

DCC405 – ESTRUTURA DE DADOS I

Aula 02 – Arrays e Structs

Arrays e Structs

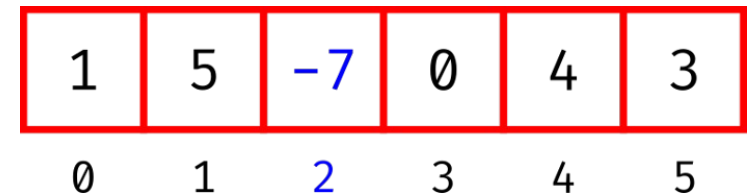
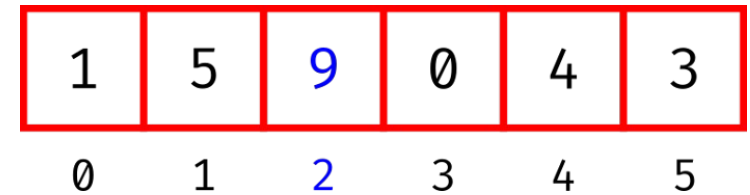
- A maioria das estruturas de dados que iremos trabalhar na disciplina (**listas**, **pilhas**, **filas**, etc) serão construídas a partir de outras estruturas básicas, como **arrays** e **structs**.

Arrays

- **Arrays** permitem organizar coleções de dados de um mesmo tipo (números inteiros, por exemplo) de forma sequencial. A ordem, tanto na hora da inserção como da extração, é dada por indexação.

O quê isso quer dizer? Se criarmos um array com espaço para n elementos, teremos também n índices, começando do **0 até $n-1$** , para identificar univocamente cada um dos elementos do array.

```
6   int num[] = {1, 5, 9, 0, 4, 3};
7
8   // aqui o índice 2 é usado para identificar
9   // o terceiro elemento do array
10
11  num[2] = -7;
```



Arrays

- **Arrays** são estruturas muito convenientes para organizar dados. Eles podem ser iterados (percorridos por um loop), e dessa forma, aplicar algoritmos eficientes de ordenação, de busca e outros. Também, seus itens podem ser acessados de forma imediata **se o índice for conhecido**.

```
6   int num[] = {1, 5, 9, 0, 4, 3};
7
8   // aqui o índice 2 é usado para identificar
9   // o terceiro elemento do array
10
11  num[2] = -7;
```



1	5	9	0	4	3
0	1	2	3	4	5

1	5	-7	0	4	3
0	1	2	3	4	5

Structs

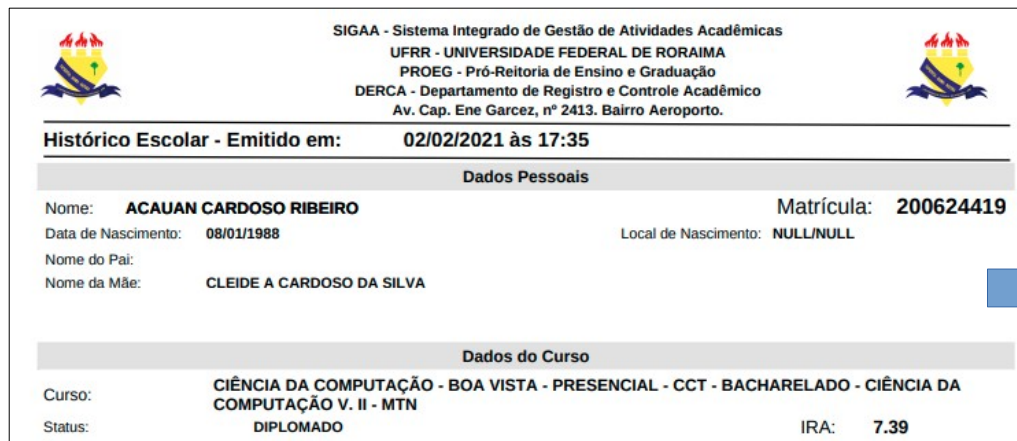
Então, porque Structs?

