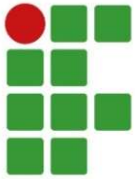


INSTITUTO FEDERAL
São Paulo
Câmpus São Carlos

Algoritmos e Programação II

Variáveis compostas heterogêneas: Registros

Profa. Dra. Eloize Seno

Registros (ou Estruturas)

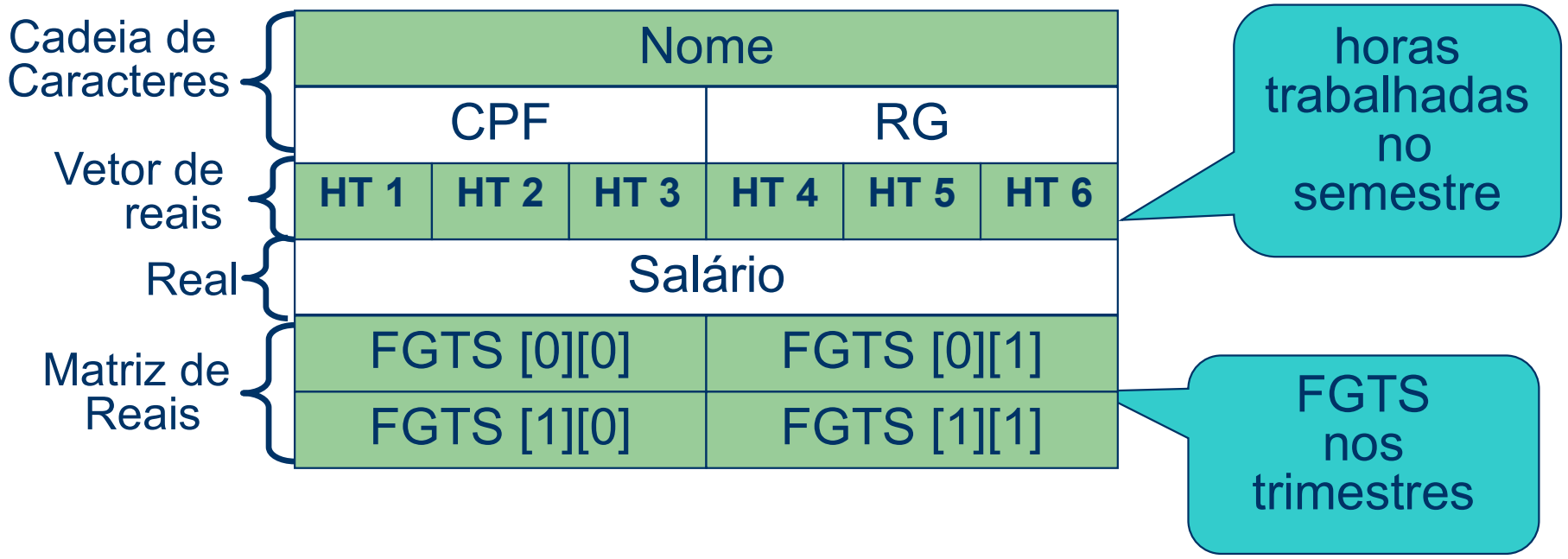
- São estruturas de dados capazes de armazenar várias informações.
- Cada informação é chamada de **campo**.
- Os campos podem ser de diferentes tipos ou, ainda, podem representar **outros registros**.
- Os registros podem gerar novos tipos de dados, não se limitando apenas aos tipos primitivos da linguagem C.

Exemplo de Registro

- Registro de pagamento de funcionário
- Quais dados?
  - Nome, RG, CPF, salário, horas trabalhadas por mês em um semestre, FGTS acumulado em cada trimestre, etc...
- Que tipo de dado usar?
  - Uma variável para cada dado?

Exemplo de Registro (cont.)

- Registro de pagamento de funcionário



Registros (ou Estruturas)

- Cada campo deve ter um nome e deve ser referenciado por esse nome
- Cada campo é uma variável ou constante

ATENÇÃO: Não confundir com *array* onde todos os elementos são do mesmo tipo e são referenciados por um índice.

