Algoritmos e Estruturas de Dados I

Prof. Lúcio Mauro Pereira

28/10/2024

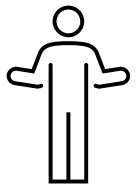
Aulas anteriores

Estruturas

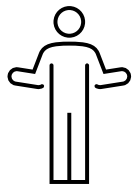
Abstração de Dados

```
const int MAX_STR =   50;
```

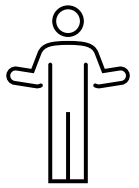
Modelando um novo tipo: Pessoa



Nome
Idade



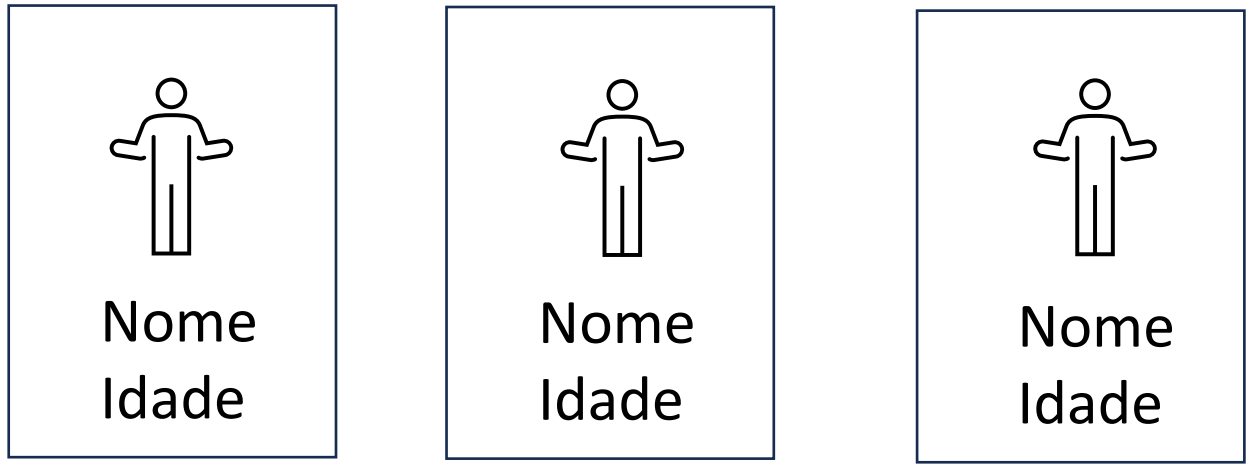
Nome
Idade



Nome
Idade

Estruturas

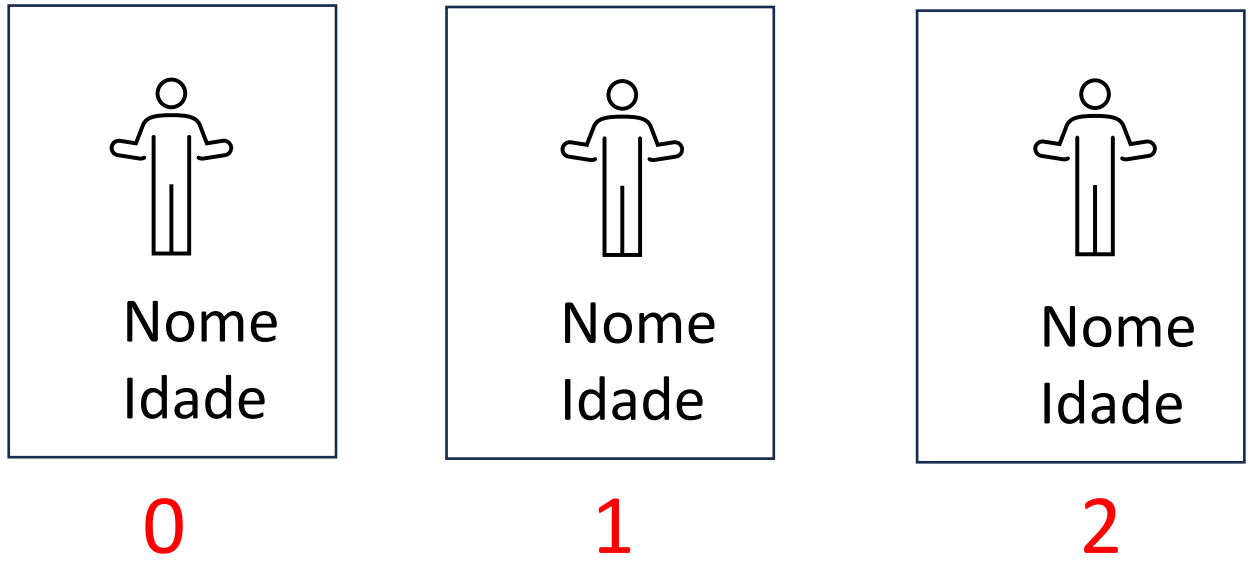
Instâncias de pessoas



Estruturas

Vetor de Pessoas

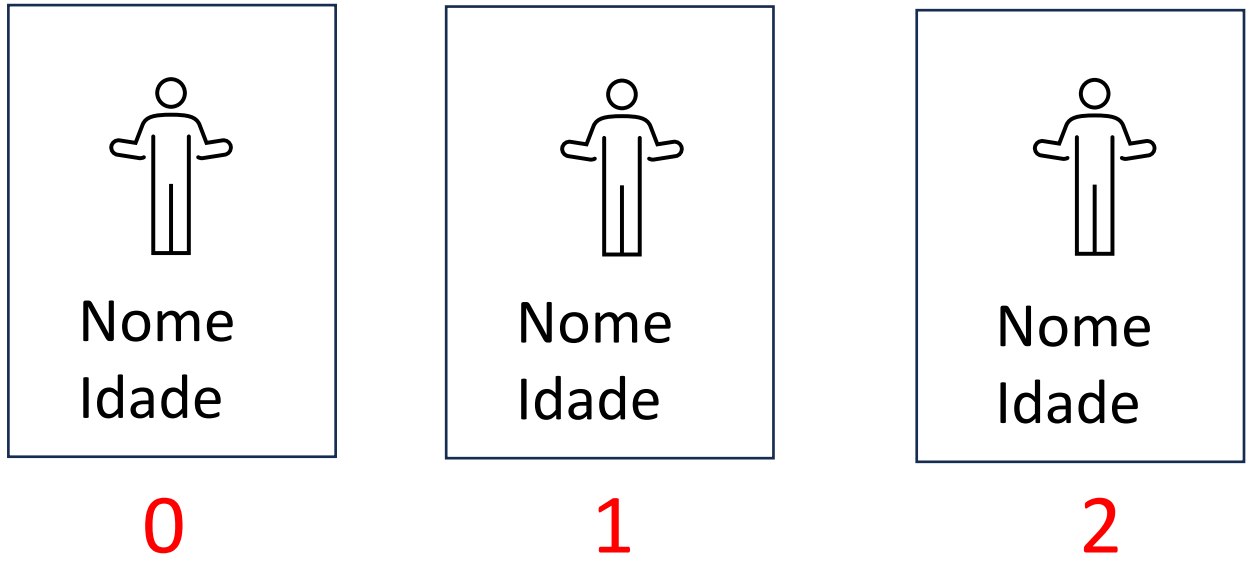
```
const int MAX_STR =   50;
const int       MAX = 100;
