



# **Introdução à linguagem C**

## **Aula 06 - Registros**

**Professor: Racyus Delano**

**E-mail: [racyuspacifico@univicosa.com.br](mailto:racyuspacifico@univicosa.com.br)**

# Registros

---

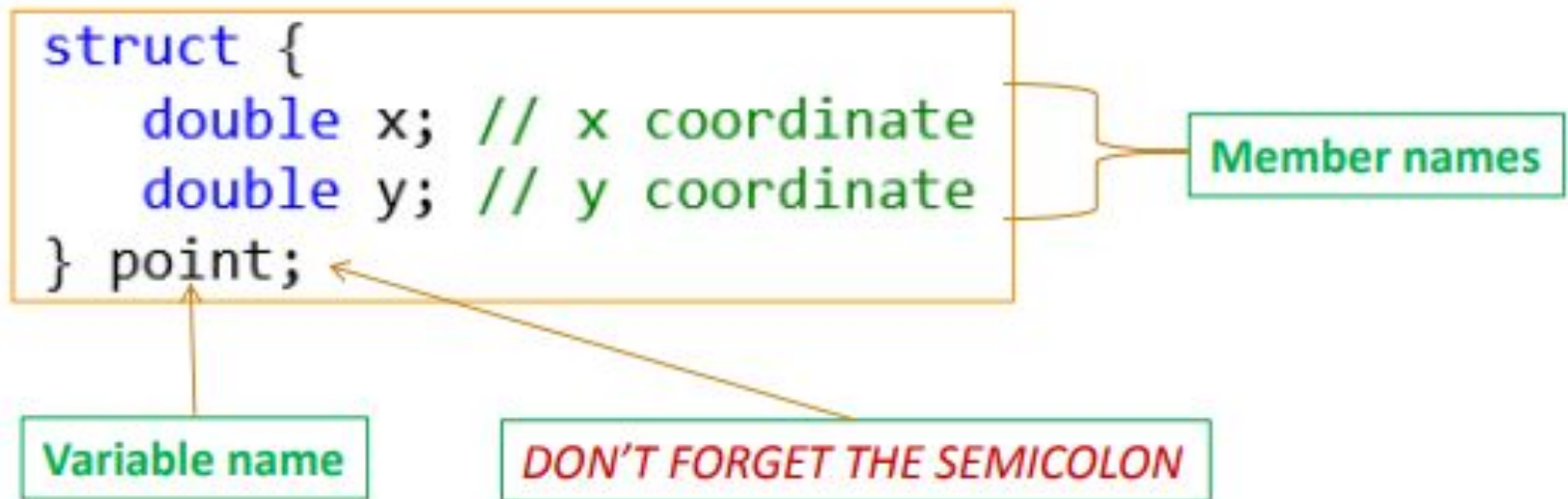
- Podem ser usados para definir um novo tipo de dados que combina diferentes tipos em um único tipo de dados;
- Similar a um template ou modelo de projeto;
- Composto por tipos distintos;
- O registro deve ser definido antes de ser usado;
- Existem três formas diferentes de definir um registro.

# Registros

---

- Definição - 1º forma

- Define o registro com um nome de variável.



# Registros

- Definição - 2ª forma

- Define o registro com uma tag, sem o nome de variável.
- Em seguida, declara as variáveis com o tipo do registro.

```
struct point_t {  
    double x; // x coordinate  
    double y; // y coordinate  
};
```

Structure tag

Member names

*DON'T FORGET THE SEMICOLON*

```
struct point_t point1, point2, point3;
```

# Registros

- Definição - 3º forma

- Define o registro usando **typedef**;
- Typedef** permite definir variáveis do tipo do registro, sem usar a palavra **struct**.

```
typedef struct {  
    long ssn;           // Social Security Number  
    int empType;        // Employee Type  
    float salary;       // Annual Salary  
} employee_t;
```

New type name      DON'T FORGET THE SEMICOLON      Member names

```
employee_t emp;
```

# Registros

---

- **Operador ponto (.)**

- Usado para acessar variáveis membro do registro;
- **Sintaxe:** `nome_registro.variavel_membro`
- As variáveis membro do registro podem ser usadas com outras variáveis do programa.

```
struct point_t {  
    double x; // x coordinate  
    double y; // y coordinate  
};  
void setPoints() {  
    struct point_t point1, point2;  
    point1.x = 7;    // Init point1 members  
    point1.y = 11;  
    point2 = point1; // Copy point1 to point2  
    ...  
}
```

# Registros

---

- **Operador seta (->)**

- Usado para acessar variáveis membro do registro usando ponteiro;
- **Sintaxe com seta:** `ponteiro_registro->variavel_membro`
- **Sintaxe com ponto:** `(*ponteiro_registro).variavel_membro`

```
typedef struct {  
    long ssn;           // Social Security Number  
    int empType;        // Employee Type  
    float salary;       // Annual Salary  
} employee_t;
```



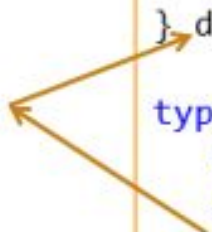
```
employee_t * newEmp(long n, int type, float sal) {  
    employee_t * empPtr = malloc(sizeof(employee_t));  
    empPtr->ssn = n;           // -> operator  
    empPtr->empType = type;    // -> operator  
    (*empPtr).salary = sal;    // dot operator  
    return empPtr;  
}
```

