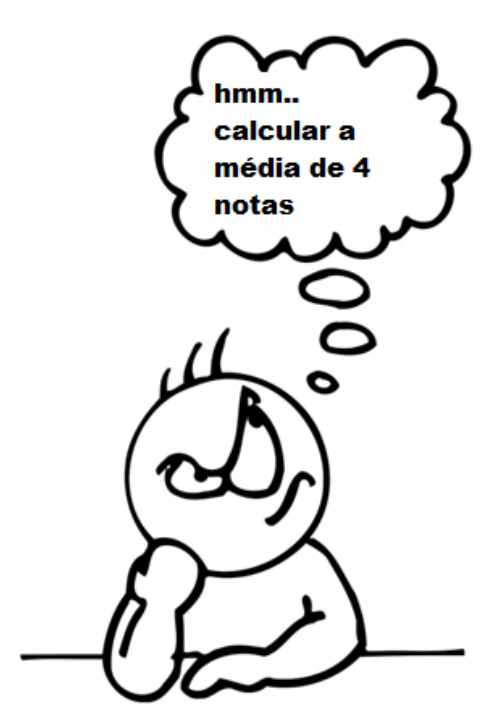
BCC 201 - Introdução à Programação I

# Estruturas (registros)

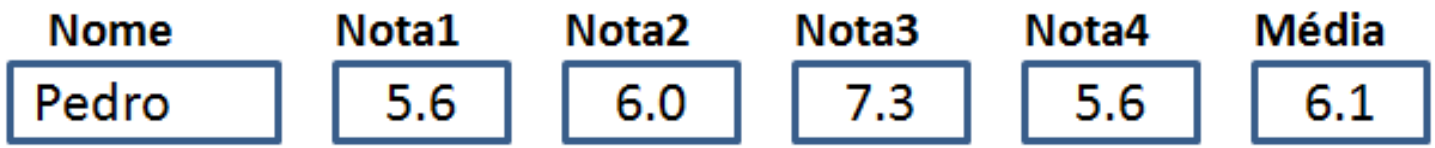
Guillermo Cámara-Chávez
UFOP

# Estruturas (*struct*) I

- Imaginemos que queremos ler as notas de 4 provas para um aluno e
- calcular a média do aluno



# Estruturas (*struct*) II



# Estruturas (*struct*) III

# Estruturas (*struct*) IV

**Aluno**

| Nome | Nota1 | Nota2 | Nota3 | Nota4 | Média |
|---|---|---|---|---|---|
| Pedro | 5.6 | 6.0 | 7.3 | 5.6 | 6.1 |

# Estruturas (*struct*) V

- "*structs*" são Estruturas de Dados Heterogêneas
- Uma estrutura **agrupa várias** variáveis **numa só**.
- Funciona como uma espécie de "ficha".
- A ficha armazena diversos **dados relacionados**, porém de **tipos diferente**.

# Estruturas (*struct*) VI

- A estrutura, então, serve para **agrupar** um **conjunto de dados não similares**, **formando um novo tipo** de dados.
- As estruturas podem conter elementos de qualquer tipo de dados tais como `int`, `char`, `float`, `double`, ponteiros, vetores, matrizes, *strings* ou mesmo outras estruturas.

# Estruturas (*struct*) VII

- Declaração

```
struct nome_da_estrutura
{
    tipo_campo1 nome_campo1;
    tipo_campo2 nome_campo2;
    ...
};
```

onde :

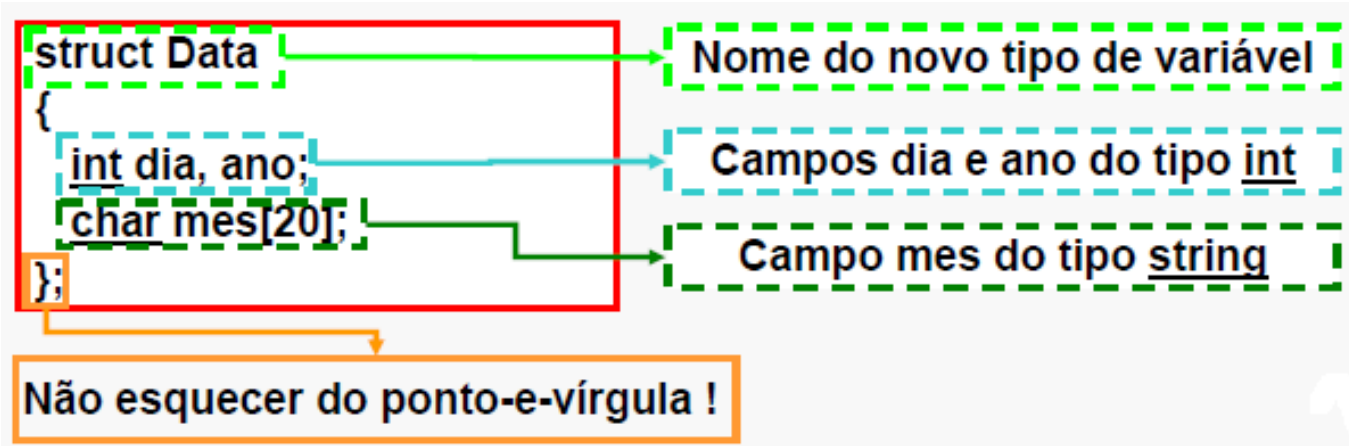
- `nome_da_estrutura` é o nome do tipo da estrutura e, entre chaves,
- `tipo_campo1`, `tipo_campo2`, ... é a lista com os tipos de dados em C (`char`, `int`, `float`, `double`, `char[]`)

