Exercício

Pretende-se um programa para efetuar a gestão dos funcionários de uma empresa.

Considerando a seguinte declaração:

```
typedef struct {
  int numero;
  char nome[80];
  float venc;
}tipoFunc;
```

a) Defina as estrutura de dados para armazenar a informação referente ao número de funcionários da empresa e aos seus dados.
Considere que a empresa tem no máximo 50 funcionários.

b) Elabore um programa com seguintes opções:

1 - Listar dados de todos os funcionários;

2 - Calcular média de vencimentos;

3 - Procurar dados de um funcionário;

4 - Acrescentar dados de um funcionário;

5 - Remover dados de um funcionário;

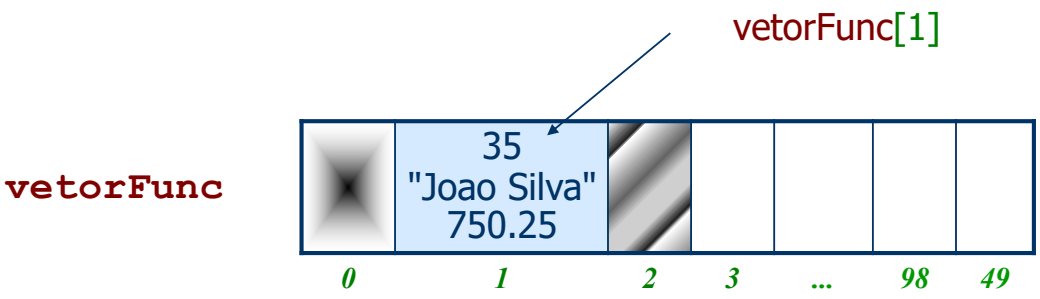
Considerando a existência das seguintes funções

```
void limpaBufferStdin(void);
int lerInteiro(int min, int max);
float lerFloat(float min, float max);
void lerString(char vetor[], int max);
void escreveFunc(tipoFunc func);
tipoFunc leDadosFunc(void);
int menu(void);
```

elabore as funções adicionais necessárias e o programa requerido.

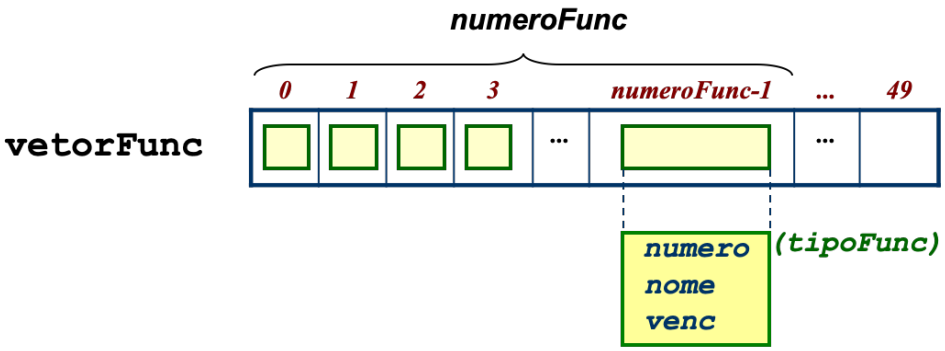
Exercício: resolução (0)

- Estruturas de dados (vetor de estruturas)



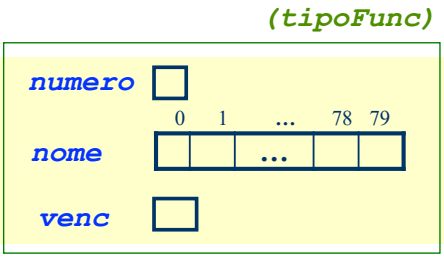
Exercício: resolução (I)

a)
```
#define MAXFUNC 50
int numeroFunc=0;  /* declara e inicializa (a zero) variável
                      para armazenar número de funcionários */


tipoFunc vetorFunc[MAXFUNC];       /* Vetor para armazenar
                                   dados dos funcionários – máximo 50
                                   (alocação estática de memória) */
```



Exercício: resolução (Ia)

```
/* main() versão inicial (estruturas de dados) */

#include <stdio.h>
#include <string.h>

#define MAXFUNC 50

typedef struct {
  int numero;
  char nome[80];
  float venc;
}tipoFunc;
```