Declaração de registros

- Sintaxe:

```
struct nome_do_registro
{ tipo campo1;
  tipo campo2;
  ...
  tipo campoN;
};
```

- A estrutura definida passa a representar um novo tipo de dado chamado ***nome_do_registro***, capaz de armazenar várias informações de tipos diferentes.

Exemplo de registro

```
struct Pessoa
{ int idade;
  char nome[40];
  char sexo;
};
```

Pessoa passa a ser um tipo de dado

INSTITUTO FEDERAL
São Paulo
Câmpus São Carlos

Declaração de Variáveis do Tipo Definido

- Para que um programa utilize uma **struct** é necessário declarar variáveis desse tipo.

- Sintaxe:

```
struct nome_do_registro nome_da_variável;
```

- Exemplo:

```
struct Pessoa p1, p2;
```

p1 e **p2** são variáveis do tipo **Pessoa** que contêm os campos idade, nome e sexo.

Outro exemplo

- Registro de pagamento de funcionário:

```
struct Reg_pagto
{  char nome[30];
   char CPF[13], RG[10];
   float horas_trab[6], salario;
   float FGTS[2][2];
};
```

Nome					
CPF			RG		
HT 1	HT 2	HT 3	HT 4	HT 5	HT 6
Salário					
FGTS [0][0]			FGTS [0][1]		
FGTS [1][0]			FGTS [1][1]		

Declaração de Variáveis do Tipo Definido (cont.)

- A declaração de variáveis também pode ser feita na definição da própria estrutura:
- Exemplos:

```
struct Pessoa
{  int idade;
   char nome[40];
   char sexo;
}Paulo,Tereza;
```

```
struct Reg_pagto
{  char nome[30];
   char CPF[13], RG[10];
   float horas_trab[6];
   float salario;
   float FGTS[2][2];
} func1, func2;
```

Acesso aos campos

- O acesso a cada campo é feito por meio do operador ponto (.)
- Sintaxe: `nome_do_registro.nome_do_campo`
- Exemplo:

```
struct Pessoa
{  int idade;
   char nome[40];
   char sexo;
};
```

```
int main()
{
    struct Pessoa p;
    p.idade = 28;
    strcpy(p.nome,"Joao da Silva");
    p.sexo = 'M';
}
```

Outro exemplo

- Exemplos de uso (na atribuição):

```
strcpy(func1.nome, "Ana");
func1.salario = 1.000;
func1.horas_trab[0] = 120.4;
func1.FGTS[0][0] = 80.00;
```

```
struct Reg_pagto
{  char nome[30];
   char CPF[13], RG[10];
   float horas_trab[6];
   float salario;
   float FGTS[2][2];
} func1, func2;
```

