Registros ou structs

Roteiro

Roteiro

Registro

- Estruturas de Dados Heterogêneos;
- Agrupa **várias variáveis** e **de vários tipos** em uma **única estrutura**;
- Funciona como um **container**:
  - Armazena diversos dados relacionados
  - Podendo ser de diversos tipos:
    - ★ Tipos elementares: char, int, float, double;
    - ★ Estruturas homogêneas: ponteiros, vetores, matrizes, strings;
    - ★ Estruturas heterogêneas: outros registros.

Registro - Definindo um novo tipo de dados

- Tipo **Data**: quais informações? de quais tipos?
  - Dia: inteiro
  - Mês: inteiro
  - Ano: inteiro
- Tipo **Disciplina**

Registro - Definindo um novo tipo de dados

- Tipo **Data**: quais informações? de quais tipos?
  - Dia: inteiro
  - Mês: inteiro
  - Ano: inteiro
- Tipo Disciplina

Registro - Definindo um novo tipo de dados

- Tipo **Data**: quais informações? de quais tipos?
  - Dia: inteiro
  - Mês: inteiro
  - Ano: inteiro
- Tipo **Disciplina**
  - Código: inteiro
  - Período: inteiro

Roteiro

Registro

- Tipo **struct** em linguagem C;
- Formato:

```
1     //define a estrutura
2     struct nome_da_estrutura {
3
4         tipo1 campo1;
5
6         tipo2 campo2;
7         ...
8
9     }; //não esquecer do ;
10
```

- Cada um dos elementos (variáveis) é denominado **campo**.

Registro - Declaração do tipo

- Declaração do **Tipo cadastroAluno**

```
1     //Criacao do tipo
2     struct cadastroAluno {
3       int ra;
4       int idade;
5     };
6
7
```

- Uma **struct não é uma variável**, é o nome um **novo tipo de dados**;

Registro - Declaração de variáveis do novo tipo

- Declaração de **Variável do Tipo cadastroAluno**

```
1   //Criacao do tipo e declaracao das variaveis
2   struct cadastroAluno {
3     int ra;
4     int idade;
5   } Alu, Alu1, Alu2;
6
7   //OU
8
9   //Posterior a declaracao do tipo
10  struct cadastroAluno Alu, Alu1 , Alu2;
11
```

Registro - Onde declarar?

```
1
2   #include <stdio.h>
3
4   struct cadastroAluno {
5     int ra;
6     int idade;
7   };
8
9   int main() {
10      struct cadastroAluno Alu1;
11      ...
12      return 0;
13  }
14
```

Registro - Onde declarar?

```
1
2     #include <stdio.h>
3
4     int main() {
5
6         struct cadastroAluno {
7           int ra;
8           int idade;
9         };
10
11        struct cadastroAluno Alu1;
12        ...
13        return 0;
14    }
15
```

Registro - Acesso

Acesso dos campos

Cada campo do registro é acessado através do **ponto '.'**

```
1     //Criacao do tipo
2     struct cadastroAluno {
3         int ra;
4         int idade;
5     };
6
7     //Declaracao da variavel
8     struct cadastroAluno Alu, Alu1 , Alu2;
9     ...
10    printf("%d\n", Alu1.ra);
11    printf("%d\n", Alu1.idade);
12
```

Registro - Inicializar

```
1
2       //Inicializacao na declaracao
3       struct cadastroAluno Alu1 = {231111111, 18};
4
5       //Declaracao
6       struct cadastroAluno Alu2;
7
8       //Inicializacao pela entrada padrao
9       scanf("%d", &Alu2.ra);
10      scanf("%d", &Alu2.idade);
11
12      //Acesso e impressao
13      printf("%d %d\n", Alu1.ra, Alu1.idade);
14      printf("%d %d\n", Alu2.ra, Alu2.idade);
15
```

Registro - Copiar structs

```c

    //Declaracao
    struct cadastroAluno Alu1 = {11111, 23};
    struct cadastroAluno Alu2;


    Alu2 = Alu1; //por copia

    Alu2.ra = 99999; //alterar valor na copia
                     //NAO altera o original!

    //Acesso e impressao
    printf("%d %d\n", Alu1.ra, Alu1.idade); //11111 23
    printf("%d %d\n", Alu2.ra, Alu2.idade); //99999 23