# Estruturas (*struct*) VIII

- Exemplo 1:
  - Definir um novo tipo de variável Data (`struct Data`)
  - A partir daquele momento o compilador passa a conhecer um outro tipo de dado, chamado `struct Data`
  - Dita estrutura está composta por dois inteiros e um vetor de caracteres (inteiros: dia e ano, string: mes)

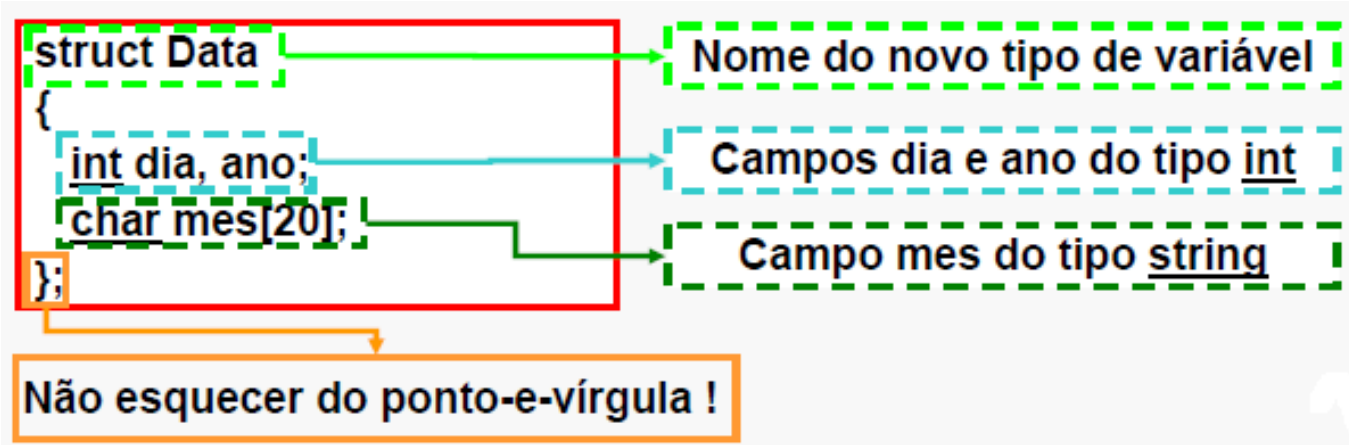
# Estruturas (*struct*) IX

- Definir um novo tipo de variável Data (`struct Data`)
- A partir daquele momento o compilador passa a conhecer um outro tipo de dado, chamado `struct Data`
- Dita estrutura está composta por dois inteiros e um vetor de caracteres (inteiros: dia e ano, string: mes)



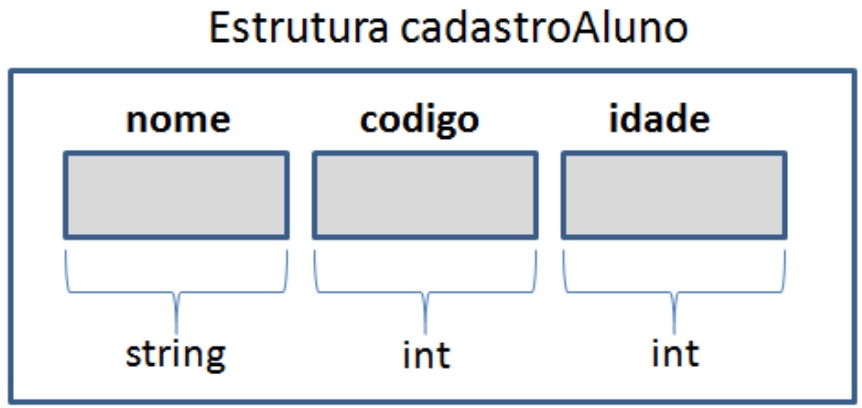
# Estruturas (*struct*) X

- Data não é uma variável, senão o nome pelo que é conhecido um **novo tipo de dados**
- Cada um dos elementos de `Data` é denominado campo.