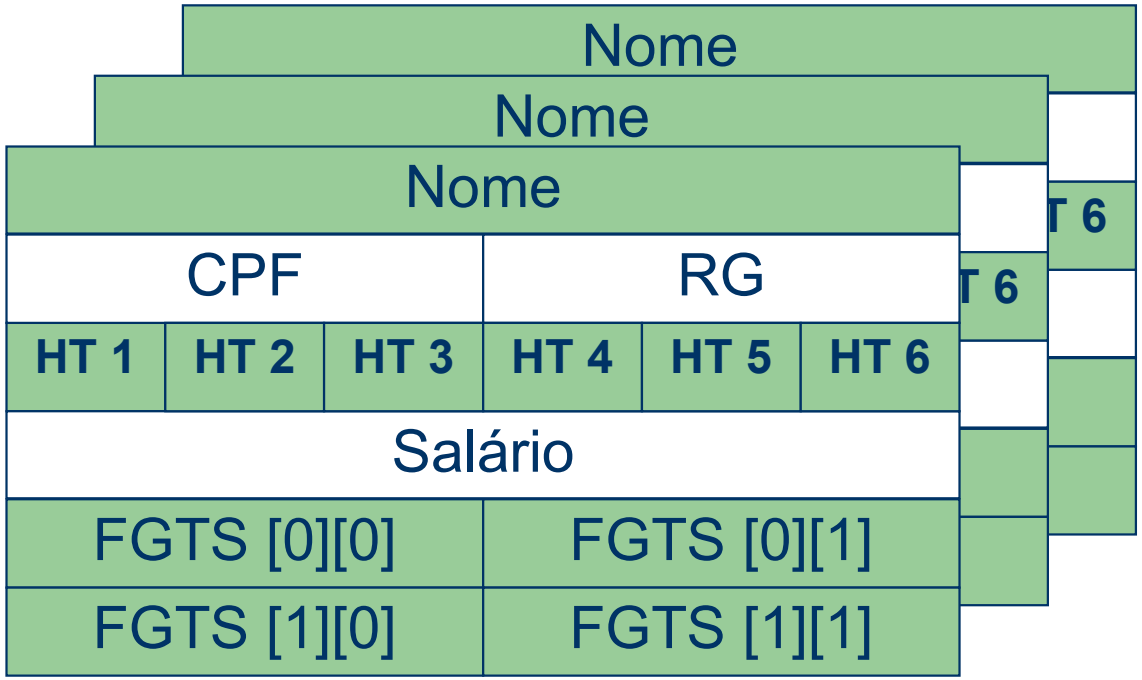
Exercício de fixação (1)

- Defina um registro de Funcionário capaz de armazenar o código, o cargo, o nome, o número de dependentes e o salário de um funcionário.
- Crie um funcionário do tipo do registro criado e solicite seus dados, preenchendo o registro.
- Imprima todos os dados do funcionário criado.
- Solicite do usuário um novo cargo e um novo salário e atualize o registro criado.
- Imprima novamente o registro.