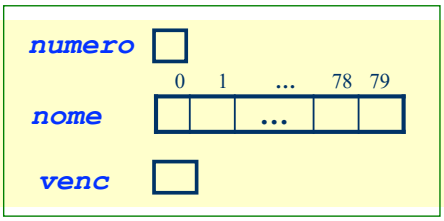



```
int main(void){
   tipoFunc vetorFunc[MAXFUNC]; /* Declara vetor  */
    int numeroFunc;
      numeroFunc = 0;           /* inicia contador */
        /* Código para programa */
 return 0;
}
```

Exercício: resolução (II)

```
/* main() versão inicial (menu e estruturas de dados) */

#include <stdio.h>
#include <string.h>

#define MAXFUNC 50

typedef struct {
  int numero;
  char nome[80];
  float venc;
}tipoFunc;

void limpaBufferStdin(void);
int lerInteiro(int min, int max);
float lerFloat(float min, float max);
void lerString(char vetor[], int max);
int menu();
```

(tipofunc)



Exercício: resolução (III)

```
int main(void){
  tipoFunc vetorFunc[MAXFUNC];   /* Declara vetor estático */
  int numeroFunc, op;
  numeroFunc = 0;          /* inicia número de funcionários */
  do {
    op = menu();
    switch (op) {
      case 1:    /* TO DO - Listar dados de todos os funcionários */
            break;
      case 2:    /* TO DO - Calcular média de vencimentos */
            break;
      case 3:    /* TO DO - Acrescentar dados de um funcionário */
            break;
      case 4:    /* TO DO - Remover dados de um funcionário */
            break;
    } /* switch */
  }while(op!=0);
  return 0;
}
```

Exercício: resolução (IVa)

```
int menu(void){
   int opcao;

      printf("\n 1 - Lista Func ");
      printf("\n 2 - Media vencimentos");
      printf("\n 3 - Acrescenta Func ");
      printf("\n 4 - Remove Func");
      printf("\n 0 - Sair");
      printf("\n\n Escolha uma opcao (0-4): ");

      do {                          /* Executar na função lerInteiro() */
        scanf ("%d", &opcao);
      } while (opcao < 0 || opcao > 4);

    return opcao;
}
```


Exercício: resolução (IVb)

```
int menu(void){
   int opcao;

      printf("\n 1 - Lista Func ");
      printf("\n 2 - Media vencimentos");
      printf("\n 3 - Acrescenta Func ");
      printf("\n 4 - Remove Func");
      printf("\n 0 - Sair");
      printf("\n\n Escolha uma opcao: ");

      opcao = lerInteiro(0,4);

    return opcao;
}
```

Exercício: resolução (V)

Considera-se a seguinte implementação para as funções existentes

```
int lerInteiro_v0(int min, int max){/* Le e devolve valor no intervalo desejado (min -> max) */
   int numero;

   do{       /* Repete leitura enquanto valor introduzido não estiver no intervalo desejado (min -> max) */
        scanf("%d", &numero);
        limpaBufferStdin();    /* limpa dados do buffer stdin */
   } while (numero<min || numero>max);

   return numero;
}
```

```
/* Situação não prevista
Validação de tipo de dados introduzido (por exemplo se utilizador não introduzir um valor numérico) */
```

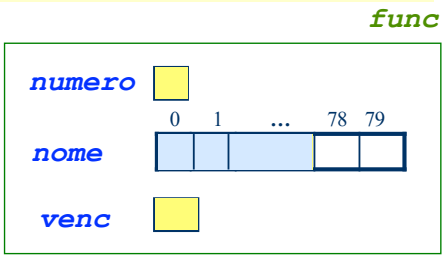
```
void limpaBufferStdin(void){       /* Chamar após leituras de dados através do teclado */
   char lixo;
  do{
        lixo=getchar();
   }while (lixo!='\n' && lixo!=EOF);
}
```


Exercício: resolução (VI)

```
int lerInteiro(int min, int max){
   int numero, controlo;

   do{ // Repete leitura enquanto valor introduzido não for numérico e não estiver no intervalo desejado (min -> max)
        controlo = scanf("%d", &numero); // scanf devolve quantidade de valores válidos obtidos
        limpaBufferStdin();     // limpa dados do buffer stdin
   } while (numero<min || numero>max || controlo==0);
   return numero;
}
float lerFloat(float min, float max){
   float numero;
   int controlo;

   do{ // Repete leitura enquanto valor introduzido não for numérico e não estiver no intervalo desejado (min -> max)
        controlo = scanf("%f", &numero); // scanf devolve quantidade de valores válidos obtidos
        limpaBufferStdin();     // limpa dados do buffer stdin
   } while (numero<min || numero>max || controlo==0);
   return numero;
}
```

Exercício: resolução (VII)

