

CP63B-DPGR3A

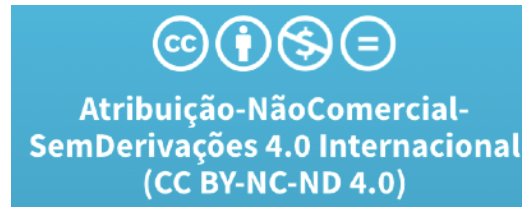
COMPUTAÇÃO 2

APNP 04 - Registros (structs)

Prof. Rafael Gomes Mantovani

Licença

Este trabalho está licenciado com uma Licença CC BY-NC-ND 4.0:



maiores informações:

https://creativecommons.org/licenses/by-nc-nd/4.0/deed.pt_BR

Roteiro



- 1** Objetivo
- 2** Registros
- 3** Como usar
- 4** Exercícios
- 5** Referências

Roteiro

- 1 Objetivo**
- 2 Registros**
- 3 Como usar**
- 4 Exercícios**
- 5 Referências**

Objetivo

- Em cada parte de um programa geralmente há várias variáveis associadas à realização de uma tarefa específica;
- Por causa disso, é conveniente ter um modo de agrupar um conjunto de variáveis relacionadas;
- Vetores e matrizes agrupam uma série de variáveis do **MESMO TIPO**, cada uma identificada por índices;
- Se, por outro lado, quisermos um tipo de agrupamento que englobe variáveis de **TIPOS DIFERENTES**, ou no qual cada variável possa ser identificada por um nome específico usamos **STRUCTS**.

Struct/Registro

- Usamos um tipo de estrutura chamado de **registro** (mais conhecido por seu nome em inglês, **struct**, uma abreviação de *structure*, ‘estrutura’);
- Esse recurso da linguagem C permite que o usuário “defina” **seus próprios tipos de dados** a partir dos tipos primitivos da linguagem (*int*, *float*, *char*, etc.);
- **Struct** contém um conjunto de variáveis, que têm tipos fixados e são identificadas por nomes (como as variáveis comuns);

Roteiro



- 1 Objetivo
- 2 Registros
- 3 Como usar
- 4 Exercícios
- 5 Referências

Struct: definição

- Por exemplo, uma **struct** que representa uma pessoa pode conter as seguintes variáveis:

```
char nome[50];  
int idade;  
float peso;  
float altura;
```

- Essas variáveis são denominadas de membros da struct.
- Cada **struct** é em si uma variável!!!

Struct: definição

- Por exemplo, uma **struct** que representa um produto numa compra pode conter as seguintes variáveis:

```
char descricao[30];  
int quantidade;  
float preco_unitario;  
float desconto;  
float preco_total;
```

- Essas variáveis são denominadas de membros da struct.
- Cada **struct** é em si uma variável!!!

Struct: definição

- Uma **struct/registro** é declarada usando a palavra chave **struct** seguida de um bloco (delimitado por chaves) contendo as declarações dos membros, como se fossem declarações de variáveis comuns.

```
struct Produto{  
    char descricao[30];  
    int quantidade;  
    float preco_unitario;  
    float desconto;  
    float preco_total;  
};
```

Definição da struct

Ponto e vírgula
(definição da struct é um comando)

```
struct Produto produto_1, produto_2, produto_3;
```

Declaração de 3
variáveis do
tipo Produto

Struct: acesso

- Para acessar campos de um registro, usamos o operador . (um ponto), colocando à esquerda dele o nome da variável que contém o registro, e à direita o nome do campo.
- No exemplo anterior:

```
struct Produto{  
    char descricao[30];  
    int quantidade;  
    float preco_unitario;  
    float desconto;  
    float preco_total;  
};
```

```
struct Produto produto_1, produto_2, produto_3;
```

- Pode-se acessar o preço do produto_1 usando a expressão:
 produto_1.preco_unitario

Struct: resumo

- Declaração de um tipo de registro

```
struct nome_do_tipo{  
    /* declarações dos membros */  
};
```