      int       TAM =   0;
```



Escrevendo Pessoas

```
void escrevePessoa(Pessoa PESSOAS[], int i){
    printf("\n\nNome: %s" , PESSOAS[i].nome );
    printf("\nIdade: %i“      , PESSOAS[i].idade );
}
```

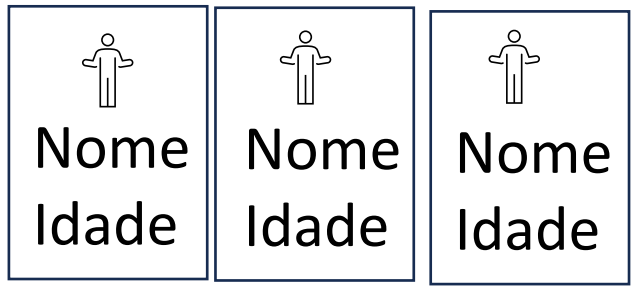
```
const int MAX_STR =   50;
const int         MAX = 100;
          int         TAM =     0;
```



Lendo uma Pessoa

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

   fflush(stdin);  //Linux __fpurge(stdin);
   printf("\nNome: " );
   fgets(PESSOAS[TAM].nome , MAX_STR, stdin);

   printf("\nIdade: “);
   scanf(“%i”, &PESSOAS[TAM].idade );

   TAM++;