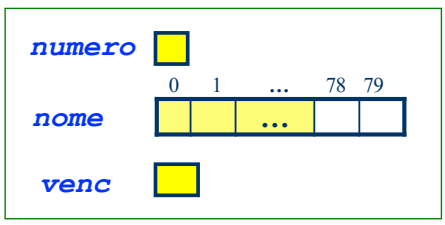
```
tipoFunc leDadosFunc(void){
   tipoFunc func;

   printf("Numero: ");
   func.numero = lerInteiro(1,100);

   printf("Nome: ");
   fgets(func.nome,80,stdin);

   printf("Vencimento: ");
   func.venc = lerFloat(665.0,1400.5);

  return func;
}
```




```
void leDadosFuncAlt(tipoFunc *pfunc){/*Função alternativa que devolve dados obtidos por referência */
   printf("Numero: ");
   pfunc->numero = lerInteiro(1,100); //  (*pFunc).numero=lerInteiro(1,100);
   printf("Nome: ");
   fgets(pfunc->nome,80,stdin);      // <=> fgets((*pFunc).nome,80)
   printf("Vencimento: ");
   pfunc->venc = lerFloat(665.0,1400.5); //(*pFunc).venc=lerFloat(665.0,1400.5)
}
```

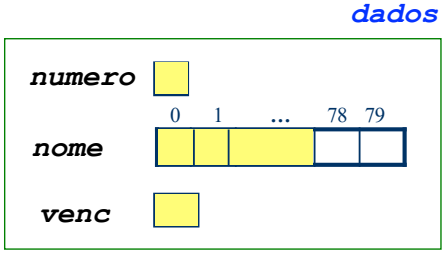

Exercício: resolução (VIII)

```
void escreveFunc(tipoFunc func){

   printf("\n Numero: %d\n", func.numero);

   printf(" Nome:%s\n", func.nome);

   printf(" Vencimento:%.2f\n",func.venc);
}
```



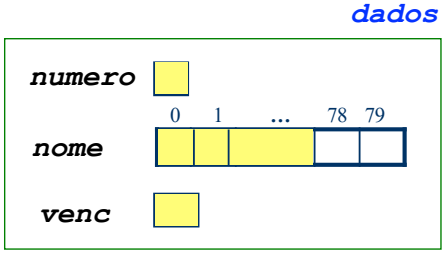
Exercício: resolução (VIIIa)

```c
/* Exemplo de Programa que obtem dados de um funcionario e lista
   no monitor (utilizando as funções anteriormente desenvolvidas) */
#include <stdio.h>
#include <string.h>
typedef struct {
   int numero;
   char nome[80];
   float venc;
}tipoFunc;
tipoFunc leDadosFunc(void);
void escreveFunc(tipoFunc func);
void limpaBufferStdin(void);
int lerInteiro(int min, int max);
float lerFloat(float min, float max);
void lerString(char vetor[], int max);
int main(void){
  tipoFunc dados;
   dados = leDadosFunc();
   escreveFunc(dados);
 return 0;
}
```



Exercício: resolução (VIIIb)

```c
/* Exemplo de Programa que obtem dados de um funcionario e lista
   no monitor (utilizando as funções anteriormente desenvolvidas) */
#include <stdio.h>
#include <string.h>
typedef struct {
   int numero;
   char nome[80];
   float venc;
}tipoFunc;
void escreveFunc (tipoFunc func);
void leDadosFuncAlt (tipoFunc *func);
void limpaBufferStdin(void);
int lerInteiro(int min, int max);
float lerFloat(float min, float max);
void lerString(char vetor[], int max);
int main(void){
  tipoFunc dados;
   leDadosFuncAlt(&dados);
   escreveFunc(dados);
 return 0;
}
```

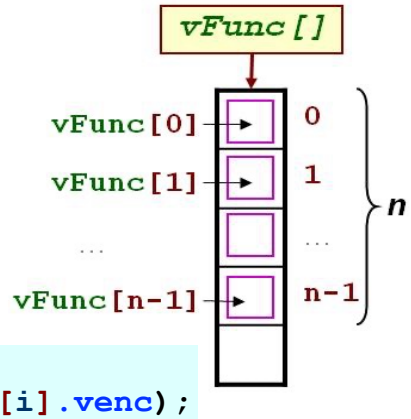


Exercício: resolução (IXa)

b1) Listar dados de todos os funcionários

A função a elaborar terá de receber vetor (**vFunc**) com dados dos funcionários, e a quantidade de funcionários (**n**) armazenados.

```
void listagemFunc_v0(tipoFunc vFunc[MAXFUNC], int n){
 int i;

 if (n==0) {
   printf("\n Não existem dados!!");
 }
 else {
   printf("\n Numero \t Nome \t Vencimento");

   for (i=0; i<n; i++) {
    printf("\n %d \t %s \t %.2f ",
          vFunc[i].numero, vFunc[i].nome, vFunc[i].venc);
   }
 }
}
```



