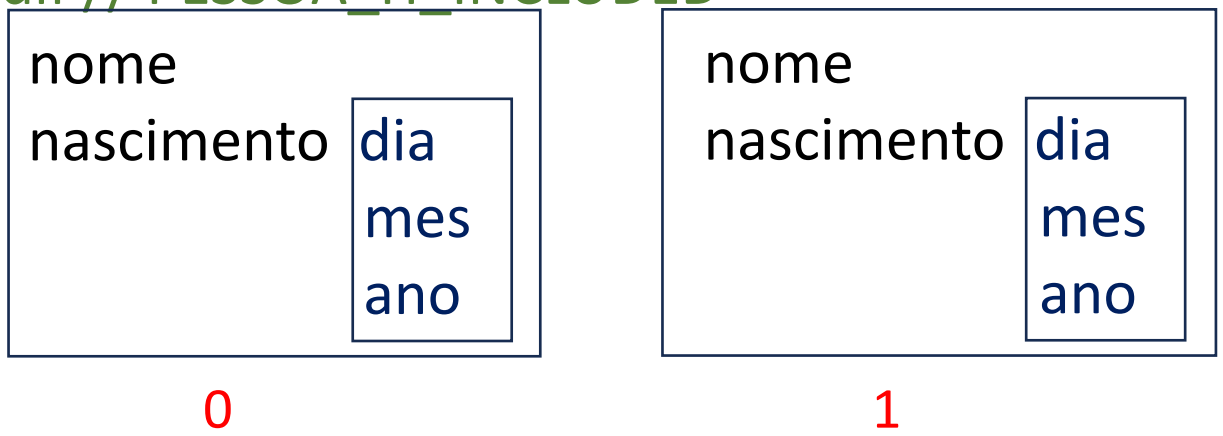
1

Tipo Pessoa: inserindo tamanho lógico

Tipo Abstrato de Dados: Tipo Pessoa

```
#ifndef PESSOA_H_INCLUDED
#define PESSOA_H_INCLUDED
#include "data.h"
typedef struct {
            char nome[50];
            Data nascimento;
          } Pessoa;

int TAM = 0;

void escrevaPessoa(Pessoa PESSOAS[], int i){
   escrevaEstPessoa( PESSOAS[i]);
}
```

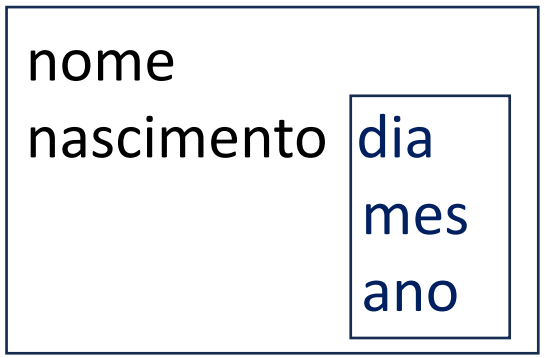
```
void escrevaEstPessoa(Pessoa PESSOA){
   printf("\nNome: %s“, PESSOA.nome );
   printf("\nData de Nascimento: %d/%d/%d “,
     PESSOA.Nascimento.dia,
     PESSOA.Nascimento.mes,
     PESSOA.Nascimento.ano
   );
}
#endif // PESSOA_H_INCLUDED
```



