Programação de Computadores I

DÉCIMA QUARTA
PARTE

CAPÍTULO V

SUBPROGRAMAS
& PONTEIROS

PARTE III – PONTEIROS
PARA ESTRUTURAS

Programação de Computadores I

prof. Marco Villaça

Passando uma estrutura a uma função sem modificá-la

```
struct revenda {char modelo[20];
                char marca[20];
                int ano;
                float valor;};

void imprime(struct revenda);
int main()
{
       ...
       struct revenda carros[50];
       ...
       imprime(carros[k]);
       ...
}
```

```
void imprime(struct revenda autos)
{
    printf("%-12s %-12s %d %9.2f\n",
 autos.modelo, autos.marca, autos.ano,
 autos.valor);
}
```

Passando uma estrutura a uma função para modificá-la

```
struct revenda {char modelo[20];
                char marca[20];
                int ano;
                float valor;};

void apaga(struct revenda *, int );
int main()
{
        ...
        struct revenda carros[50];
        ...
        apaga(carros, k);
        ...
}
```

```
void apaga(struct revenda *ptr, int i)
{
        (*(ptr + i)).modelo[0]='\0';
}
```

Passando uma estrutura a uma função usando o formato ponteiro->membro

```
struct revenda {char modelo[20];
                char marca[20];
                int ano;
                float valor;};

void apaga(struct revenda *, int );
int main()
{
        ...
        struct revenda carros[50];
        ...
        apaga(carros, k);
        ...
}
```

```
void apaga(struct revenda *ptr, int i)
{
        (ptr + i)->modelo[0]='\0';
}
```

EXEMPLO 1

Calcular em uma função a distância entre 2 pontos, considerando os pontos declarados como uma estrutura no programa principal:

```
struct ponto{ int x;
              int y; };
int main()
{
     struct ponto a,b;
     ...
```

O programa principal deve imprimir o resultado.

EXEMPLO 2

Criar uma função para trocar apenas as abcissas (posição x) de 2 pontos, considerando os pontos declarados como uma estrutura no programa principal:

```
struct ponto{ int x;
              int y; };
int main()
{
     struct ponto a,b;
     ...
```

O programa principal deve imprimir o resultado.

EXEMPLO 3

Criar uma função para trocar a posição de 2 pontos no plano cartesiano considerando os pontos pertencentes a um vetor declarado como uma estrutura no programa principal

```
struct ponto{ int x;
              int y; };
int main()
{
     struct ponto pontos[2];
     ...
```

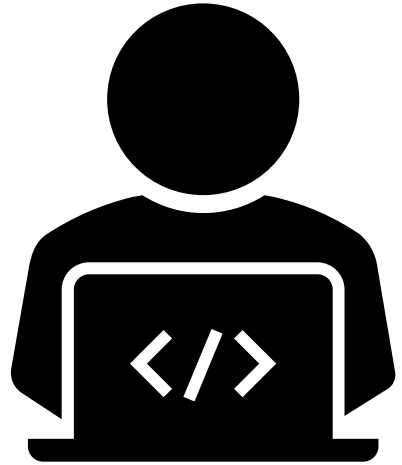
O programa principal deve imprimir o resultado.

Exercício 1

Criar uma função para trocar a posição das abcissas de 2 pontos no plano cartesiano considerando os pontos pertencentes a um vetor declarado como uma estrutura no programa principal

```
struct ponto{ int x;
              int y; };
int main()
{
   struct ponto pontos[2];
   ...
```

O programa principal deve imprimir o resultado.



Exercício 2

Criar uma estrutura para armazenar informação sobre alunos de uma academia de ginástica:

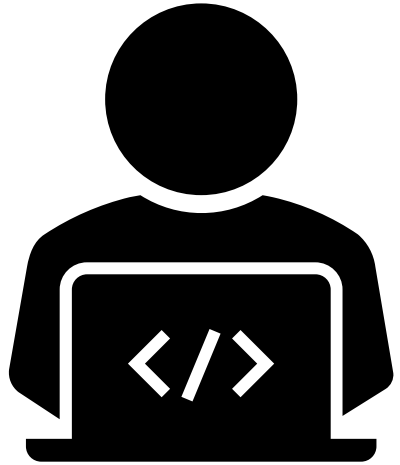
Nome do aluno, sexo, idade, peso, altura, **imc**.

Você usara um vetor (matriz) de estruturas, em que o número de elementos é o número alunos

Entrar com os dados pelo teclado

O programa deve imprimir os alunos com imc > 30, informando ao lado do aluno “Obesidade”, e com imc < 18,5, informando ao lado do nome do aluno “Baixo peso”.

O imc de cada aluno deve ser calculado em uma função.



Bibliografia e crédito das figuras

OUALLINE, S. Practical C Programming. 3a ed. O'Reilly, 1997.

SEBESTA, R. Conceitos de Linguagens de Programação. 5a ed. Porto Alegre: Bookman, 2003.