- É que as vezes gostaríamos de agrupar conjuntos de dados que, embora também estejam relacionados entre si, não necessariamente vão ser do mesmo tipo nem vão respeitar uma determinada ordem. Pensa nos dados de um aluno da UFRR, por exemplo:

	SIGAA - Sistema Integrado de Gestão de Atividades Acadêmicas UFRR - UNIVERSIDADE FEDERAL DE RORAIMA PROEG - Pró-Reitoria de Ensino e Graduação DERCA - Departamento de Registro e Controle Acadêmico Av. Cap. Ene Garcez, nº 2413. Bairro Aeroporto.	
Histórico Escolar - Emitido em: 02/02/2021 às 17:35		
Dados Pessoais		
Nome:	ACAUAN CARDOSO RIBEIRO	Matrícula: 200624419
Data de Nascimento:	08/01/1988	Local de Nascimento: NULL/NULL
Nome do Pai:		
Nome da Mãe:	CLEIDE A CARDOSO DA SILVA	
Dados do Curso		
Curso:	CIÊNCIA DA COMPUTAÇÃO - BOA VISTA - PRESENCIAL - CCT - BACHARELADO - CIÊNCIA DA COMPUTAÇÃO V. II - MTN	
Status:	DIPLOMADO	IRA: 7.39

Structs

- Temos dados os quais faz todo sentido agrupar, já que eles descrevem uma entidade, o aluno de uma determinada instituição. Também temos que essas informações combinam dados com características bem diferentes um dos outros: dados textuais, dados numéricos, números inteiros e de ponto flutuante. Inclusive, na linguagem C, os dados textuais (strings), eles são, tecnicamente falando, arrays de caracteres (char).



SIGAA - Sistema Integrado de Gestão de Atividades Acadêmicas
UFRR - UNIVERSIDADE FEDERAL DE RORAIMA
PROEG - Pró-Reitoria de Ensino e Graduação
DERCA - Departamento de Registro e Controle Acadêmico
Av. Cap. Ene Garcez, nº 2413. Bairro Aeroporto.

Histórico Escolar - Emitido em: 02/02/2021 às 17:35

Dados Pessoais

Nome: **ACAUAN CARDOSO RIBEIRO** Matrícula: **200624419**
Data de Nascimento: **08/01/1988** Local de Nascimento: **NULL/NULL**
Nome do Pai:
Nome da Mãe: **CLEIDE A CARDOSO DA SILVA**

Dados do Curso

Curso: **CIÊNCIA DA COMPUTAÇÃO - BOA VISTA - PRESENCIAL - CCT - BACHARELADO - CIÊNCIA DA COMPUTAÇÃO V. II - MTN**
Status: **DIPLOMADO** IRA: **7.39**

```
18 struct aluno {  
19     char nome[50];  
20     char data_nascimento[9];  
21     int matricula;  
22     char curso[50];  
23     double ira;  
24     //outros campos...  
25 };
```

Definindo uma Estrutura

- Uma estrutura pode ser definida de formas diferentes. No corpo da estrutura encontram-se os membros, ou seja, as variáveis de diversos tipos que comporão esse **tipo de dado heterogêneo** definido pelo usuário. Depois de definida uma estrutura, uma (ou mais) variável do tipo estrutura deve ser definida, para permitir a manipulação dos membros da estrutura.