Exercício: resolução (IXb)

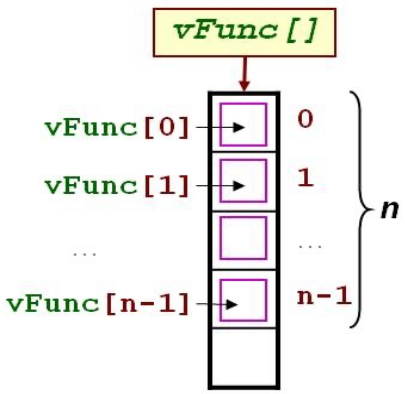
b1) Listar dados de todos os funcionários

Utilizando-se a função

void escreveFunc(tipoFunc func);

A função a elaborar terá de receber vetor (**vFunc**) com dados dos funcionários, e a quantidade de funcionários (**n**) armazenados.

```
void listagemFunc(tipoFunc vFunc[MAXFUNC], int n){
   int i;

   if (n==0) {
      printf("\n Não existem dados!!");
   }
   else {

    for (i=0; i<n; i++) {
      escreveFunc(vFunc[i]);
    }
   }
}
```



Exercício: resolução (X)

```
// main() Opcao 1 OK

#include <stdio.h>
#include <string.h>

#define MAXFUNC 50
typedef struct {
  int numero;
  char nome[80];
  float venc;
}tipoFunc;

void listagemFunc(tipoFunc vFunc[MAXFUNC], int n);
void escreveFunc(tipoFunc func);
tipoFunc leDadosFunc(void);

void limpaBufferStdin(void);
int lerInteiro(int min, int max);
float lerFloat(float min, float max);
void lerString(char vetor[], int max);
int menu();
```


Exercício: resolução (XI)

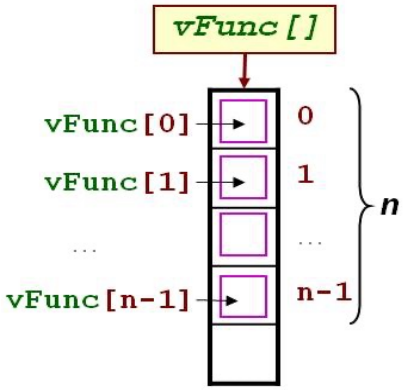
```
int main(void){
 tipoFunc vetorFunc[MAXFUNC];   /* Declara vetor estático */
 int numeroFunc, op;
 numeroFunc = 0;          /* inicia número de funcionários */
 do {
   op = menu();
   switch (op) {
    case 1: listagemFunc(vetorFunc, numeroFunc);
            break;
    case 2:    // TO DO - Calcular média de vencimentos
            break;
    case 3:    // TO DO - Acrescentar dados de um funcionário
            break;
    case 4:    // TO DO - Remover dados de um funcionário
            break;
   }
 }while(op!=0);
 return 0;
}
```

Exercício: resolução (XII)

b2) Calcular média de vencimentos dos funcionários

A função a elaborar terá de receber vetor (**vFunc**) com dados dos funcionários, e a quantidade de funcionários (**n**) armazenados.

```
float calcMediaVenc(tipoFunc vFunc[MAXFUNC], int n){
   float soma, media;
   int i;
   soma = 0.0;
   if (n==0) {
      printf("Não existem dados!!");
      media = 0.0;
   }
   else {
     for (i=0; i<n; i++) {
        soma += vFunc[i].venc;
     }
     media = soma/n;
   }
   return media;
}
```



Exercício: resolução (XIII)

```
// main() Opcao 1 e Opcao 2 OK
#include <stdio.h>
#include <string.h>

#define MAXFUNC 50
typedef struct {
  int numero;
  char nome[80];
  float venc;
}tipoFunc;

float calcMediaVenc(tipoFunc vFunc[MAXFUNC], int n);

void listagemFunc(tipoFunc vFunc[MAXFUNC], int n);
void escreveFunc(tipoFunc func);
tipoFunc leDadosFunc(void);
void limpaBufferStdin(void);
int lerInteiro(int min, int max);
float lerFloat(float min, float max);
void lerString(char vetor[], int max);
int menu();
```

Exercício: resolução (XIV)