}
```

```
const int MAX_STR =   50;
const int       MAX = 100;
      int       TAM =   0;
```



Em discussão:

Tipo Definido pelo Usuário

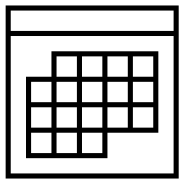
versus

Tipo Abstrato de Dados

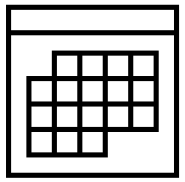
Estruturas

Abstração de Dados

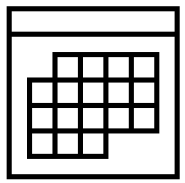
Modelando um novo tipo: Data



Dia
Mês
Ano



Dia
Mês
Ano



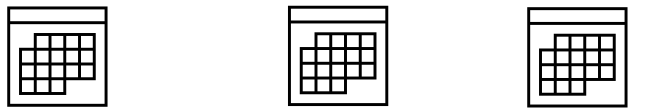
Dia
Mês
Ano

Estruturas

Abstração de Dados

Abstração de Operações

Modelando um Tipo Abstrato de Dados: Data



Dia
Mês
Ano

Dia
Mês
Ano

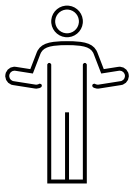
Dia
Mês
Ano

integer diasMes()
string mesExtenso()
boolean ehBissexto()
boolean dataValida()
string diaSemana()
...

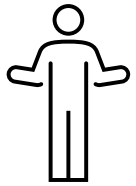
Remodelando o Tipo Pessoa

Pessoa: Nome e data de Nascimento

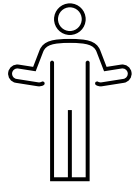
```
const int MAX_STR =   50;
```



Nome
dataNascimento



Nome
dataNascimento

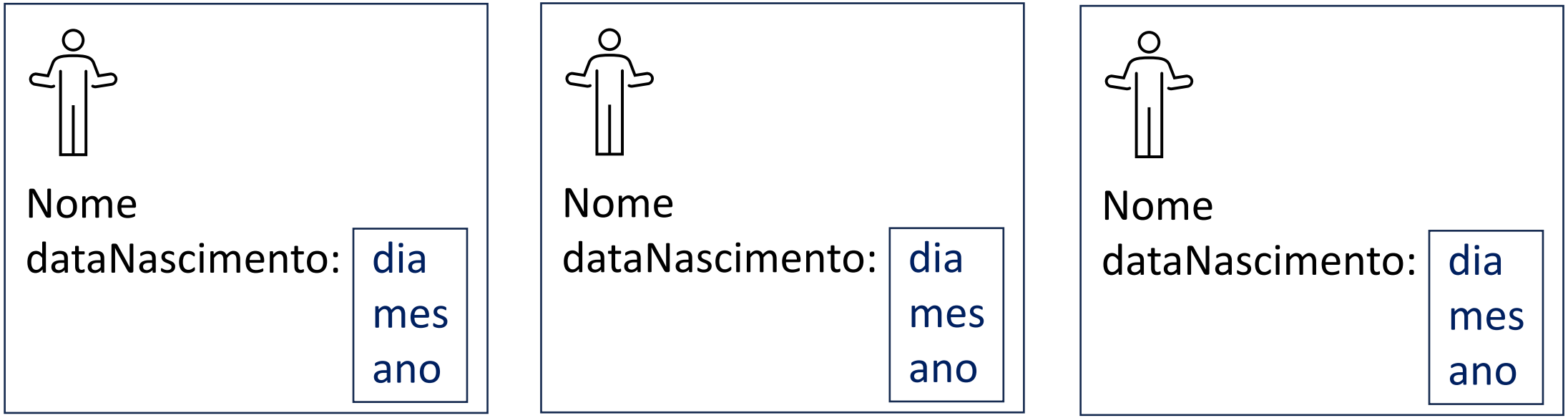


Nome
dataNascimento

Remodelando o Tipo Pessoa

Pessoa: Nome e data de Nascimento