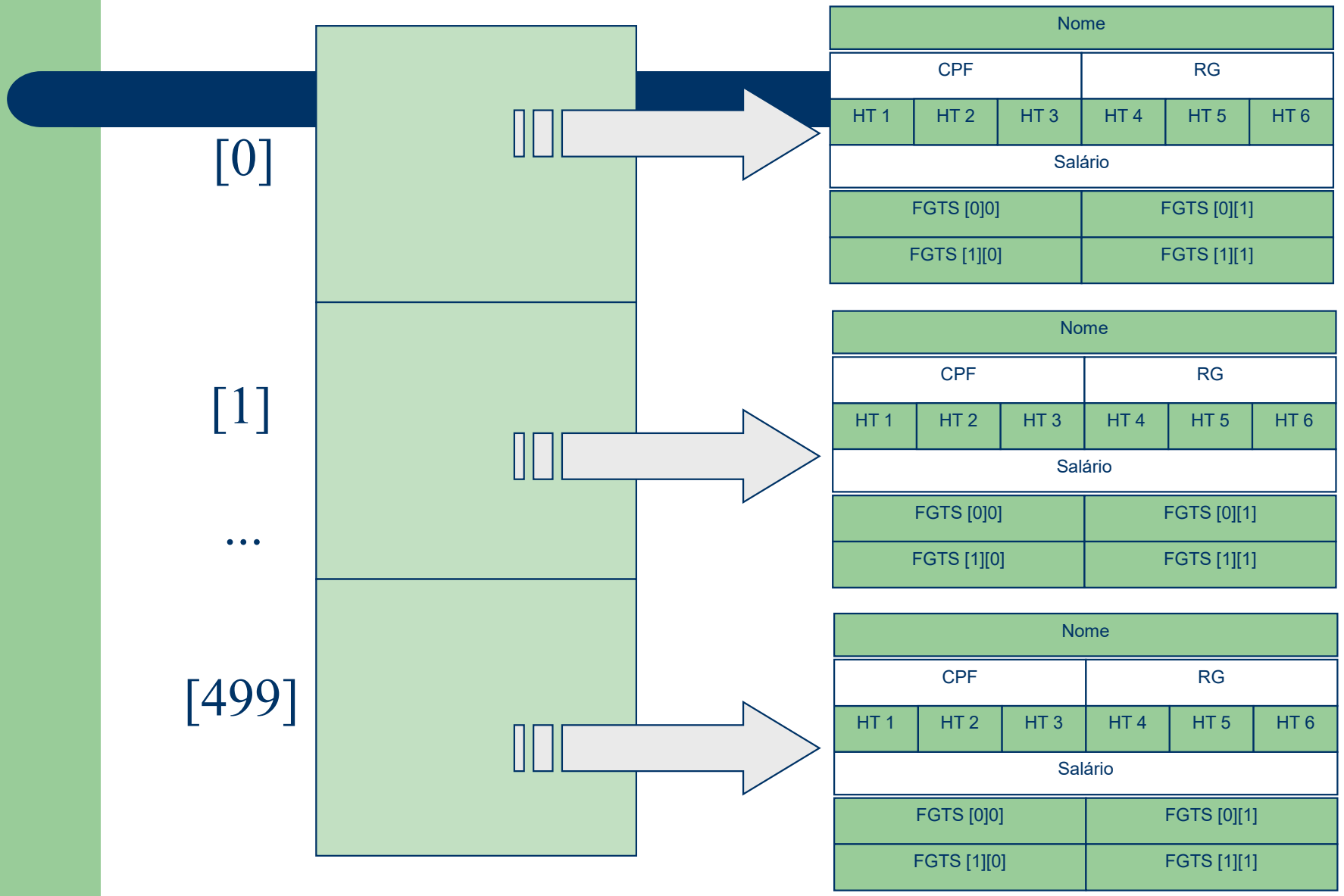
Vetor de registros

- E se tivermos que tratar os registros de 500 funcionários?
  - Declaramos 500 variáveis do tipo registro?

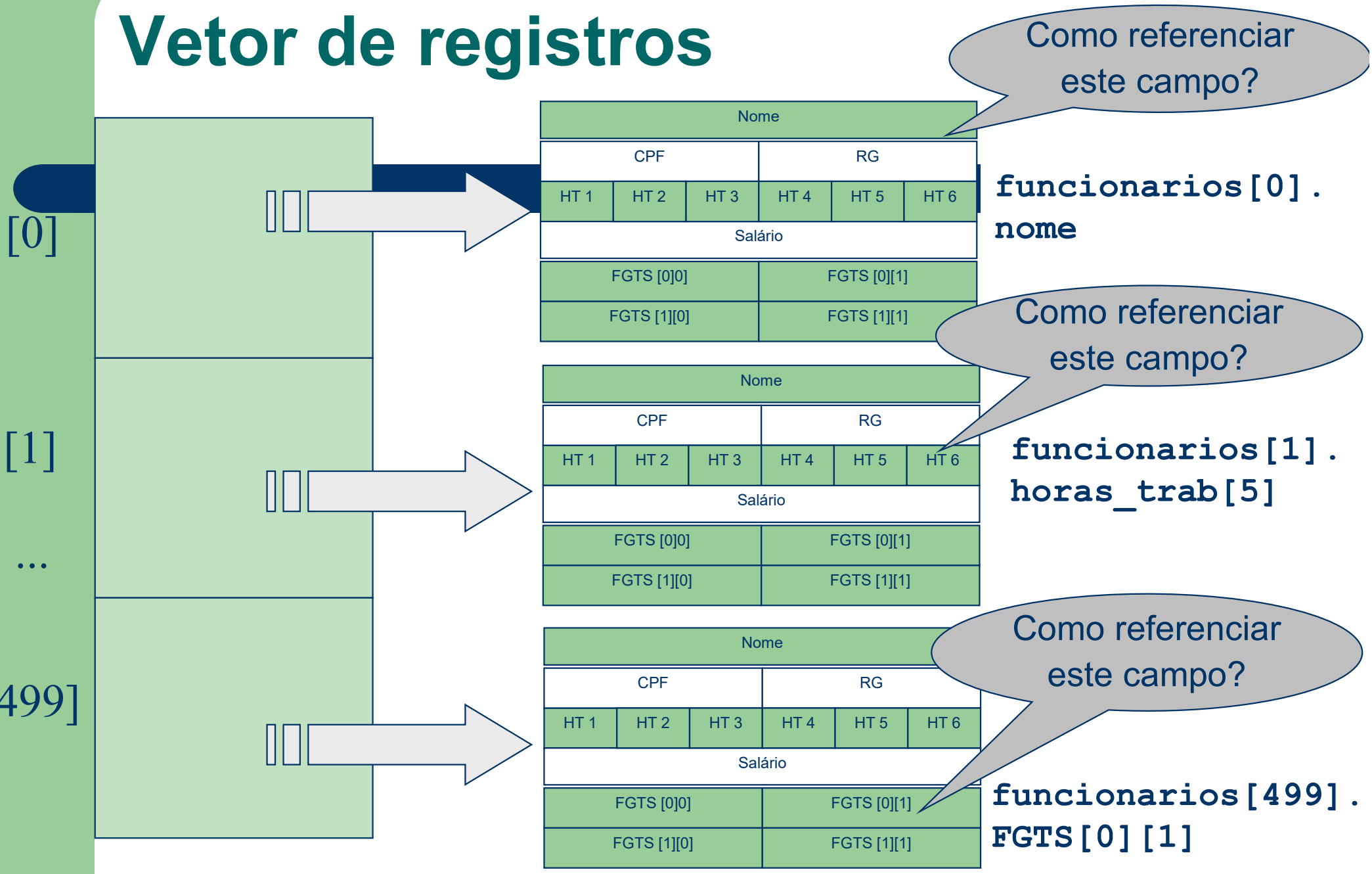
SOLUÇÃO: criar um vetor de registros !