# Estruturas (*struct*) XI

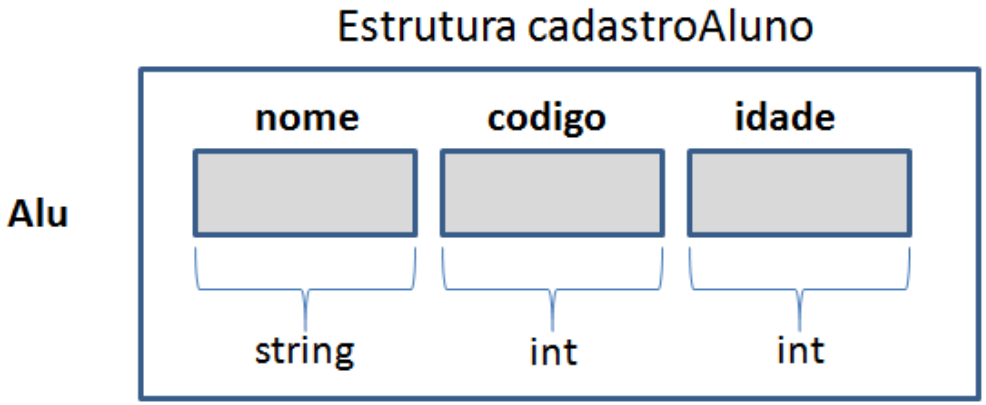
- Exemplo 2




Para atribuir valores a seus campos fazemos diretamente inserindo um "." (ponto) entre o nome da variável e o campo que nos interessa.

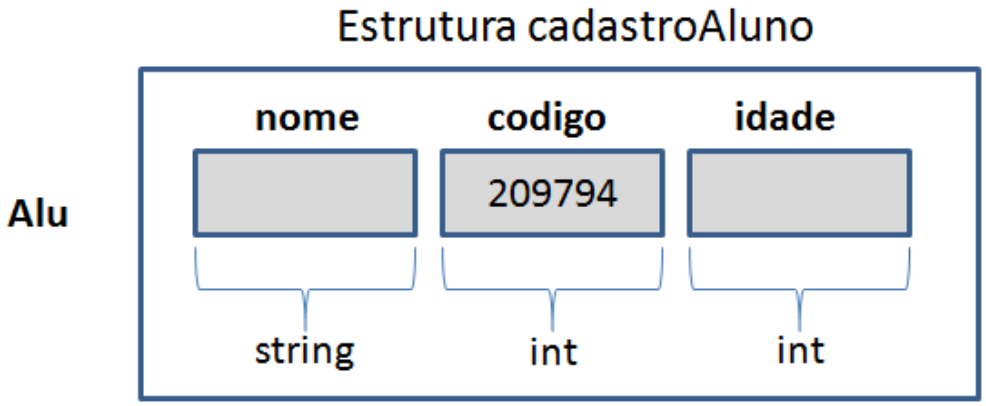
# Estruturas (*struct*) XII

**struct cadastroAluno** Alu;



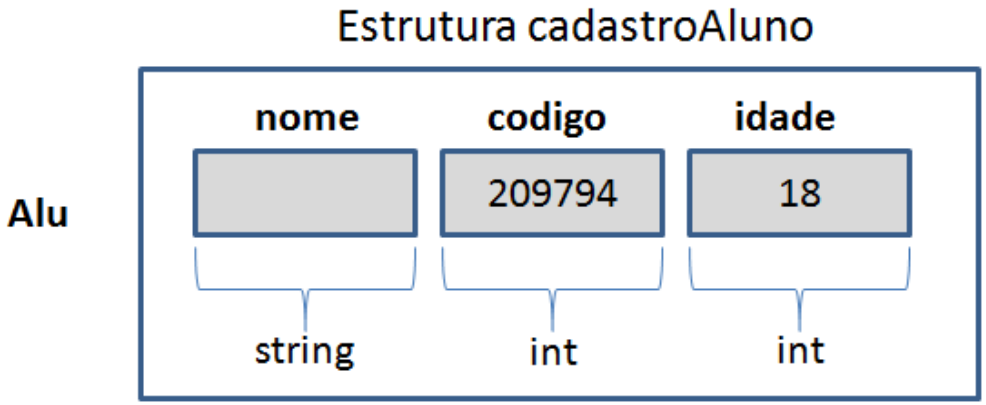
# Estruturas (*struct*) XIII

Alu.codigo = 209794;



