

Lab 3 – Estruturas

Ricardo e Myriam

Conceitos básicos:

1. Definição de uma estrutura (antes do main) tipo **struct vetor**:

```
struct vetor{  
    int n;  
    float elem[20];  
};
```
2. Declaração de uma variável **v** do tipo struct vetor:

```
struct vetor v;
```
3. Acesso aos membros de uma variável **v** do tipo struct vetor: **v.n** é uma variável do tipo int; **v.elem[i]** é uma variável do tipo float, elemento **i** de uma variável indexada
4. **typedef**, definição de uma variável do tipo **Vetor**:

```
typedef struct vetor Vetor;
```
5. Declaração de uma variável **v** do tipo Vetor (novo tipo):

```
Vetor v;
```

1. **Exemplo**: a função **ordena** com o seguinte protótipo:

```
void ordena(struct vetor v);
```

```
#include <stdio.h>  
#include <stdlib.h>  
  
struct vetor{  
    int n;  
    float elem[20];  
};  
  
void ordena(struct vetor x);  
  
int main(int argc, char *argv[])  
{  
    struct vetor v;  
    /* declaração de outras variáveis e leitura dos dados de entrada */  
    ordena(v);  
    /* imprime resultado */  
    system("PAUSE");  
    return 0;  
}  
  
void ordena(struct vetor x)  
{  
    for(j=0; j<x.n-1; j++){  
        for(i=0; i<x.n-1; i++){  
            if (x.elem[i] > x.elem[i+1]){  
                aux=x.elem[i];  
                x.elem[i]= x.elem[i+1];  
                x.elem[i+1]=aux;  
            }  
        }  
    }  
}
```

2. Implemente em C uma função de nome **pEscalar** que obtém o produto escalar de dois vetores. Os vetores são passados à função nos dois primeiros argumentos (**Vetor v1, Vetor v2**). A função retorna (VALOR DE RETORNO) o produto escalar calculado, caso a dimensão dos vetores seja a mesma. Caso contrário, a função deverá retornar o valor **-1**, indicando **erro**, ou seja, não é possível calcular o produto escalar de dois vetores de diferentes dimensões.

Use o seguinte protótipo para a função:

float pEscalar(Vetor v1, Vetor v2);

O tipo Vetor é um novo tipo definido por typedef:

```
typedef struct vetor{
    int n;
    float elem[20];
} Vetor;
```

3. Implemente em C uma função de nome **distancia** que calcula a distância euclidiana entre dois pontos do espaço tridimensional. Os pontos são passados à função nos dois primeiros argumentos (**Ponto p1, Ponto p2**). A função retorna (VALOR DE RETORNO) a distância calculada.

Use o seguinte protótipo para a função:

float distancia(Ponto p1, Ponto p2);

O tipo Ponto é um novo tipo definido por typedef:

```
typedef struct ponto{
    float x, y, z;
} Ponto;
```