```
const int MAX_STR =   50;
```



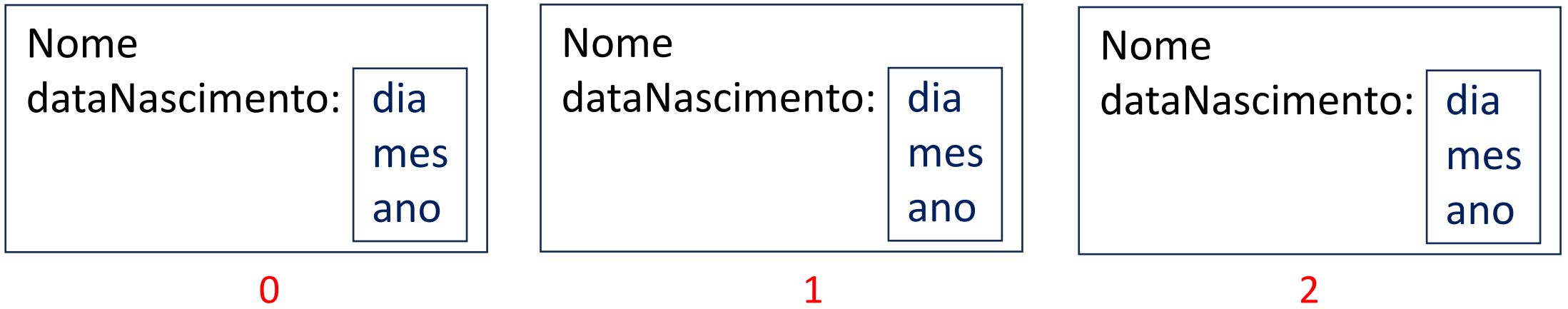
Remodelando o Tipo Pessoa

Um vetor de Pessoas

```
const int MAX_STR =  50;
const int     MAX = 100;
      int     TAM =   0;
```

Escrevendo Pessoas

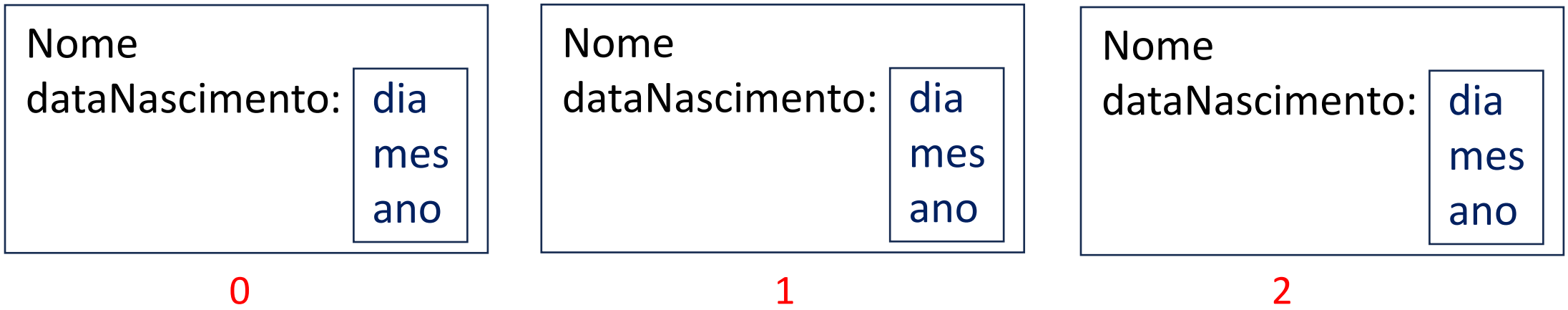
```
const int MAX_STR =   50;
const int       MAX = 100;
      int       TAM =   0;
```



Escrevendo Pessoas

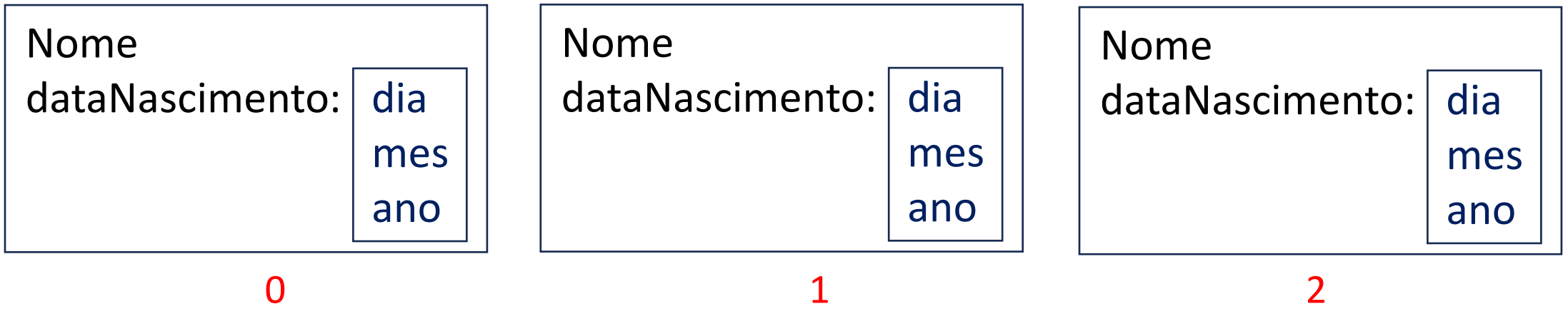
```
void escrevePessoa(Pessoa PESSOAS[], int i){
    printf("\n\nNome: %s" , PESSOAS[i].nome );
    printf("\nNascimento: %i“     , PESSOAS[i].idade );
}
```