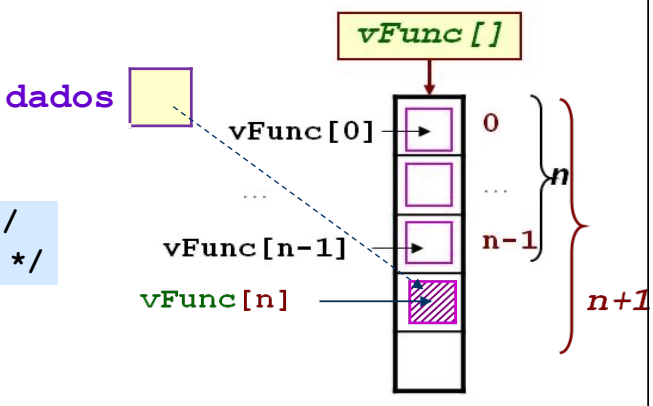
```
int main(void){
  tipoFunc vetorFunc[MAXFUNC];   /* Declara vetor estático */
  int numeroFunc, op; float mediaVenc;
  numeroFunc = 0;          /* inicia número de funcionários */
  do {
    op = menu();
    switch (op) {
      case 1: listagemFunc(vetorFunc, numeroFunc);
              break;
      case 2: mediaVenc = calcMediaVenc(vetorFunc, numeroFunc);
              printf ("\n Media vencimentos = %.2f", mediaVenc);
              break;
      case 3:     // TO DO - Acrescentar dados de um funcionário
              break;
      case 4:     // TO DO - Remover dados de um funcionário
              break;
    }
  }while(op!=0);
  return 0;
}
```


Exercício: resolução (XVa)

b4_v0) Acrescenta dados de um (novo) funcionário

A função a elaborar terá de receber vetor (**vFunc**) com dados dos funcionários e o número de funcionários (**n**) – que poderá ser atualizado (e devolvido por return). Não verifica a existência de elementos repetidos.

```
int acrescentaNoVetor_V0(tipoFunc vFunc[MAXFUNC], int n){
  tipoFunc dados; int pos;

  if (n == MAXFUNC){
    printf("Impossível acrescentar");
  }
  else {
    dados=leDadosFunc();
    vFunc[n]=dados; /* Insere no vetor */
    n++;      /* Atualiza n° funcionarios */
  }
  return n;
}
```

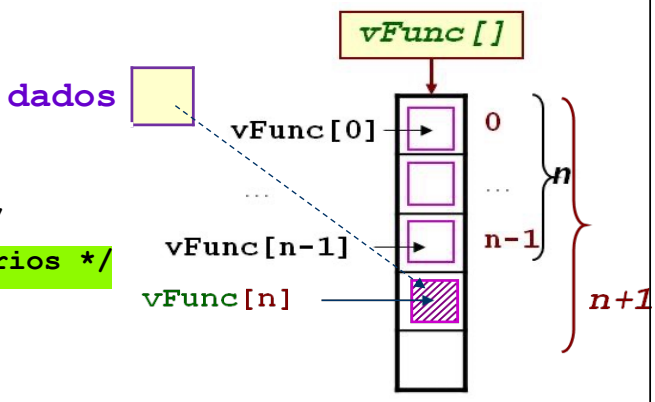


Exercício: resolução (XVb)

b4_v0alt) Acrescenta dados de um (novo) funcionário

A função a elaborar terá de receber vetor (**vFunc**) com dados dos funcionários e o número de funcionários (**n**) – que poderá ser atualizado (por referência).
Não verifica a existência de elementos repetidos.

```
void acrescentaNoVetor_V0alt(tipoFunc vFunc[MAXFUNC], int *n){
  tipoFunc dados; int pos;

 if (*n == MAXFUNC){
   printf("Impossível acrescentar");
 }
 else {
   dados=leDadosFunc();
   vFunc[*n]=dados; /* Insere no vetor */
   (*n)++;          /* Atualiza nº funcionarios */
 }
}
```



Exercício: resolução (XVc)

```
#include <stdio.h>
#include <string.h>
#define MAXFUNC 50
typedef struct {
  int numero;
  char nome[80];
  float venc;
}tipoFunc;

int acrescentaNoVetor_V0(tipoFunc vFunc[MAXFUNC], int n);
void acrescentaNoVetor_V0alt(tipoFunc vFunc[MAXFUNC], int n);

float calcMediaVenc(tipoFunc vFunc[MAXFUNC], int n);
void listagemFunc(tipoFunc vFunc[MAXFUNC], int n);
void escreveFunc(tipoFunc func);
tipoFunc leDadosFunc(void);
void limpaBufferStdin(void);
int lerInteiro(int min, int max);
float lerFloat(float min, float max);
void lerString(char vetor[], int max);
int menu();
```