- Declaração de um registro

```
nome_da_struct nome_da_variável;
```

- Acesso de membros de um registro

```
variavel.nome_do_membro;
```

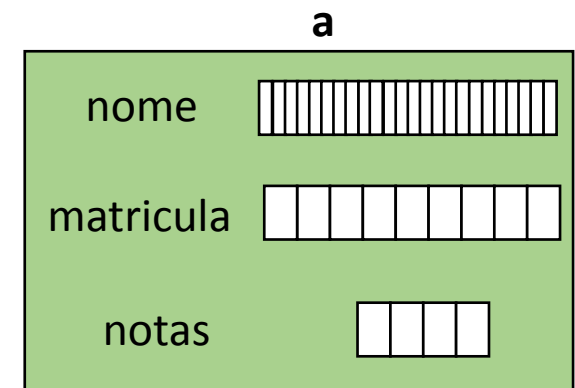
Exemplo

```
1  #include<stdio.h>
2  #include<stdlib.h>
3
4  struct Aluno {
5      char nome[30];
6      char matricula[10];
7      float notas[4];
8  };
9
10 int main()
11 {
12     struct Aluno a;
13     int i;
14
15     printf("Qual o nome do aluno? ");
16     fgets(a.nome, 30, stdin);
17
18     printf("Qual a matricula? ");
19     fgets(a.matricula, 10, stdin);
20
21     for(i=0; i<4; i++)
22     {
23         printf("Digite a nota do %d bimestre: ", i+1);
24         scanf("%f", &a.notas[i]);
25     }
26
27     return 0;
28 }
```

Nenhuma variável foi criada
até esse momento!!!

Definição da **struct**

Declarando/criando
uma variável **a** do "tipo"
Aluno



Roteiro



- 1** Objetivo
- 2** Registros
- 3** Como usar
- 4** Exercícios
- 5** Referências

Como usar: inicializando

- Pode-se inicializar a **struct** de forma estática, como ocorre com variáveis comuns, vetores e matrizes;

```
struct Produto{  
    char descricao[30];  
    int quantidade;  
    float preco_unitario;  
    float desconto;  
    float preco_total;  
};
```

```
struct Produto agua_sanitaria = {"Qboa", 1, 7.99, 0, 7.99};
```

```
strcpy(agua_sanitaria.descricao, "Qboa");  
agua_sanitaria.quantidade = 1;  
agua_sanitaria.preco_unitario = 7.99;  
agua_sanitaria.desconto = 0;  
agua_sanitaria.preco_total = 7.99;
```

Como usar: inicializando

- Pode-se inicializar a *struct* de forma estática, como ocorre com variáveis comuns, vetores e matrizes;

```
struct Aluno{  
    char nome[30];  
    char matricula[10];  
    float notas[4];  
};  
  
struct Aluno a = {"Joao", "123456789", 7.8, 5.4, 6.1, 9.3};
```

```
strcpy(a.nome, "Joao");  
strcpy(a.matricula, "123456789");  
a.notas[0] = 7.8;  
a.notas[1] = 5.4;  
a.notas[2] = 6.1;  
a.notas[3] = 9.3;
```


Como usar: atribuição

- Uma variável estrutura pode ser atribuída a outra do mesmo tipo por meio de uma atribuição simples

```
struct Produto{
    char descricao[30];
    int quantidade;
    float preco_unitario;
    float desconto;
    float preco_total;
};

struct Produto feijao = {"redondo", 1, 20.0, 0, 20.0};
struct Produto feijao_carioca;

feijao_carioca = feijao;
```

Como usar: Typedef

- É possível nomear um tipo baseado em uma estrutura
 - Para isso utiliza-se typedef na declaração

```
typedef struct{  
    char descricao[30];  
    int quantidade;  
    float preco_unitario;  
    float desconto;  
    float preco_total;  
}Mercadoria;
```

```
int main(){  
    Mercadoria feijao = {"redondo", 1, 20.0, 0, 20.0};  
    Mercadoria feijao_carioca;  
  
    feijao_carioca = feijao;
```

Mercadoria é o nome do tipo

Agora não existe mais a
necessidade de:

struct Mercadoria feijao;