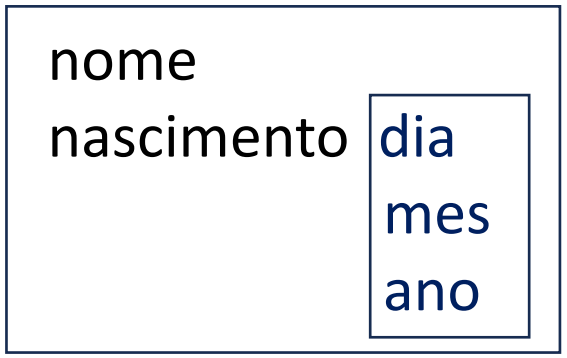
Tamanho lógico em arquivo

```
int tamanho(char* arq) {
    FILE* arqTamanho = fopen(arq,"rb");

    if(arqTamanho == NULL){
        arqTamanho = fopen(arq,"wb");
        TAM=0;
        fprintf(arqTamanho, "%i", TAM);
    }
    else {
        fscanf(arqTamanho, "%i", &TAM);
    }
    fclose(arqTamanho);
    return TAM;
}
```

```c
#ifndef PESSOA_H_INCLUDED
#define PESSOA_H_INCLUDED
#include "data.h"
typedef struct {
        char nome[50];
        Data nascimento;
      } Pessoa;
int TAM = 0; // ???
int tamanho(char* arq) {
   FILE* arqTamanho = fopen(arq,"rb");
   if(arqTamanho == NULL){
      arqTamanho = fopen(arq,"wb");
      TAM=0;
      fprintf(arqTamanho, "%i", TAM);
   } else {
      fscanf(arqTamanho, "%i", &TAM);
   }
   fclose(arqTamanho);
   return TAM;
}
void escrevaPessoa(Pessoa PESSOAS[], int i){
   escrevaEstPessoa( PESSOAS[i]);
}
void escrevaEstPessoa(Pessoa PESSOA){
   printf("\nNome: %s", PESSOA.nome );
   printf("\nData de Nascimento: %d/%d/%d ",
     PESSOA.Nascimento.dia,
     PESSOA.Nascimento.mes,
     PESSOA.Nascimento.ano
   );
}
#endif // PESSOA_H_INCLUDED
```



0



1

```c
#ifndef PESSOA_H_INCLUDED
#define PESSOA_H_INCLUDED
#include "data.h"
typedef struct {
        char nome[50];
        Data nascimento;
      } Pessoa;
int TAM = 0;
int tamanho(char* arq) {
   FILE* arqTamanho = fopen(arq,"rb");
   if(arqTamanho == NULL){
      arqTamanho = fopen(arq,"wb");
      TAM=0;
      fprintf(arqTamanho, "%i", TAM);
   } else {
      fscanf(arqTamanho, "%i", &TAM);
   }
   fclose(arqTamanho);
   return TAM;
}
void escrevaPessoa(Pessoa PESSOAS[], int i){
   escrevaEstPessoa( PESSOAS[i]);
}
void escrevaEstPessoa(Pessoa PESSOA){
   printf("\nNome: %s“, PESSOA.nome );
   printf("\nData de Nascimento: %d/%d/%d “,
     PESSOA.Nascimento.dia,
     PESSOA.Nascimento.mes,
     PESSOA.Nascimento.ano     );
}
void abertura(){
   printf("\nControle de Pessoas\n");
   TAM = tamanho("tamanhoArq.dat");
}
#endif // PESSOA_H_INCLUDED
```

```c
#ifndef PESSOA_H_INCLUDED
#define PESSOA_H_INCLUDED
#include "data.h"
typedef struct {
        char nome[50];
        Data nascimento;
      } Pessoa;
int TAM = 0;
int tamanho(char* arq) {
   FILE* arqTamanho = fopen(arq,"rb");
   if(arqTamanho == NULL){
      arqTamanho = fopen(arq,"wb");
      TAM=0;
      fprintf(arqTamanho, "%i", TAM);
   } else {
      fscanf(arqTamanho, "%i", &TAM);
   }
   fclose(arqTamanho);
   return TAM;
}
void escrevaPessoa(Pessoa PESSOAS[], int i){
   escrevaEstPessoa( PESSOAS[i]);
}
void escrevaEstPessoa(Pessoa PESSOA){
   printf("\nNome: %s“, PESSOA.nome );
   printf("\nData de Nascimento: %d/%d/%d “,
     PESSOA.Nascimento.dia,
     PESSOA.Nascimento.mes,
     PESSOA.Nascimento.ano    );
}
void abertura(){
   printf("\nControle de Pessoas\n");
   TAM = tamanho("tamanhoArq.dat");
}
void cadastrePessoa(Pessoa PESSOAS[]){
   fflush(stdin);  //Linux __fpurge(stdin);
   printf("\nNome: " );
   fgets(PESSOAS[TAM].nome , MAX_STR, stdin);
   printf("\nData de Nascimento [dd/mm/aaaa]: “);
   scanf(“%d/%d/%d”,
     &PESSOAS[TAM].Nascimento.dia,
     &PESSOAS[TAM].Nascimento.mes,
     &PESSOAS[TAM].Nascimento.ano
   );
   TAM++;
}
#endif // PESSOA_H_INCLUDED
```

```c
//Ao abrir – Por exemplo, em abertura()
void carregaPessoas(Pessoa PESSOAS[]) {
   FILE* arqPessoas = fopen("pessoas.dat", "rb+");

   if(arqPessoas == NULL){
      arqPessoas = fopen("pessoas.dat","wb+");
   }

    fread(PESSOAS, sizeof(Pessoa),TAM,arqPessoas);

    fclose(arqPessoas);
}
```

```c
//Ao fechar – Por exemplo, em despedida()
void gravaPessoas(Pessoa PESSOAS[]){
   FILE* arqPessoas = fopen("pessoas.dat", "wb");
   fwrite(PESSOAS, sizeof(Pessoa), TAM, arqPessoas);
   fclose(arqPessoas);

   FILE* arqTamanho = fopen("tamanhoArq.dat", "wb");
   fprintf(arqTamanho, "%i", TAM);
   fclose(arqTamanho);
}
```