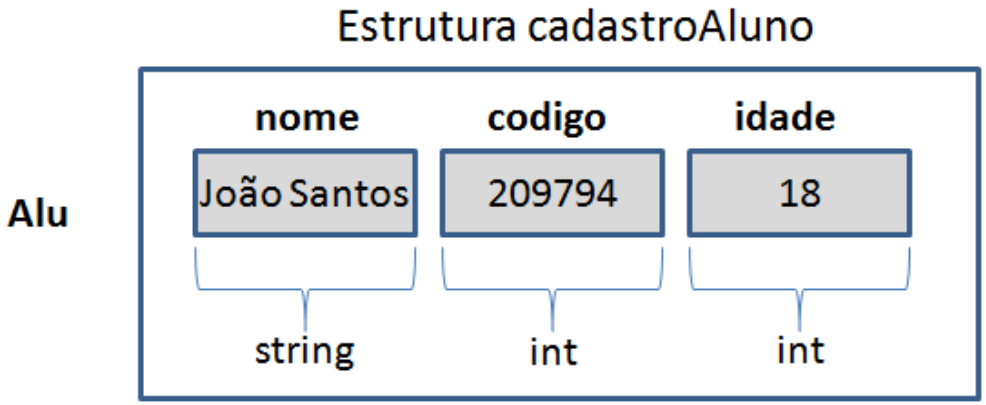
# Estruturas (*struct*) XIV

Alu.idade = 18;



# Estruturas (*struct*) XV

strcpy(Alu.nome, “Joao Santos”);



# Estruturas (*struct*) XVI

```
struct cadastroAluno
{
    string nome;
    int codigo;
    int idade;
};
```

# Estruturas (*struct*) XVII

```
int main()
{
    struct cadastroAluno Alu1, Alu2;
    Alu1.nome = "João Santos";
    Alu1.codigo = 365833;
    Alu1.idade = 19;

    cout << "Inserir Nome: ";
    getline(cin, Alu2.nome);
    cout << "Inserir codigo e idade: ";
    cin >> Alu2.codigo >> Alu2.idade;

    cout << Alu1.nome << Alu1.idade << Alu1.codigo
         << Alu2.nome << Alu2.idade << Alu1.codigo;
    return 0;
}
```

# Vetor de Registros I

- Declaração:

```
struct nome_registro nome_variavel[tamanho_vetor];
```

- Uso:

```
nome_variavel[indice].nome_do_campo;
```

# Vetor de Registros II

Criar o programa que permita cadastrar 20 alunos.

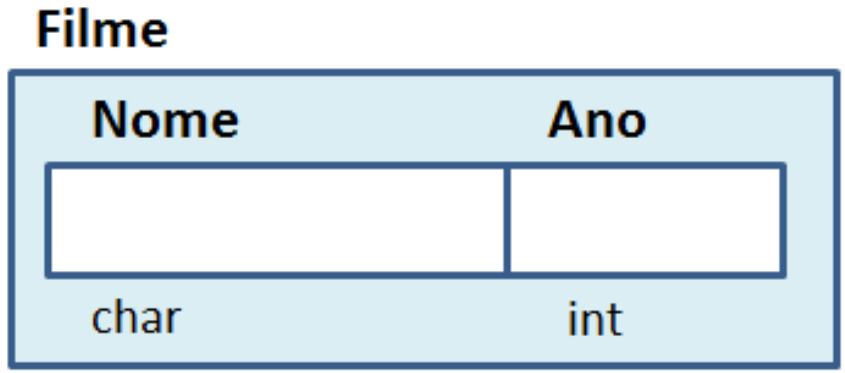
# Vetor de Registros III

```
struct cadastroAluno{
    string nome;
    int codigo;
    int idade;
};
int main(){
    struct cadastroAluno A[20];
    int i;
    for (i = 0; i < 20; i++){
        cout << "Inserir dados aluno " << i+1;
        cout << "Digite nome:";
        getline(cin, A[i].nome);
        cout << "Digite codigo e idade:";
        cin >> A[i].codigo >> A[i].idade;
    }
    for (i = 0; i < 20; i++)
        cout << A[i].nome << A[i].codigo
             << A[i].idade << endl;
    return 0;
}
```