Como usar: Typedef

- É possível nomear um tipo baseado em uma estrutura
 - Para isso utiliza-se typedef na declaração

```
typedef struct{
    int dia, mes, ano;
}Data;

int main()
{
    Data atual;

    return 0;
}
```

Como usar: Typedef

```
typedef struct{  
    int dia, mes, ano;  
}Data;
```

```
int main()  
{  
    Data atual;  
  
    return 0;  
}
```

São equivalentes



```
struct Data{  
    int dia, mes, ano;  
};
```

```
int main()  
{  
    struct Data atual;  
  
    return 0;  
}
```

Dica: Limpando buffer do teclado

- Toda a informação que digitamos no teclado é armazenada em um *buffer* e fica disponível para nossa utilização;
 - **Buffer:** é uma região de memória física utilizada para armazenar temporariamente os dados;
- Quando usamos a função `scanf()`, `getch()`, `fgets()`, etc. ela recupera a informação do buffer;
- Porém, ela pode deixar “sujeira” no *buffer*, comprometendo futuras leituras

Dica: Limpando buffer do teclado

```
int main(){
    int i=0;
    char c;

    for(i=0;i<5;i++){
        printf("Digite o %d caracter: ", i+1);
        scanf("%c", &c);
    }
}
```

```
Digite o 1 caracter: a
Digite o 2 caracter: Digite o 3 caracter: b
Digite o 4 caracter: Digite o 5 caracter: c

Process returned 0 (0x0)   execution time : 14.029 s
Press any key to continue.
_
```

Não foi possível
informar o segundo
e quarto caracteres

Dica: Limpando buffer do teclado

```
int main(){
    int i=0;
    char c;

    for(i=0;i<5;i++){
        printf("Digite o %d character: ", i+1);
        scanf("%c", &c);
        setbuf(stdin, NULL);
    }
}
```

→ Limpando o buffer

`stdin` = *standard input* = teclado

```
Digite o 1 character: a
Digite o 2 character: b
Digite o 3 character: c
Digite o 4 character: d
Digite o 5 character: e

Process returned 0 (0x0)   execution time : 7.646 s
Press any key to continue.
```

Como usar: Vetor de Struct

- Cadastrando informações de 4 pessoas

```
1  #include<stdlib.h>
2  #include<stdio.h>
3
4  int main()
5  {
6      char nome1[30], nome2[30], nome3[30], nome4[30];
7      int idade1, idade2, idade3, idade4;
8      char rua1[30], rua2[30], rua3[30], rua4[30];
9      int numero1, numero2, numero3, numero4;
10
11
12      return 0;
13  }
```

Método trabalhoso e confuso. Sem **struct**.

```
1  #include<stdlib.h>
2  #include<stdio.h>
3
4  typedef struct{
5      char nome[30];
6      int idade;
7      char rua[30];
8      int numero;
9  } Pessoa;
10
11  int main()
12  {
13      Pessoa p1, p2, p3, p4;
14
15
16      Pessoa p[4];
17
18      return 0;
19  }
```