```

Registro

- E se ao invés de guardar a idade do aluno, armazenássemos a data de seu nascimento?
- Tipo **Aluno**
  - Matricula: inteiro
  - Nascimento: ??
    - Data
- Tipo Data:

Registro

- E se ao invés de guardar a idade do aluno, armazenássemos a data de seu nascimento?
- Tipo **Aluno**
  - Matricula: inteiro
  - Nascimento: ??
    - ⋆ Data
- Tipo **Data**:
  - Dia: inteiro
  - Mês: inteiro
  - Ano: inteiro

Roteiro

Estruturas aninhadas: registro dentro de registro

- Registro é um tipo de dados;
- Variáveis deste tipo podem ser declaradas;
- Inclusive dentro de outros registros;

```
1
2  struct nome_estrutura {
3
4      struct outra_estrutura nome_campo1;
5
6      tipo_campo2 nome_campo2;
7
8      ...
9
10 };
11
```

Estruturas aninhadas: registro dentro de registro

```c
struct data {
    int dia, mes, ano;
};

struct ficha_cadastral {
    long int cpf;
    struct data nascimento;
    struct data cadastro;
};

```

```c
//Declaracao
struct ficha_cadastral aluno;

//Leitura: acesso por '.'
scanf("%ld", &aluno.cpf);

scanf("%d %d %d", &aluno.nascimento.dia,
                  &aluno.nascimento.mes,
                  &aluno.nascimento.ano);

scanf("%d %d %d", &aluno.cadastro.dia,
                  &aluno.cadastro.mes,
                  &aluno.cadastro.ano);
```

Estruturas aninhadas: registro dentro de registro

```c
struct data {
    int dia, mes, ano;
};

struct disciplina {
    int codigo;
    int periodo;
};

struct aluno {
    int ra;
    struct data nascimento;
    struct data cadastro;
};

struct matricula {
    struct disciplina materia;

    //struct dentro de struct que possui struct
    struct aluno participante;
};

```

Registro - Auto-referenciamento

- Tipo **Disciplina**
  - Código: inteiro
  - Período: inteiro
  - Pré-requisito: ??

```
* Disciplina (ponteiro - endereço)

struct disciplina {
    int codigo;
    int periodo;
    struct disciplina *requisito;
};
```

Registro - Auto-referenciamento

- Tipo **Disciplina**
  - Código: inteiro
  - Período: inteiro
  - Pré-requisito: ??
    - ★ Disciplina (ponteiro - endereço)

```
1
2  struct disciplina {
3        int codigo;
4        int periodo;
5        struct disciplina *requisito;
6  };
7
```

Roteiro

Typedef: sinônimo - apelido

```
1
2  struct Data {
3      int dia;
4      int mes;
5      int ano;
6  };
7
8  typedef struct Data Dt;
9
```

```
1
2  Dt cadastro = {1, 1, 2001};
3  Dt nascimento;
4
```

Typedef: sinônimo - apelido

```
1
2  typedef struct data Data;
3  struct data {
4      int dia, mes, ano;
5  };
6
7  //OU
8
9  typedef struct {
10     int codigo;
11     float preco;
12 }Produto;
13
```

```
1
2  Data cadastro;
3  struct data nascimento;
4
5  Produto prod1;
6  struct prod prod2;
7
```

Roteiro

Vamos praticar

- Faça um registro denominado "Produto" que possua os campos código (inteiro), data de fabricação (tipo data), data de vencimento (tipo data) e preço (ponto flutuante).
- Faça um algoritmo que utilize essa estrutura para armazenar 3 produtos, cujos os conteúdos serão inseridos através da entrada padrão.
  - Imprima na tela o código e a data de vencimento de cada produto lido.
- Imprima na tela o código e a data de vencimento do produto que vencerá primeiro.

```
1  //Declaracao do tipo
2  struct nome_estrutura {
3   tipo_campo1 nomeCampo1;
4   tipo_campo2 nomeCampo2;
5   ...
6  };
7  typedef struct nome_estrutura novoT;
8
9  //Declaracao das variaveis
10 novoT var1, var2;
11
12 //Acessar os campos '.' (ponto)
13 scanf("%d", &var1.campo1);
14 printf("%d\n", var1.campo1);
```