Exercício: resolução (XVd)

```
int main(void){
  tipoFunc vetorFunc[MAXFUNC];   /* Declara vetor estático */
  int numeroFunc, op; float mediaVenc;
  numeroFunc = 0;          /* inicia número de funcionários */
  do {
    op = menu();
    switch (op) {
      case 1: listagemFunc(vetorFunc, numeroFunc);
              break;
      case 2: mediaVenc = calcMediaVenc(vetorFunc, numeroFunc);
              printf ("\n Media vencimentos = %.2f", mediaVenc);
              break;
      case 3: numeroFunc=acrescentaNoVetor_V0(vetorFunc,numeroFunc);
              /* ou   acrescentaNoVetor_V0alt(vetorFunc, &numeroFunc); */
              break;
      case 4:      /* TO DO - Remover dados de um funcionário */
              break;
    } // switch
  }while(op!=0);
  return 0;
}
```


Exercício: resolução (XVI)

b3) Procurar dados de um funcionário

A função a elaborar terá de receber vetor (**vFunc**) com dados dos funcionários, a quantidade de funcionários (**n**) armazenados, e o número do funcionário a procurar.

Caso o encontre devolve a respetiva posição no vetor, senão devolve o valor -1.

```
int procuraFunc(tipoFunc vFunc[], int n, int numFuncionario){
  int i, pos;            /* posicao dos dados do func a procurar */
  pos = -1;

  for (i=0; i<n; i++){
    if (vFunc[i].numero == numFuncionario){/* Elemento encontrado */
      pos = i;               /* guarda posição de numero em vFunc */
      i = n;              /* para concluir pesquisa (sair do for) */
    }
  }
  return pos;
}
```

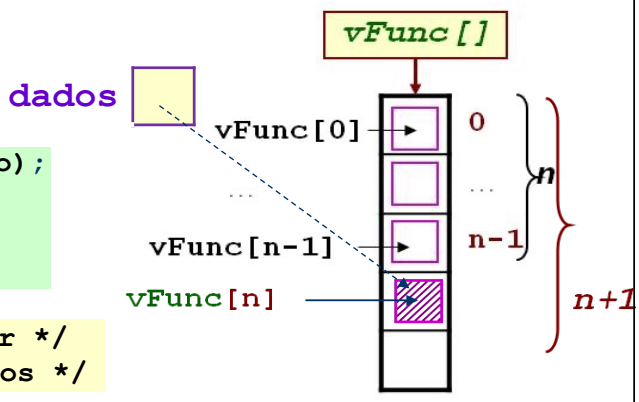
Exercício: resolução (XVII)

b4) Acrescenta dados de um (novo) funcionário

A função a elaborar terá de receber vetor (**vFunc**) com dados dos funcionários e o número de funcionários (**n**) – que poderá ser atualizado (e é devolvido pela função).

Antes de inserir os dados no vetor terá de verificar (pelo número) se o funcionário já se encontra na estrutura, de forma a não inserir elementos repetidos (com mesmo número).

```
int acrescentaNoVetor(tipoFunc vFunc[MAXFUNC], int n){
   tipoFunc dados; int pos;
  if (n == MAXFUNC){
    printf("Impossível acrescentar");
  }
  else {
    dados=leDadosFunc();
    pos=procuraFunc(vFunc,n,dados.numero);
    if (pos != -1){
      printf("Func já existente");
    }
    else {
      vFunc[n]=dados; /* Insere no vetor */
      n++;     /* Atualiza n° funcionarios */
    }
  }
  return n;
}
```



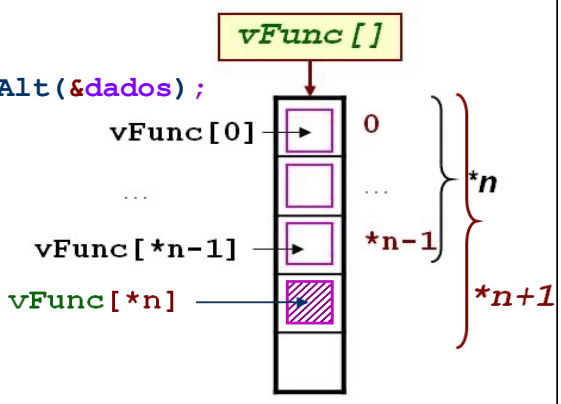
Exercício: resolução (XVIIb)

b4alt) Acrescenta dados de um (novo) funcionário

A função a elaborar terá de receber vetor (**vFunc**) com dados dos funcionários e o número de funcionários (***n**) – que poderá ser atualizado.