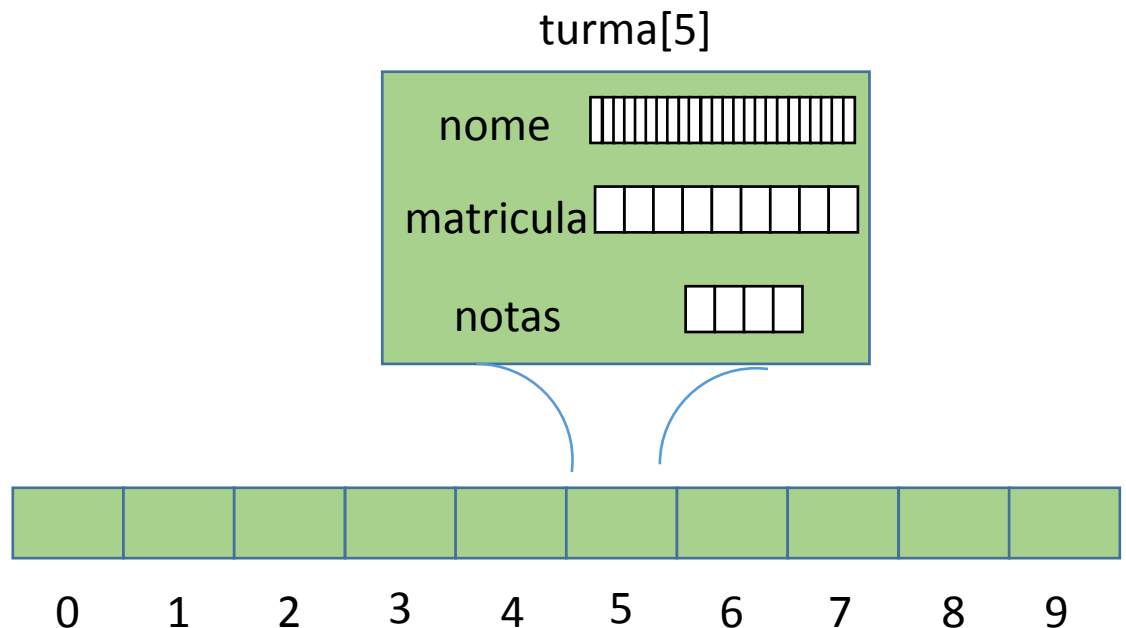
- Usando **struct**
- Usando um vetor de **struct**

Como usar: Vetor de Struct

- Vetores são usados para guardar diversas variáveis do mesmo tipo (*float, int, char, etc.*)
- Porque não criar um vetor de **structs**?

```
typedef struct{
    char nome[30];
    char matricula[10];
    float notas[4];
}Aluno;

int main()
{
    Aluno turma[10];
    ...
}
```



Como usar: Vetor de Struct

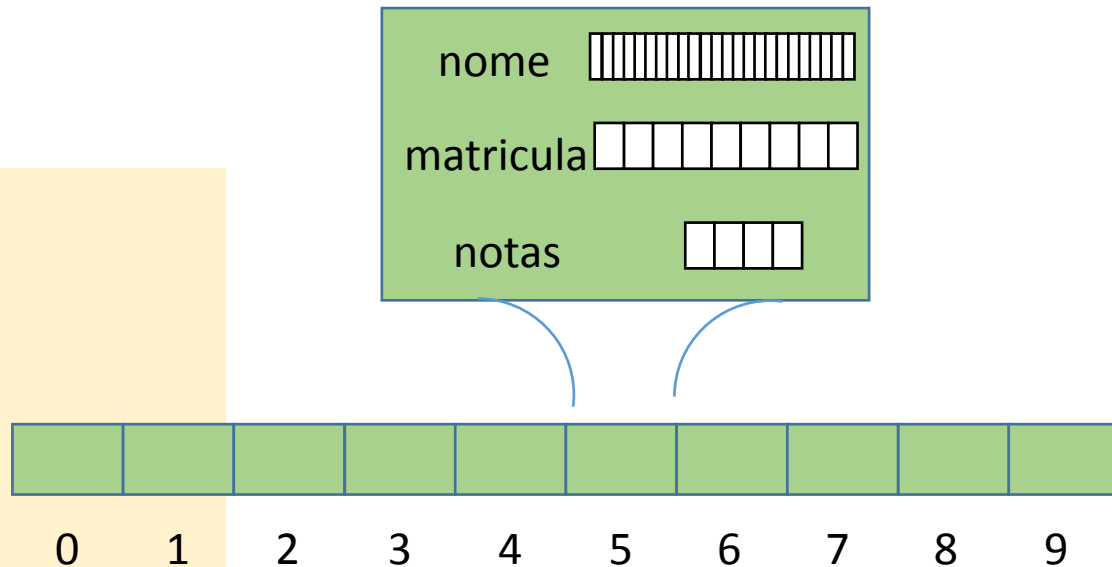
```
typedef struct{
    char nome[30];
    char matricula[10];
    float notas[4];
}Aluno;

int main()
{
    Aluno turma[10];

    strcpy(turma[5].nome, "Joao");
    strcpy(turma[5].matricula, "123456789");
    turma[5].notas[0] = 8.9;
    turma[5].notas[1] = 7.5;
    turma[5].notas[2] = 6.0;
    turma[5].notas[3] = 2.5;

    ...

}
```