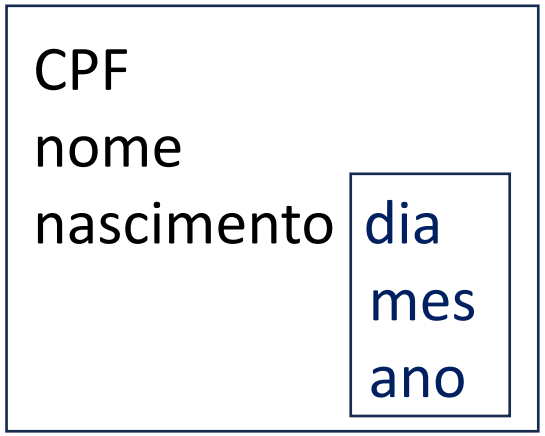
Trabalho Prático Final:

Chave Primária

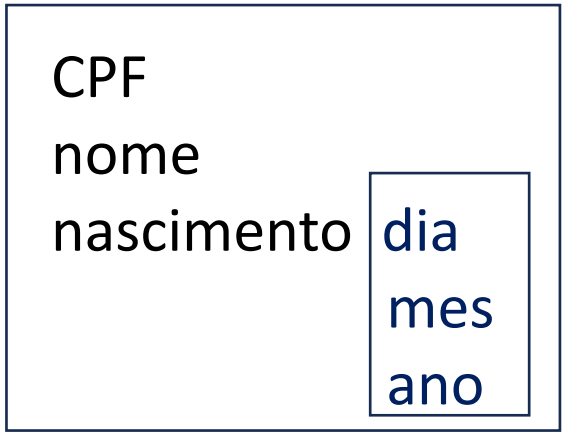
Chave Primária: identifica de forma única

Exemplos:

CPF
e-mail
celular
ID



0

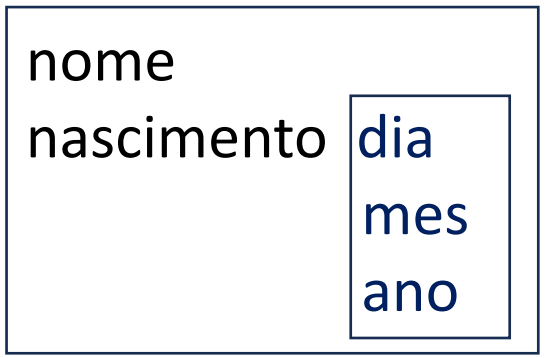


1

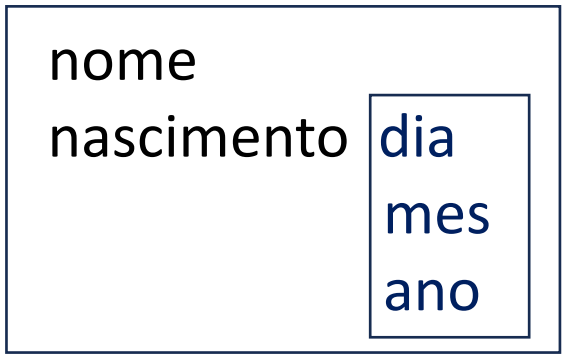
a) Inserir CPF na descrição de Pessoa

Tipo Abstrato de Dados: Tipo Pessoa

```
#include "data.h"

typedef struct
        {

          char nome[50];
          Data nascimento;
        } Pessoa;
```



0

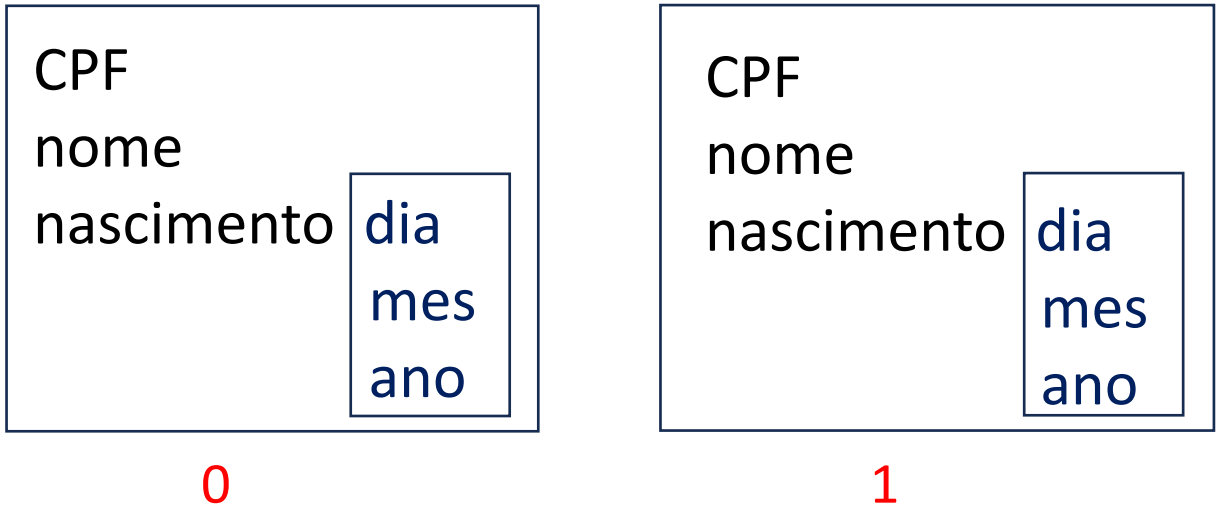


1

b) Inserir funcionalidade: Pesquisa pelo nome

Chave de Pesquisa: nome

```
void pesquisaPessoaNome( Pessoa PESSOAS[], char* nome )
{




}
```



Mais um pouco sobre a biblioteca *string.h*

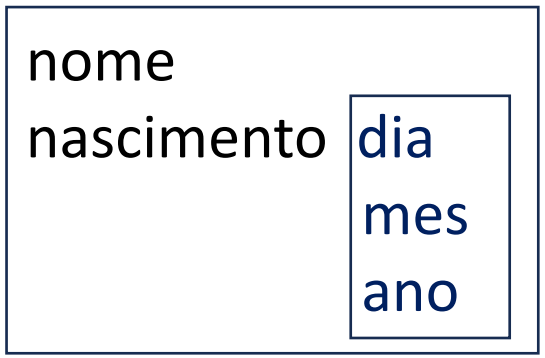
int strlen(*string*)

char* strcpy(*string1, string2*)

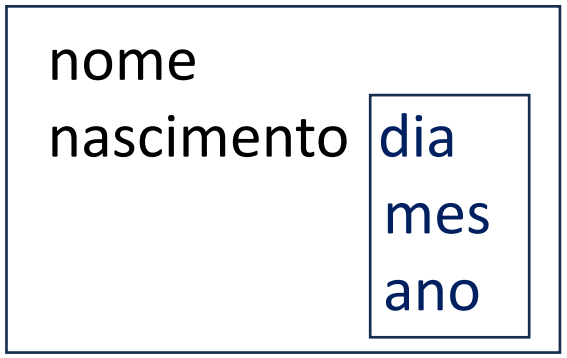
int strcmp(*string1, string2*)