Antes de inserir os dados no vetor terá de verificar (pelo número) se o funcionário já se encontra na estrutura, de forma a não inserir elementos repetidos (com mesmo número).

```
void acrescentaNoVetorAlt(tipoFunc vFunc[MAXFUNC], int *n){
   tipoFunc dados; int pos;
  if (*n == MAXFUNC){
    printf("Impossível acrescentar");
  }
  else {
    dados=leDadosFunc();      //ou leDadosFuncAlt(&dados);
    pos=procuraFunc(vFunc,*n,dados.numero);
    if (pos != -1){
      printf("Func já existente");
    }
    else {
      vFunc[*n]= dados; /* Insere no vetor */
      (*n)++;    /* Atualiza n° funcionarios */
    }
  }
}
```

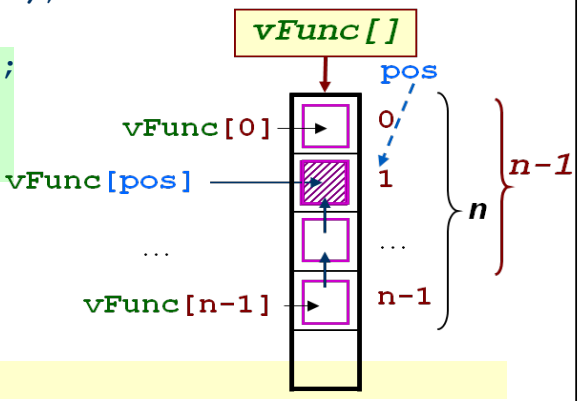


Exercício: resolução (XVIII)

b5) Elimina dados de um funcionário (existente no vetor)

A **função eliminaDoVetor** terá de receber o vetor (**vFunc**) com dados e a quantidade de funcionários (**n**) – que poderá ser atualizado.
Antes de eliminar do vetor os dados de um funcionário, ter-se-á de verificar se existem dados armazenados no vetor, e se o funcionário a eliminar se encontra na estrutura.

```
int eliminaDoVetor(tipoFunc vFunc[MAXFUNC], int n){
  int i, numeroFunc, pos;       /* posicao dos dados do func a eliminar   */

 if (n == 0){
   printf("Não há funcionários");
 }
 else {
   printf("Numero do funcionário (a eliminar): ");
   numeroFunc = lerInteiro(1,100);
   pos = procuraFunc(vFunc, n, numeroFunc);
   if (pos == -1) {
      printf ("Funcionario não existe!");
   }
   else {
         for (i=pos; i < n-1; i++){
             vFunc[i] = vFunc[i+1];
          }
        n--; /* Atualiza nº funcionarios */
    }
  }
  return n;
}
```



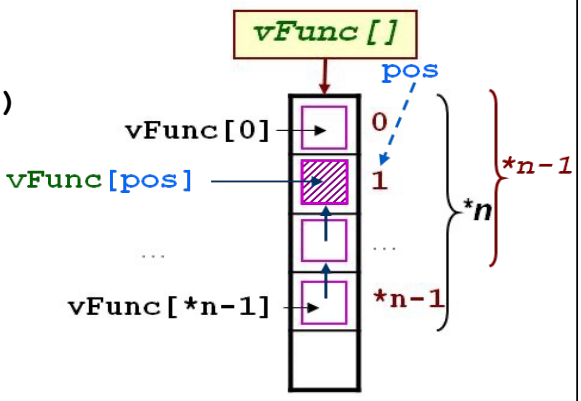
Exercício: resolução (XVIIIb)

b5alt) Elimina dados de um funcionário (existente no vetor)

A **função eliminaDoVetor** terá de receber o vetor (**vFunc**) com dados e a quantidade de funcionários (***n**) – que poderá ser atualizado.