Como usar: Estruturas Aninhadas


- Assim como estruturas de controle `while`, `for` e `if` podem ser aninhadas, as estruturas também podem:

```
typedef struct{  
    char nome[50], rua[50], cep[9];  
    int idade, numero;  
}Pessoa;
```

- As variáveis `nome` e `idade` estão relacionadas à pessoa;
- As variáveis `rua`, `cep` e `numero` estão relacionadas ao endereço da pessoa.

Como usar: Estruturas Aninhadas

- Então, pode-se dividir a estrutura em duas:



```
typedef struct{  
    char rua[50], cep[9];  
    int numero;  
}Endereco;
```

```
typedef struct{  
    char nome[50];  
    int idade;  
    Endereco end;  
}Pessoa;
```

Como usar: Estruturas Aninhadas

```
typedef struct{
    char rua[50], cep[9];
    int numero;
}Endereco;

typedef struct{
    char nome[50];
    int idade;
    Endereco end;
}Pessoa;
```

```
int main(){
    Pessoa p;

    strcpy(p.nome, "Cardoso");
    p.idade = 45;
    strcpy(p.end.cep, "123456789");
    p.end.numero = 4;
    strcpy(p.end.rua, "Rua Alan Turing");

    puts(p.nome);
    printf("%d", p.idade);
    puts(p.end.cep);
    puts(p.end.rua);
    printf("%d", p.end.numero);

    return 0;
}
```

Como usar: Estruturas Aninhadas

```
typedef struct{  
    int rodas, capacidade;  
    char marca[20], modelo[20];  
    char combustivel[10];  
    float peso;  
}Veiculo;
```

```
typedef struct{  
    int cilindradas;  
    Veiculo v;  
}Motocicleta;
```

```
typedef struct{  
    int motores;  
    float alturaMaxima;  
    Veiculo v;  
}Aviao;
```

int motores
float alturaMaxima



int rodas, capacidade
char marca, modelo, combustivel
float peso

int cilindradas



Roteiro



- 1 Objetivo
- 2 Registros
- 3 Como usar
- 4 Exercícios
- 5 Referências

Exercícios

Universidade Tecnológica Federal do Paraná – Câmpus Apucarana
Computação 2 (CP63B) - DPGR3A – Structs
Prof. Dr. Rafael Gomes Mantovani

Instruções:

- Antes de codificar, esboce em um papel a sequência de passos necessários para criar o seu programa. Isso ajuda a programar a solução;
- Crie um arquivo .c para cada um dos exercícios. Por exemplo, na resolução do exercício 01, crie um arquivo chamado 'ex01.c'.

Exercícios Structs

Exercício 1. Escreva um trecho de código para fazer a criação dos novos tipos de dados conforme solicitado abaixo:

- a) **Horário:** composto por hora, minutos e segundos;
- b) **Data:** composto por dia, mês e ano;
- c) **Compromisso:** local, horário e texto que descreve o compromisso.

Exercícios - Solução 1 a)

Universidade Tecnológica Federal do Paraná – Câmpus Apucarana
Computação 2 (CP63B) - DPGR3A – Structs
Prof. Dr. Rafael Gomes Mantovani

Instruções:

- Antes de codificar, esboce em um papel a sequência de passos necessários para criar o seu programa. Isso ajuda a programar a solução;
- Crie um arquivo .c para cada um dos exercícios. Por exemplo, na resolução do exercício 01, crie um arquivo chamado 'ex01.c'.

Exercícios Structs

Exercício 1. Escreva um trecho de código para fazer a criação dos novos tipos de dados conforme solicitado abaixo:

- a) **Horário:** composto por hora, minutos e segundos;
- b) **Data:** composto por dia, mês e ano;
- c) **Compromisso:** local, horário e texto que descreve o compromisso.

