

Linguagem de Programação

Tipos Estruturados II - *Structs*

ECT2303

helton.maia@ect.ufrn.br

Passando estruturas como parâmetros de uma função

```
5 //sensor de rodovias|
6 struct Sensor{
7     int code;
8     float velocidade;
9     char placa[MAX];
10 };
11
```

```
12 void cadastroSensor(Sensor &s){
13     cout << "Codigo: ";    cin >> s.code;
14     cout << "Velocidade: "; cin >> s.velocidade;
15     cin.ignore();
16     cout << "Placa: ";    cin.getline(s.placa, MAX);
17 }
```


```
int main(){
    Sensor sensor;

    cadastroSensor(sensor);
}
```

Passagem por referência



Passando estruturas como parâmetros de uma função

```
//sensor de rodovias  
 struct Sensor{  
    int code;  
    float velocidade;  
    char placa[MAX];  
};
```

Escreva uma função para imprimir cadastro realizado utilizando *structs*.

Passando estruturas como parâmetros de uma função

Escreva uma função para imprimir o cadastro realizado utilizando *structs*.

```
void cadastroSensor(Sensor &s){  
  
void printSensor(Sensor s){  
    cout << endl << "-----"  
    cout << "Codigo: " << s.code << endl;  
    cout << "Velocidade: " << s.velocidade << endl;  
    cout << "Placa: " << s.placa << endl;  
}  
  
int main(){  
    Sensor sensor;  
  
    cadastroSensor(sensor);  
}
```

Passando estruturas como parâmetros de uma função

```
//sensor de rodovias
struct Sensor{
    int code;
    float velocidade;
    char placa[MAX];
};
```

Escreva uma função do tipo *bool* que testa se a velocidade registrada pelo sensor ultrapassou um certo limite, por exemplo 80km.

Passando estruturas como parâmetros de uma função

```
struct Sensor{  
    //verifica se a velocidade ultrapassa o limite  
    bool verificaVelocidade(Sensor s){  
        if (s.velocidade > 80){  
            return true;  
        } else return false;  
    }  
}  
  
void cadastroSensor(Sensor &s){  
  
void printSensor(Sensor s){  
  
int main(){  
    Sensor sensor;  
  
    cadastroSensor(sensor);  
    printSensor(sensor);  
    cout << "Ultrapassou 80km: " << verificaVelocidade(sensor)
```

Passando arrays de estruturas como parâmetros de uma função

- Esta passagem é feita automaticamente por referência

Passando arrays de estruturas como parâmetros de uma função

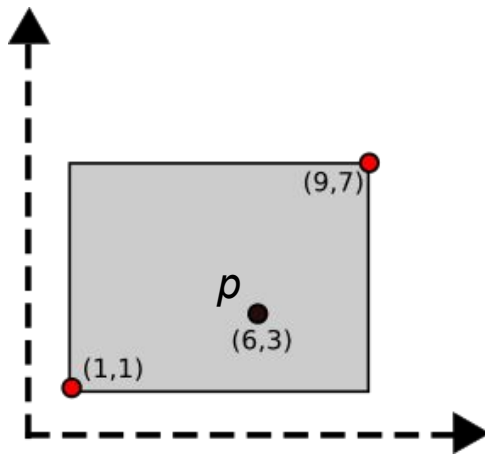
```
void cadastroSensor(Sensor s[], int n){  
    for (int i=0; i< n; i++){  
        cout << "Codigo: ";        cin >> s[i].code;  
        cout << "Velocidade: ";    cin >> s[i].velocidade;  
        cin.ignore();  
        cout << "Placa: ";        cin.getline(s[i].placa, MAX);  
    }  
}  
  
int main(){  
    int n = 2;  
    Sensor sensor[n];  
  
    cadastroSensor(sensor, n);  
}
```


Exercício 1

Defina um tipo estruturado *Aluno*, contendo os campos: nome, matrícula e média. Em seguida, implemente uma **função** para receber como parâmetro, um vetor com n alunos e suas respectivas informações. Calcule e exiba os alunos que possuem a maior e menor nota, considerando que podem ocorrer notas repetidas nestas categorias maior/menor.

Exercício 2

Sejam p um ponto e r um retângulo. Defina uma **função** que retorna verdadeiro se p está dentro do retângulo ou falso caso contrário. Esta função vai receber como argumentos de entrada, dois novos tipos, a *struct* do ponto p com as coordenadas cartesianas (x,y) e a *struct* do retângulo r com as informações de seus vértices, esquerdo inferior e direito superior.



Perguntas ?