```
const int MAX_STR =   50;
const int       MAX = 100;
      int       TAM =    0;
```



Lendo Pessoas

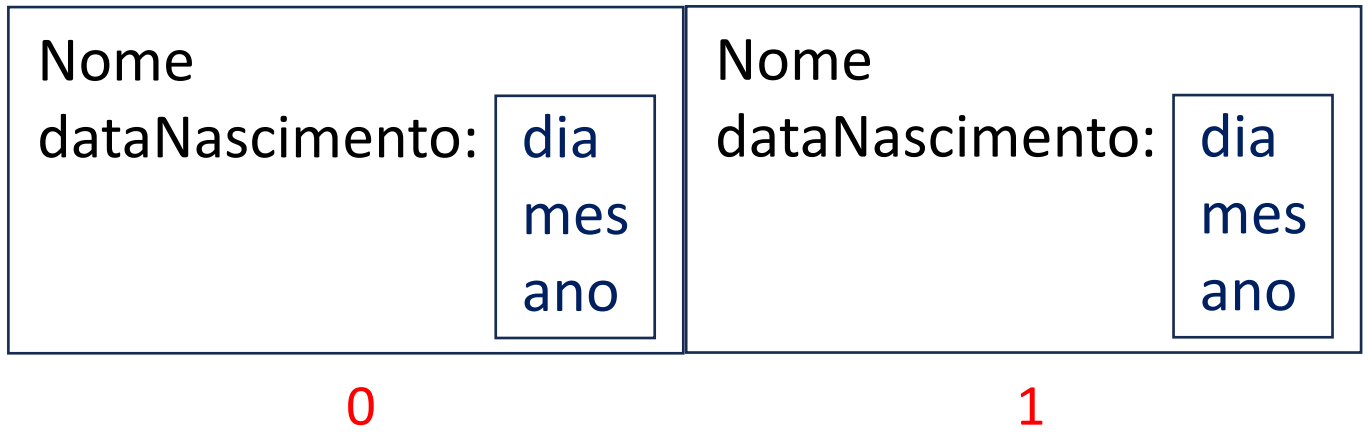
```
const int MAX_STR =   50;
const int       MAX = 100;
      int       TAM =   0;
```



Lendo Pessoas

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

    fflush(stdin);  //Linux __fpurge(stdin);
    printf("\nNome: " );
    fgets(PESSOAS[TAM].nome , MAX_STR, stdin);



    TAM++;

}
```

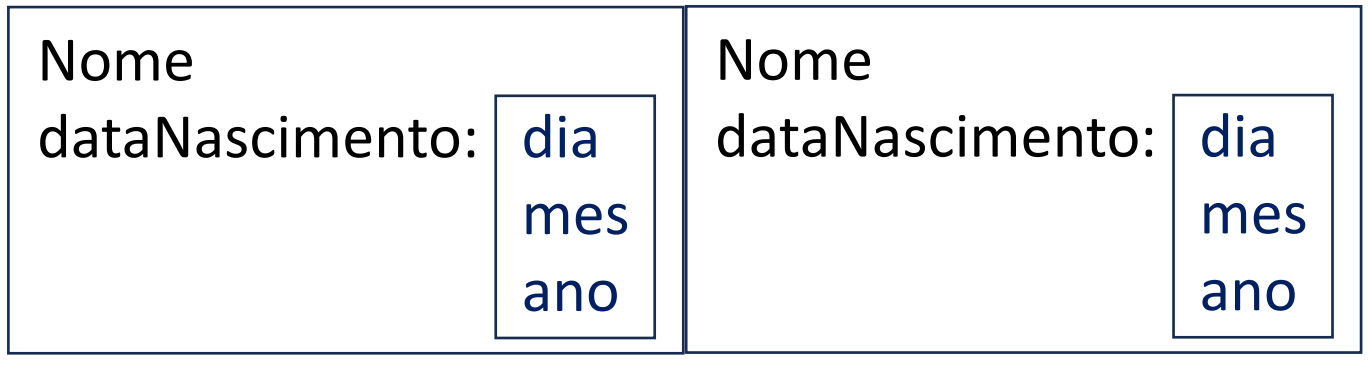
```
const int MAX_STR =   50;
const int       MAX = 100;
      int       TAM =    0;
```