Tipo Abstrato de Dados: Tipo Pessoa

```
#include "data.h"

typedef struct
        {

          char nome[50];
          Data nascimento;
        } Pessoa;
```



0

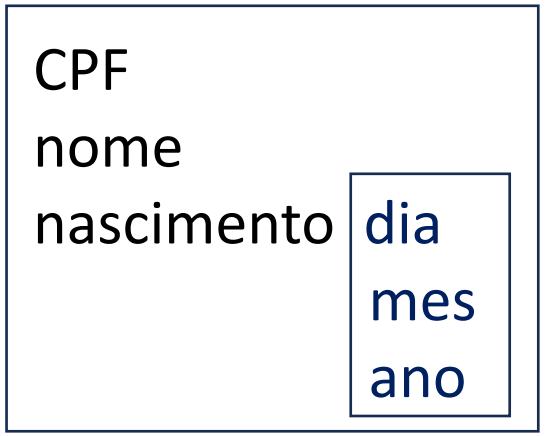


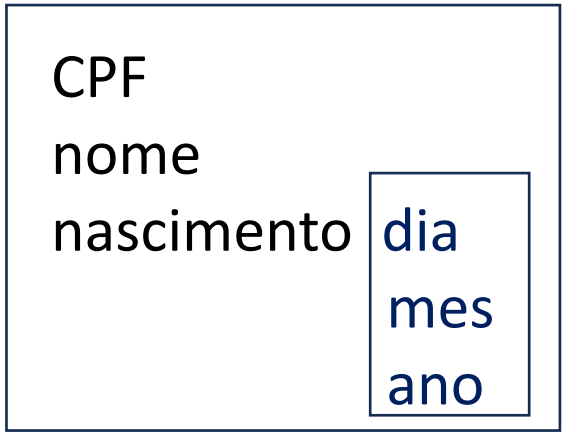
1

c) Inserir funcionalidade: Pesquisa pelo CPF

Chave de Pesquisa: nome

```
Pessoa* pesquisaPessoaCPF( Pessoa PESSOAS[], char* cpf )
{




}
```