# Registros

---

- Registros aninhados

- Um membro que é de um tipo de registro está aninhado.



```
typedef struct {
    int month;
    int day;
    int year;
} date_t;

typedef struct {
    double height;
    int weight;
    date_t birthday;
} personInfo_t;

// Define variable of type personInfo_t
personInfo_t person;
...

// person.birthday is a member of person
// person.birthday.year is a member of person.birthday
printf("Birth year is %d\n", person.birthday.year);
```



# Registros

---

- **Inicializando registros**

- Podem ser inicializados no momento em que são declarados;
- A ordem é essencial
  - A sequência de valores é usada para inicializar as variáveis;
  - Se não for passado todos os valores aos membros, os membros sem valores são inicializados com 0.

```
typedef struct {  
    int month;  
    int day;  
    int year;  
} date_t;  
  
date_t due_date = {12, 31, 2020};
```

# Registros

---

- **Vetor dentro do registro**
  - Um membro de um registro pode ser um vetor;

```
typedef struct {  
    long ssn;           // SSN  
    double payRate;     // Hourly rate  
    float hoursWorked[7]; // Daily hours worked Sun-Sat  
} timeCard_t;  
  
timeCard_t empTime;  
  
empTime.hoursWorked[5] = 6.5; // Thur hours worked
```

# Registros

---

- **Vetor de registros**

- Pode-se criar um vetor de registros em C.

```
typedef struct {  
    long ssn;           // SSN  
    double payRate;     // Hourly rate  
    float hoursWorked[7]; // Daily hours worked Sun-Sat  
} timeCard_t;  
  
timeCard_t empTime[1000];  
  
// Thur hours worked, emp # 10  
  
empTime[9].hoursWorked[5] = 6.5;
```

# Registros

- Registro como parâmetro de função - cópia ou valor

```
typedef struct {  
    double x; // x coordinate  
    double y; // y coordinate  
} point_t;  
  
void changePoint(point_t p) {  
    printf("x=%.11f, y=%.11f\n", p.x, p.y);  
    //  
    p.x = 3.4;  
    p.y = 4.5;  
}  
  
void mainPoint() {  
    point_t point = {1.2, 2.3};  
    changePoint(point);  
    printf("x=%.11f, y=%.11f\n", point.x, point.y);  
    //  
}
```

x=1.2, y=2.3

x=1.2, y=2.3

# Registros

- Registro como parâmetro de função - referência

```
typedef struct {  
    double x; // x coordinate  
    double y; // y coordinate  
} point_t;  
  
void changePoint(point_t * p) {  
    printf("x=%.11f, y=%.11f\n", p->x, p->y);  
    // x=1.2, y=2.3  
    p->x = 3.4;  
    p->y = 4.5;  
}  
  
void mainPoint() {  
    point_t point = {1.2, 2.3};  
    changePoint(&point);  
    printf("x=%.11f, y=%.11f\n", point.x, point.y);  
    // x=3.4, y=4.5  
}
```

# Exercícios sala de aula

---

1. Dados os seguintes campos de um registro: nome e data de aniversário(dia, mês e ano), desenvolver um programa que mostre para cada um dos meses do ano quem são as pessoas que fazem aniversário e também a idade atual de cada pessoa (pedir ao usuário a data de hoje). Considere um conjunto de 40 pessoas.
  
2. Crie uma estrutura para armazenar dados sobre filmes com as seguintes informações: nome do filme, diretor, gênero e ano. Crie um vetor com informações para 10 filmes. Em seguida, faça as seguintes funções.
  - a. Cadastrar um filme.
  - b. Imprimir os filmes cadastrados.
  - c. Buscar por filmes do gênero aventura produzidos entre 2001 e 2005.

# Exercícios da lista 01

---

- **Data de entrega: 14/03/2024.**

1. Crie um registro para os funcionários de uma empresa com as seguintes informações: número do funcionário, nome, idade, telefone, cargo e salário. O programa deve manter o cadastro de até 100 funcionários. Crie funções que realizem as seguintes tarefas:

- a. Inserir funcionário.
- b. Listar todos os funcionários cadastrados.
- c. Procurar funcionário pelo nome ou pelo número, e imprimir seus dados.
- d. Eliminar o cadastro de um funcionário
- e. Editar as informações de um funcionário, dado o seu número de registro.

# Exercícios da lista 01

---

2. Faça um programa para gerenciamento de compras em um supermercado, com as seguintes características:
- a. O programa deve mostrar um menu de opções ao usuário: adicionar item, remover item, obter valor total da compra, mostrar relatório (lista de itens) e sair.
  - b. O programa deve definir o tipo de registro Item, que pretende representar um item sendo comprado no supermercado. Observação: Campos: nome do item, valor unitário e quantidade.
  - c. O programa deve definir um array (vetor) para permitir criar uma lista de compras no supermercado. O tamanho do array deve ser igual à N, sendo o valor de N definido pelo usuário.