Exercícios - Solução 1 a)

```
5 #include <stdio.h>
6
7 /* define um horario */
8 typedef struct {
9     int hora;
10    int minutos;
11    int segundos;
12 } Horario;
13
14 /* ----- */
15 /* ----- */
16
17 int main(int argc, const char * argv[]) {
18
19     // Horario var = {14, 00, 00}; // tambem funciona
20     Horario var;
21     var.hora = 14;
22     var.minutos = 00;
23     var.segundos = 00;
24
25     printf("* Horário: %d:%d:%d\n", var.hora,
26         var.minutos, var.segundos);
27
28     return 0;
29 }
```

Define um novo tipo para **Horario**: ele contém três campos inteiros para: horas, minutos e segundos.

Exercícios - Solução 1 a)

```
5 #include <stdio.h>
6
7 /* define um horario */
8 typedef struct {
9     int hora;
10    int minutos;
11    int segundos;
12 } Horario;
13
14 /* ----- */
15 /* ----- */
16
17 int main(int argc, const char * argv[]) {
18
19 // Horario var = {14, 00, 00}; // tambem funciona
20 Horario var;
21 var.hora = 14;
22 var.minutos = 00;
23 var.segundos = 00;
24
25 printf("* Horário: %d:%d:%d\n", var.hora,
26     var.minutos, var.segundos);
27
28 return 0;
29 }
```

Define um novo tipo para **Horario**: ele contém três campos inteiros para: horas, minutos e segundos.

Define uma variável do tipo Horário

Inicializa os campos

Imprime os valores

Exercícios - Solução 1b)

Universidade Tecnológica Federal do Paraná – Câmpus Apucarana
Computação 2 (CP63B) - DPGR3A – Structs
Prof. Dr. Rafael Gomes Mantovani

Instruções:

- Antes de codificar, esboce em um papel a sequência de passos necessários para criar o seu programa. Isso ajuda a programar a solução;
- Crie um arquivo .c para cada um dos exercícios. Por exemplo, na resolução do exercício 01, crie um arquivo chamado 'ex01.c'.

Exercícios Structs

Exercício 1. Escreva um trecho de código para fazer a criação dos novos tipos de dados conforme solicitado abaixo:

- a) **Horário:** composto por hora, minutos e segundos;
- b) **Data:** composto por dia, mês e ano;
- c) **Compromisso:** local, horário e texto que descreve o compromisso.

Exercícios - Solução 1 b)

```
34 #include <stdio.h>
35
36 typedef struct {
37     int dia;
38     int mes;
39     int ano;
40 } Data;
41
42 int main() {
43
44     Data d = {19, 9, 1987};
45
46     printf("* Data: %d/%d/%d\n",
47         d.dia, d.mes, d.ano);
48
49     return(0);
50 }
```

Exercícios - Solução 1 b)

```
34 #include <stdio.h>
35
36 typedef struct {
37     int dia;
38     int mes;
39     int ano;
40 } Data;
41
42 int main() {
43
44     Data d = {19, 9, 1987};
45
46     printf("* Data: %d/%d/%d\n",
47         d.dia, d.mes, d.ano);
48
49     return(0);
50 }
```

Define um novo tipo para **Data**: ele contém três campos inteiros para: dia, mes e ano.

Exercícios - Solução 1 b)

```
34 #include <stdio.h>
35
36 typedef struct {
37     int dia;
38     int mes;
39     int ano;
40 } Data;
41
42 int main() {
43
44     Data d = {19, 9, 1987};
45
46     printf("* Data: %d/%d/%d\n",
47           d.dia, d.mes, d.ano);
48
49     return(0);
50 }
```

Define um novo tipo para **Data**: ele contém três campos inteiros para: dia, mes e ano.

Define e inicializa uma variável do tipo **Data**

Imprime os valores

Exercícios - Solução 1c)

Universidade Tecnológica Federal do Paraná – Câmpus Apucarana
Computação 2 (CP63B) - DPGR3A – Structs
Prof. Dr. Rafael Gomes Mantovani

Instruções:

- Antes de codificar, esboce em um papel a sequência de passos necessários para criar o seu programa. Isso ajuda a programar a solução;
- Crie um arquivo .c para cada um dos exercícios. Por exemplo, na resolução do exercício 01, crie um arquivo chamado 'ex01.c'.