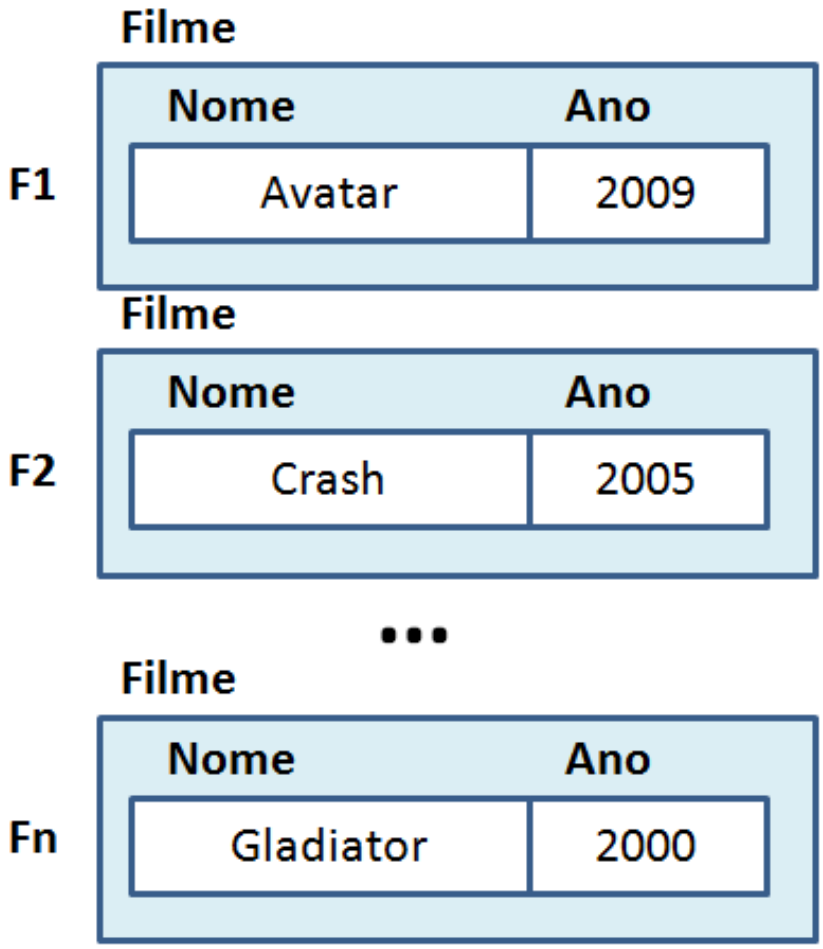
# Exemplos I

Inserir uma lista de $n$ filmes. A estrutura para os filmes deve conter os seguintes campos: nome e o ano de lançamento.