Structs – Exemplo 1

- Neste exemplo o nome, ou **Etiqueta (TAG)**, da estrutura é colocado logo em seguida da palavra-chave Struct. A Etiqueta da estrutura é nomeada como ALUNO. Ao final da estrutura, são declaradas várias variáveis do tipo da estrutura, isto é, aluno_especial, aluno_regular e aluno_ouvinte, são variáveis do tipo aluno.

```
31 struct aluno { // tipo de dado
32     int codigo;
33     char nome[200];
34     float nota;
35 }; // definição da struct
36
37 //Declaração das variaveis
38 struct aluno aluno_especial, aluno_regular, aluno_ouvinte;
```


Structs – Exemplo 2

- Neste exemplo o nome da estrutura é colocado ao final, logo em seguida ao fechamento da estrutura, isto é, a Etiqueta da estrutura é nomeada depois. Após a declaração da estrutura, são definidas as várias variáveis do tipo da estrutura.

```
41  
42     struct { // tipo de dado  
43         int codigo;  
44         char nome[200];  
45         float nota;  
46     } aluno, aluno_especial, aluno_regular, aluno_ouvinte; //variaveis  
47
```

Structs – Exemplo 3 (prefiro esse)

- Neste exemplo, usei a palavra-chave **Typedef** para definir uma estrutura, antes da palavra-chave **Struct**. O nome da estrutura é colocado ao final, logo em seguida ao fechamento da estrutura. Após a declaração são definidas várias variáveis do tipo da estrutura.

```
50     typedef struct {  
51         int codigo;  
52         char nome[200];  
53         float nota;  
54     } aluno;  
55  
56     aluno aluno_especial, aluno_regular, aluno_ouvinte;
```

Acessando os membros da Estrutura

- Você pode atribuir valores aos membros das estruturas diretamente, e em qualquer parte do programa, conforme a seguir:

```
62     aluno_especial.codigo = 10;  
63     strcpy(aluno_especial.nome, "Manoel");  
64     aluno_especial.nota = 10.0;
```

- Para atribuir um valor a uma string é necessário utilizar a função **Strcpy** (CPY = copiar; STR = string). A função copiará o que está dentro das aspas duplas para o membro STRING da estrutura.

Inicializando a Estrutura

- É sempre importante inicializar as variáveis que serão utilizadas em nosso programa, com as estruturas não é diferente. Você pode fazer isso de duas formas: ou atribui valores padrão diretamente aos membros da estrutura, ou então cria uma função, que será chamada no programa principal, para fazer isso por você.

```
62     aluno_especial.codigo = 10;
63     strcpy(aluno_especial.nome, "Manoel");
64     aluno_especial.nota = 10.0;
65
66     //FUNÇÃO PARA INICIALIZAR UMA ESTRUTURA SIMPLES
67     void inicializa(){
68         aluno_especial.codigo = 0;
69         strcpy(aluno_especial.nome, "\0");
70         aluno_especial.nota= 0.0;
71     }
```

Ponteiros e Structs

- Embora iremos revisar ponteiros nos próximos slides, este assunto é importante neste momento. Sabemos que existem duas formas de acessar os membros de uma struct: uma, usando o operador **.** (**ponto**), outra, usando o operador **->** (**seta**):

```
73      // usando operador ponto
74      aluno_regular.codigo = 001234567;
75
76      // usando operador seta
77      aluno_ouvinte->codigo = 007654321;
```

Ponteiros e Structs

- **Do que vai depender isso?** Da forma em que for necessário acessar a estrutura. Se for **diretamente**, usaremos o operador **ponto**. Se for **indiretamente** através de um **ponteiro**, será o operador **seta**:

```
83 // declaramos uma estrutura "joao" de tipo Aluno
84 // e acessamos diretamente com o operador ponto
85
86 aluno joao;
87 joao.nota = 7.8;
88
89 // declaramos um ponteiro a uma estrutura e
90 // acessamos indiretamente com o operador seta
91
92 aluno *ptr_joao = &joao;
93 ptr_joao->nota = 9.5;
```

Ponteiros e Structs

- O operador seta é uma forma de se fazer **dereferenciação** de um ponteiro a uma estrutura usando uma sintaxe mais limpa, mas isso também poderia ser feito da forma tradicional. Essas duas expressões são equivalentes:

```
60 // usando o operador seta
61 ptr->membro = valor;
62
63 // usando a forma tradicional
64 (*ptr).membro = valor;
```