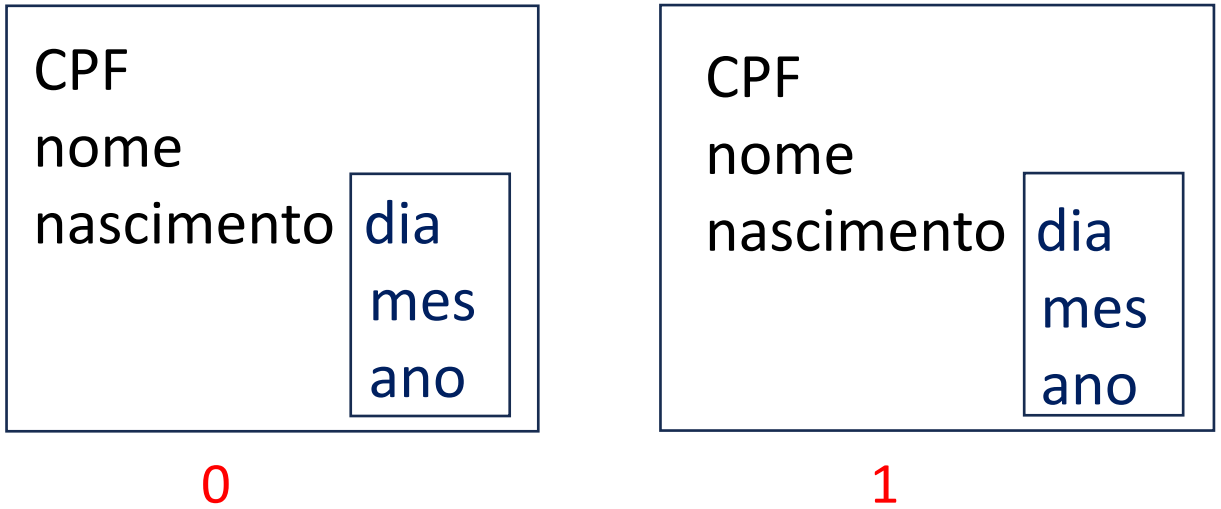




d) Planeje bem: inserir a funcionalidade Excluir – localizar pelo CPF

Exclusão. Chave: CPF

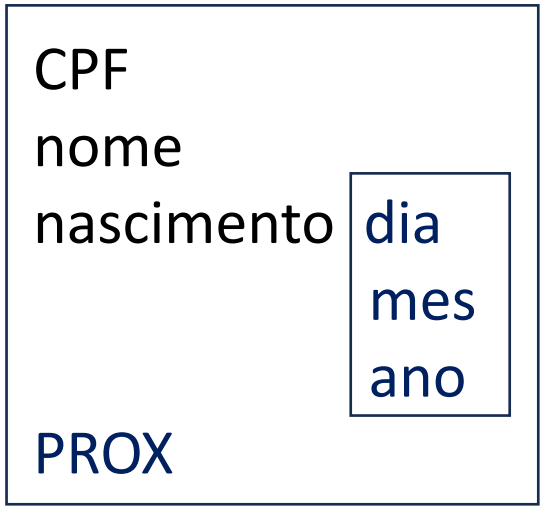
```
bool deletaPessoa ( Pessoa PESSOAS[], char* CPF )
{




}
```



Planeje: uma lista encadeada

Alocação dinâmica

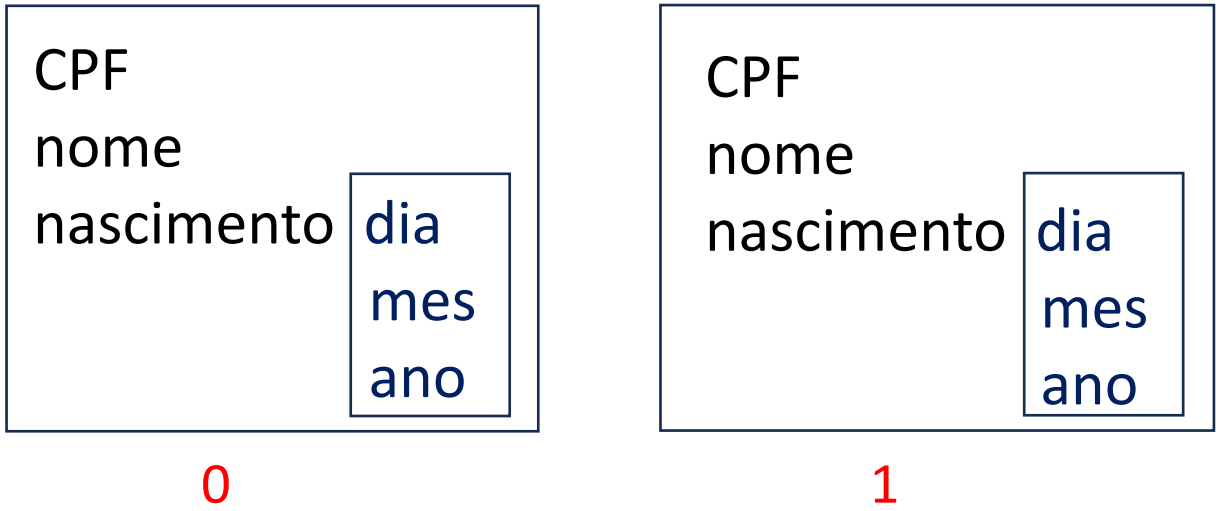
```
void listaPessoa ( Pessoa* LISTA)
{




}
```



Questões

Iterativo: listar pessoas

```
void listaPessoa ( Pessoa PESSOAS[])
{




}
```



Recursivo: listar pessoas

```
void listaPessoa ( Pessoa PESSOAS[], int tamanho)
{




}
```