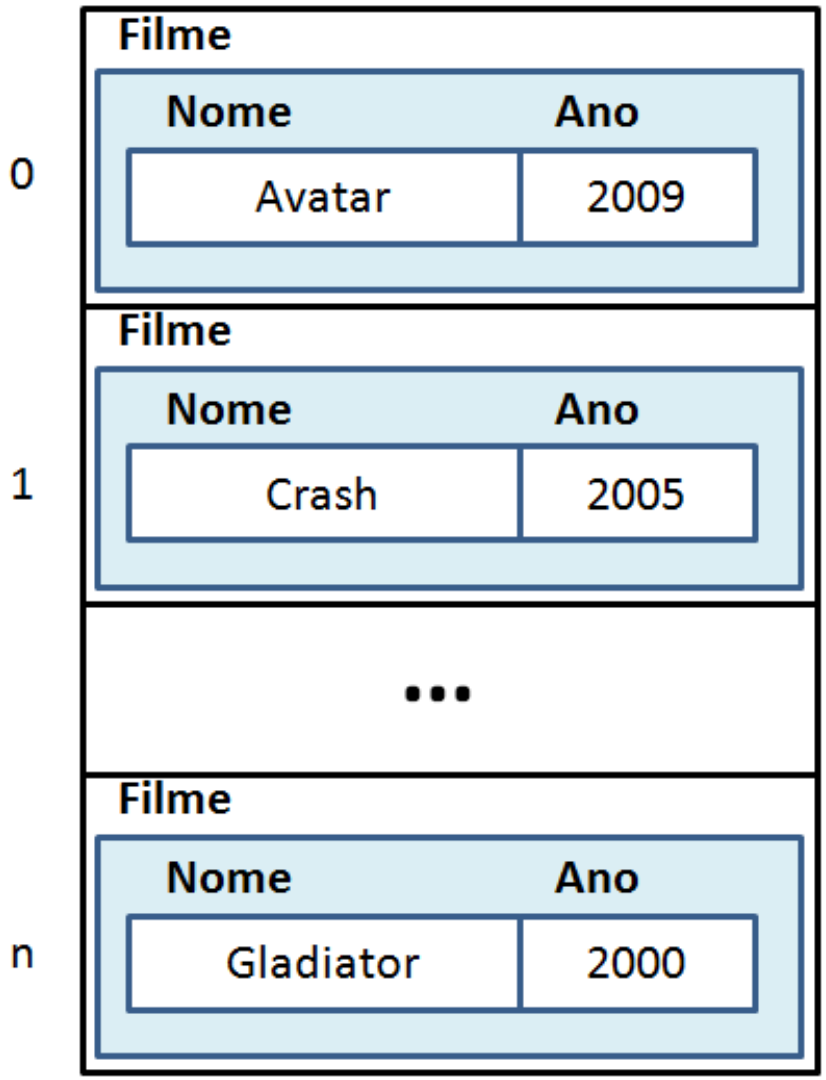
# Exemplos II



# Exemplos III



# Exemplos IV


0
Filme
Nome
Ano
Avatar
2009
1
Filme
Nome
Ano
Crash
2005
...
n
Filme
Nome
Ano
Gladiator
2000

# Exemplos V

```
struct filme{
    string nome;
    int ano;
};
int main(){
    int n;
    struct filme f[100];
    do{
        cout << "Inserir número de filmes ";
        cin >> n;
    }while (n > 100 || n < 0);
    //Insere n nomes de filmes
    for (int i = 0; i < n; i++)
    {
        cout << "Nome e ano: \n";
        getline(cin, f[i].nome);
        cin >> f[i].ano;
    }
    //... (continua)
```

# Exemplos VI

```
    ...
    //Mostra os filmes inseridos
    for (int i = 0; i < n; i++)
    {
        cout << f[i].nome << " " << f[i].ano << endl;
    }
    return 0;
}
```

# Exemplos VII

Inicialização: é possível inicializar uma variável do tipo estrutura

```
struct filme
{
    char nome[50];
    int ano;
};
int main()
{
    struct filme f = {"Avatar", 2009};
    cout << "Filme: "<< f.nome << " Ano: " << f.ano;
    return 0;
}
```

# Exemplos VIII

Criar uma estrutura aluno que deve conter os seguintes campos: nome, idade, identidade. Criar um vetor de 10 elementos e logo inserir valores nele. Mostrar os dados do aluno mais novo.

# Exemplos IX

```
struct Aluno{
    string nome;
    int idade;
    int id;
};
int main()
{
    struct Aluno lista[10];
    int menor, pos;
    for (int i = 0; i < 10; i++)
    {
        cout << "Inserir dados do aluno " << i+1 <<endl;
        getline(cin, lista[i].nome);
        cin >> lista[i].idade >> lista[i].id;
    }
    ...
```

# Exemplos X

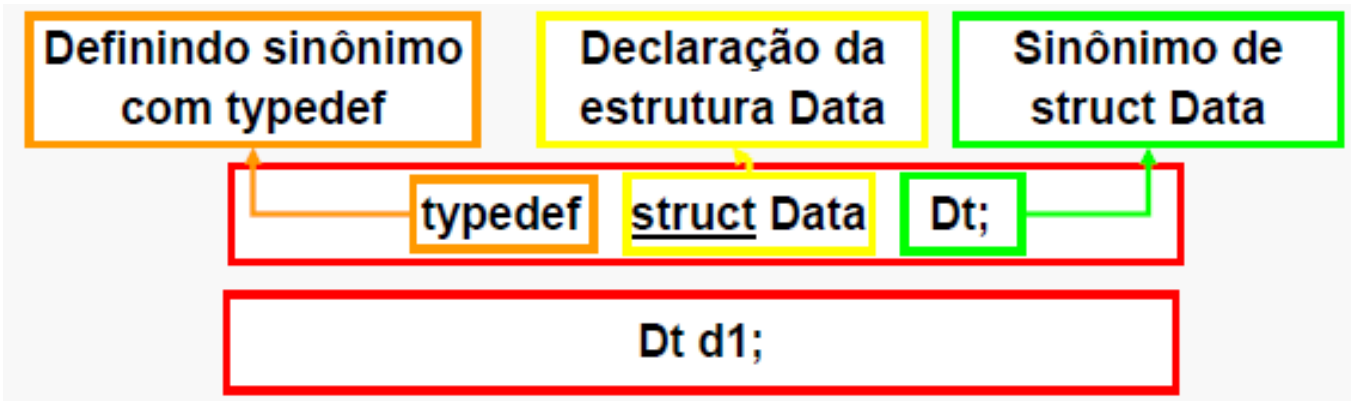
```
    ...
    pos = 0;
    menor = lista[0].idade;
    for (int i = 1; i < 10; i++)
        if (lista[i].idade < menor)
        {
            menor = lista[i].idade;
            pos = i;
        }
    cout << "Aluno mais novo: "
         << lista[pos].nome << " "
         << lista[pos].idade << " "
         << lista[pos].id);
    return 0;
}
```