Vetor de registros



Vetor de registros



Declaração de Variáveis do Tipo Registro (cont.)

- Além de variáveis simples, variáveis compostas como os vetores e as matrizes também podem ser declaradas como **struct**

- Exemplo:

```
struct Pessoa vet_pessoas[50];
struct Pessoa mat_pessoas[10][10];
struct Reg_pagto funcionarios[500];
```

Declaração de struct com typedef

- Sintaxe:

```
typedef struct {
  tipo campo1;
  tipo campo2;
  ...
  tipo campoN;
} nome_do_registro;
```

Exemplo:

```
typedef struct
{   int codigo;
    char nome[40];
    float salario;
} funcionario;
```

Uso de dado definido como struct

- Uso equivalente aos demais tipos de dados: não é preciso usar a palavra **struct**

- Exemplo:

```
funcionario func;
funcionario vet_func[10];
```

- Em uma função:

```
funcionario maiorSalario(funcionario
  vet_func[], int n)
{
  //código da funcao...
}
```

Redefinindo tipos com typedef

- Sintaxe:

  typedef tipo_C identificador;

Palavra-chave

int
float
char
struct

Redefinindo tipos com typedef (cont.)

- Exemplo:

```
typedef int idade;
typedef float vetor[100];
typedef char string[30];
int main(void)
{
    idade i = 10;
    vetor v = {1,2,3,4};
    string s1 = "teste";
    ...
}
```

Obs: **typedef** não cria um novo tipo. Apenas permite que determinado tipo seja denominado de forma diferente.

Exercício de fixação (2)

- A prefeitura de uma cidade fez uma pesquisa entre seus habitantes, coletando dados sobre salário, sexo, idade e numero de filhos.
- Crie um registro capaz de armazenar os dados de um habitante e defina um vetor com capacidade para até 20 habitantes.
- Crie um programa para ler os dados de cada habitante, calcular e mostrar:
  - A média de salários da população;
  - O número médio de filhos;
  - O maior salário e o menor salário;
  - O percentual de mulheres com salário superior a R$1.500,00.