Lendo Pessoas

```
void cadastrePessoa(Pessoa PESSOAS[]){
   fflush(stdin);  //Linux __fpurge(stdin);
   printf("\nNome: " );
   fgets(PESSOAS[TAM].nome , MAX_STR, stdin);

   printf("\nData de Nascimento: “);
   printf("\n\tDia: “);
   scanf(“%d”, &PESSOAS[TAM].dataNascimento.dia);

   TAM++;

}
```

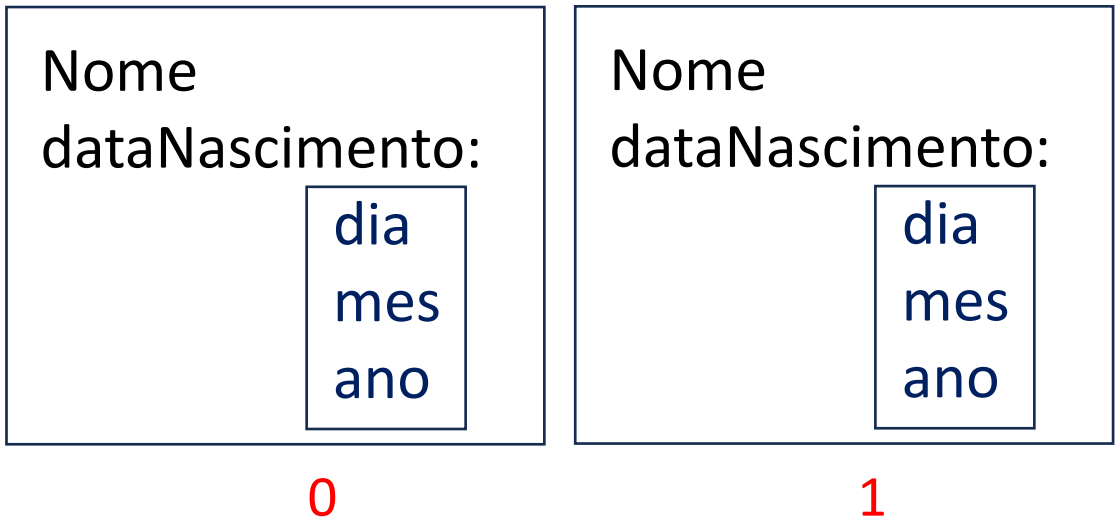
```
const int MAX_STR =   50;
const int        MAX = 100;
      int        TAM =    0;
```



Lendo Pessoas

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

   fflush(stdin);  //Linux __fpurge(stdin);
   printf("\nNome: " );
   fgets(PESSOAS[TAM].nome , MAX_STR, stdin);

   printf("\nData de Nascimento: “);
   scanf(“%d/%d/%d”,
      &PESSOAS[TAM].dataNascimento.dia,
      &PESSOAS[TAM].dataNascimento.mes,
      &PESSOAS[TAM].dataNascimento.ano
   );
   TAM++;
}
```

```
const int MAX_STR =   50;
const int     MAX = 100;
      int     TAM =    0;
```



Trabalho Prático Final

Modelagem de dados

Uma Pessoa é descrita por:

- Seu nome
- Sua data de nascimento: dia/mês/ano

Funcionalidades providas

Menu de opções:

0 – Sair

1 – Cadastrar uma pessoa

2 – Listar pessoas

Requisitos

O tamanho lógico deve ser um dado persistente

Um arquivo deve armazenar o tamanho

Os dados das pessoas devem ser dados persistentes

Um vetor de pessoas deve ser carregado ao iniciar

O arquivo que armazena pessoas deve ser atualizado ao encerrar

Desafios

Tente

Construa uma função que receba um vetor de triângulos e o seu tamanho e calcule o número de triângulos equiláteros nele presentes.

Um triângulo é descrito por seus três lados (tipo real):

A

B

C