Antes de eliminar do vetor os dados de um funcionário, ter-se-á de verificar se existem dados armazenados no vetor, e se o funcionário a eliminar se encontra na estrutura.

```
void eliminaDoVetorAlt(tipoFunc vFunc[MAXFUNC], int *n){
  int i, numeroFunc, pos;       /* posicao dos dados do func a eliminar   */

 if (*n == 0){
   printf("Não há funcionários");
 }
 else {
   printf("Numero do funcionario: ");
   numeroFunc = lerInteiro(1,100);
   pos = procuraFunc(vFunc, *n, numeroFunc)
   if (pos == -1) {
      printf ("Funcionario não existe!");
   }
   else {
         for (i=pos; i < *n-1; i++){
             vFunc[i] = vFunc[i+1];
          }
        (*n)--; /* Atualiza nº funcionarios */
    }
 }
}
```



Exercício: resolução (XIX)

```
#include <stdio.h>
#include <string.h>
#define MAXFUNC 50
typedef struct {
  int numero;
  char nome[80];
  float venc;
}tipoFunc;
int procuraFunc(tipoFunc vFunc[], int n, int numFuncionario);
int acrescentaNoVetor(tipoFunc vFunc[MAXFUNC], int n);
int eliminaDoVetor(tipoFunc vFunc[MAXFUNC], int n);
float calcMediaVenc(tipoFunc vFunc[MAXFUNC], int n);
void listagemFunc(tipoFunc vFunc[MAXFUNC], int n);
void escreveFunc(tipoFunc func);
tipoFunc leDadosFunc(void);
void limpaBufferStdin(void);
int lerInteiro(int min, int max);
float lerFloat(float min, float max);
void lerString(char vetor[], int max);
int menu();
```


Exercício: resolução (XX)

```
// Programa utilizando vetor de estruturas (cont)

int main(void){
 tipoFunc vetorFunc[MAXFUNC];   /* Declara vetor estático */
 float mediaVenc;
 int numeroFunc, op;
 numeroFunc = 0;          /* inicia número de funcionários */
 do {
  op = menu();
  switch (op) {
   case 1: listagemFunc(vetorFunc, numeroFunc);
           break;
   case 2: mediaVenc = calcMediaVenc(vetorFunc, numeroFunc);
           printf ("\n Media vencimentos = %.2f", mediaVenc);
           break;
```

Exercício: resolução (XXI)

```
/* Programa utilizando vetor de estruturas (cont) */

  case 3: numeroFunc=acrescentaNoVetor(vetorFunc,numeroFunc);
    /*ou   acrescentaNoVetorAlt(vetorFunc, &numeroFunc); */
        break;
  case 4: numeroFunc=eliminaDoVetor(vetorFunc,numeroFunc);
     /*ou eliminaDoVetorAlt(vetorFunc, &numeroFunc); */
        break;

    } // switch
   }while(op!=0);
  return 0;
 }
```


Exercício: resolução (XXII)

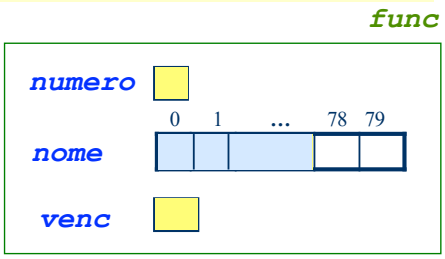
```
void lerString(char vetor[], int max){
   int tamanhoString;

   do{                  // Repete leitura caso sejam introduzidas strings apenas com o \n (tamanhoString == 1)
      fgets(vetor, max, stdin);
      tamanhoString = strlen(vetor);
   } while (tamanhoString == 1);
   if (vetor[tamanhoString-1] != '\n'){// não foram armazenados no vetor todos os caracteres
         limpaBufferStdin();           // elimina caracteres que ficaram no buffer stdin
   }
   else{
         vetor[tamanhoString-1] ='\0';/* substitui por \0 o caracter \n da string armazenada */
   }
}
/*
```

- Como a leitura de caracteres termina com o caracter '**\n**' (mudança de linha), o **fgets()** considera que o único caracter pertence à *string* lida.
- Se o '**\n**' não for encontrado no vetor (penúltimo caracter da *string*), é porque não foi armazenada toda a string introduzida, tendo ficado caracteres no stdin.
- Para eliminar o '**\n**' da *string*, esse caracter é substituído pelo '**\0**'.

```
*/
```

Exercício: resolução (XXIII)

```
tipoFunc leDadosFunc (void){
    tipoFunc func;
    printf("Numero: ");
    func.numero = lerInteiro(1,100);
    printf("Nome: ");
    lerString(func.nome,80);
    printf("Vencimento: ");
    func.venc = lerFloat(505,1200.5);
  return func;
}
```




```
void leDadosFuncAlt (tipoFunc *pfunc){
    printf("Numero: ");
    pfunc->numero = lerInteiro(1,100); //  (*pFunc).numero = lerInteiro(1,100);
    printf("Nome: ");
    lerString(pfunc->nome,80);        // <=> lerString((*pFunc).nome,80)
    printf("Vencimento: ");
    pfunc->venc = lerFloat(505,1200.5); //(*pFunc).venc = lerFloat(505,1200.5)
}
```