# Novos Tipos: *typedef* I

- A declaração de uma variável do tipo estrutura emprega a palavra reservada `struct`

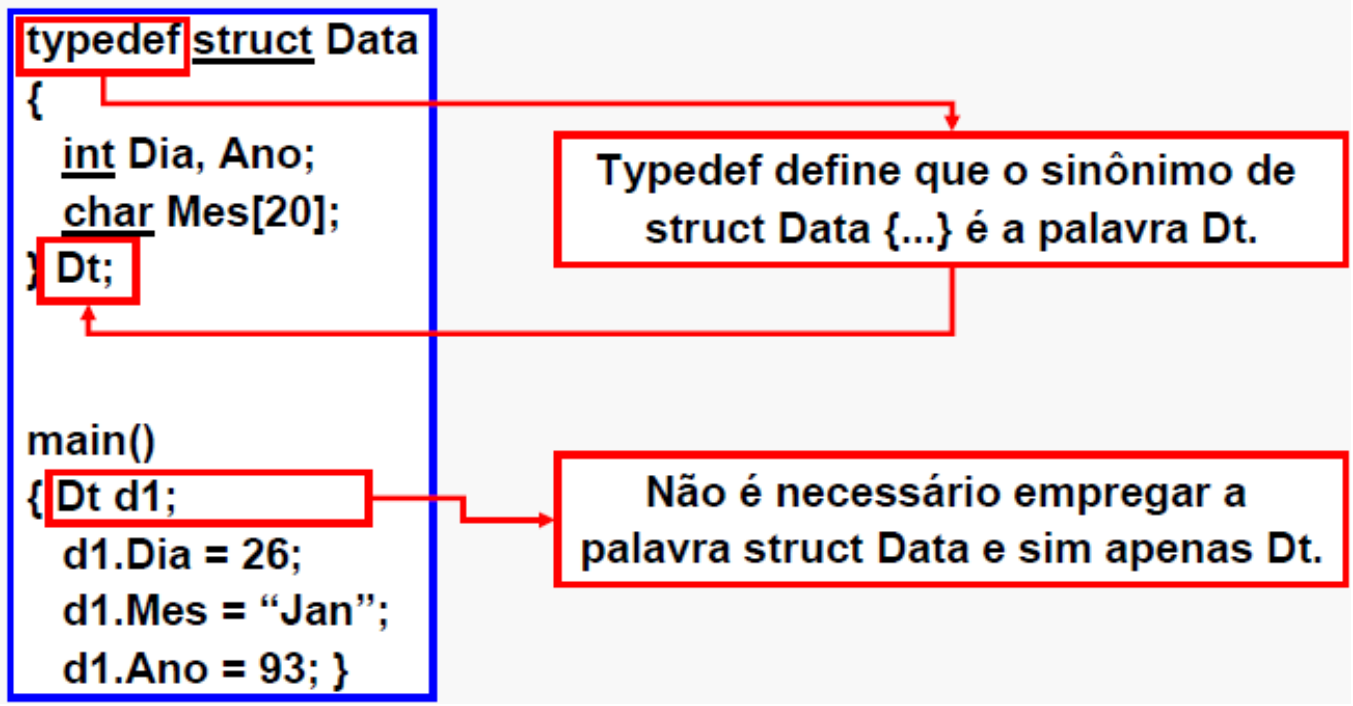
```
struct Data d1;
```

- A palavra reservada `typedef` estabelece um sinônimo para qualquer conjunto de palavras



# Novos Tipos: *typedef* II

- Outra alternativa é empregar a palavra reservada `typedef` ao mesmo tempo em que é definida a estrutura:



# Novos Tipos: *typedef* III

- As três formas possíveis para declarar um novo tipo Data com ou sem typedef são:

```
struct Data
{
  int Dia, Ano;
  char Mes[20];
};


main()
{ struct Data d1;
  d1.Dia = 26;
  d1.Mes = "Jan";
  d1.Ano = 93;}
```

```
struct Data
{
  int Dia, Ano;
  char Mes[20];
};

typedef struct Data Dt;
main()
{ Dt d1;
  d1.Dia = 26;
  d1.Mes = "Jan";
  d1.Ano = 93; }
```

```
typedef struct Data
{
  int Dia, Ano;
  char Mes[20];
} Dt;


main()
{ Dt d1;
  d1.Dia = 26;
  d1.Mes = "Jan";
  d1.Ano = 93; }
```

# Estruturas Compostas I

Declaração de estrutura composta

```
struct nome_do_tipo_da_estrutura
{
    struct nome_de_outra_estrutura nome_campo1;
    tipo_campo2 nome_campo2;
    ...
};
```

Desta forma, uma estrutura pode ser parte de outra estrutura!

# Estruturas Compostas II

Exemplo
Criar uma ficha cadastral de alunos. Essa lista deve contar a seguinte informação: nome, identidade e data de nascimento;

# Estruturas Compostas III

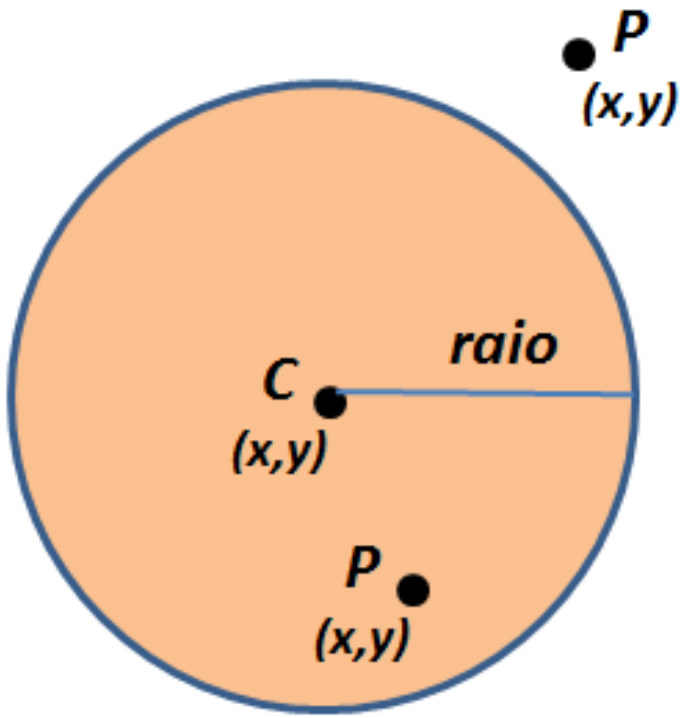
```
struct data
{
    int dia, mes, ano;
};
struct ficha_cadastral
{
    string nome;
    int id;
    struct data nascimento;
};
```

# Estruturas Compostas IV

```
int main()
{
    struct ficha_cadastral alunos;
    cout << "Inserir nome: \n";
    getline(cin, alunos.nome);
    cout << "Inserir identidade: \n";
    cin >> alunos.id;
    cout << "Inserir data de nascimento (dd/mm/aa) \n";
    cin >> alunos.nascimento.dia
        >> alunos.nascimento.mes
        >> alunos.nascimento.ano;
    return 0;
}
```

# Estruturas Compostas V

Definir uma estrutura círculo, essa estrutura deve conter a seguinte informação: ponto de origem (criar estrutura ponto) e raio. Logo, inserir um ponto e responde se esse ponto está dentro do círculo. (criar uma função distancia que receba como dados de entrada os dois pontos)



# Estruturas Compostas VI

```
typedef struct Ponto{
    int x, y;
}TPonto;
typedef struct Circulo{
    TPonto C;
    double raio;
}TCirculo;
double distancia(TPonto, TPonto);
```

# Estruturas Compostas VII

```
int main()
{
    TCirculo circ;
    TPonto P;
    double d;
    cout << "Inserir dados do circulo: \n";
    cout << "Coordenadas em x e y: \n";
    cin >> circ.C.x >> circ.C.y;
    cout << "Raio: \n";
    cin >> circ.raio;
    cout << "Inserir ponto P (x,y) \n";
    cin >> P.x >> P.y;
    ...
```

# Estruturas Compostas VIII

```
    ...
    d = distancia (circ.C, P);
    if (d <= circ.Raio)
        cout << "ponto dentro do circulo \n";
    else
        cout << "ponto fora do circulo \n";
    return 0;
}

double distancia(TPonto A, TPonto B)
{
    double d;
    d = sqrt( pow(A.x - B.x, 2) + pow(A.y - B.y, 2) );
    return d;
}
```

# Exercícios Propostos I

1. Criar um programa que permita ao usuário digitar os dados de 3 alunos, (seus nomes, idades e registros acadêmicos). Em seguida, pedir ao usuário para digitar uma idade, e o programa deve imprimir os nomes e idades dos alunos com idade menor que a digitada.
2. Foi realizada uma pesquisa entre 500 habitantes de uma certa região. De cada habitante foram coletados os dados: idade, sexo, salário e número de filhos. Crie a estrutura de dados adequada para armazenar estas informações e faça uma função que armazene as informações digitadas pelo usuário na estrutura de dados criada. Faça também uma função que calcula a média do salário dos habitantes.

FIM