Exercícios Structs

Exercício 1. Escreva um trecho de código para fazer a criação dos novos tipos de dados conforme solicitado abaixo:

- a) **Horário:** composto por hora, minutos e segundos;
- b) **Data:** composto por dia, mês e ano;
- c) **Compromisso:** local, horário e texto que descreve o compromisso.

Exercícios - Solução 1c)

```
52 #include <stdio.h>
53 #include <string.h>
54
55 typedef struct {
56     int hora;
57     int minutos;
58     int segundos;
59 } Horario;
60
61 typedef struct {
62     char localidade[100];
63     char texto[100];
64     Horario horario;
65 } Compromisso;
66
```

Exercícios - Solução 1c)

```
52 #include <stdio.h>
53 #include <string.h>
```

```
54
55 typedef struct {
56     int hora;
57     int minutos;
58     int segundos;
59 } Horario;
```

```
60
61 typedef struct {
62     char localidade[100];
63     char texto[100];
64     Horario horario;
65 } Compromisso;
```

Define um novo tipo para **Horario**: ele contém três campos inteiros para: horas, minutos e segundos.
(o mesmo do ex 1 a)

Define um novo tipo para **Compromisso**: ele contém duas strings (localidade, texto) e um horário do tipo **Horario**

Exercícios - Solução 1c)

```
67 int main() {
68
69     Compromisso c;
70     c.horario.hora = 13;
71     c.horario.minutos = 50;
72     c.horario.segundos = 0;
73     strcpy(c.localidade, "UTFPR");
74     strcpy(c.texto, "Aula de Comp 2");
75
76     // imprimir o compromisso
77     printf("* Compromisso: %s\n", c.texto);
78     printf("* Localidade: %s\n", c.localidade);
79     printf("* Horário: %d:%d:%d\n", c.horario.hora,
80         c.horario.minutos, c.horario.segundos);
81
82     return(0);
83 }
```

Define uma variável Compromisso

Inicializa os campos

Copia texto para as strings

Imprime os valores

Exercícios

Universidade Tecnológica Federal do Paraná – Câmpus Apucarana
Computação 2 (CP63B) - DPGR3A – Structs
Prof. Dr. Rafael Gomes Mantovani

Instruções:

- Antes de codificar, esboce em um papel a sequência de passos necessários para criar o seu programa. Isso ajuda a programar a solução;
- Crie um arquivo .c para cada um dos exercícios. Por exemplo, na resolução do exercício 01, crie um arquivo chamado 'ex01.c'.

Mais exercícios no Moodle :)

Exercício 1. Escreva um trecho de código para fazer a criação dos novos tipos de dados conforme solicitado abaixo:

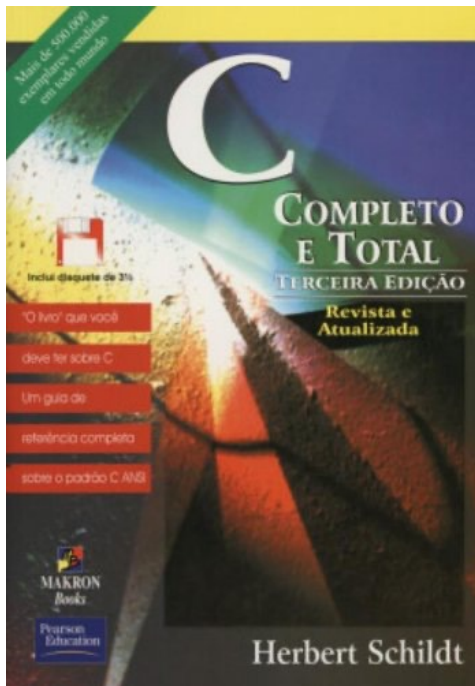
- a) **Horário:** composto por hora, minutos e segundos;
- b) **Data:** composto por dia, mês e ano;
- c) **Compromisso:** local, horário e texto que descreve o compromisso.

Roteiro

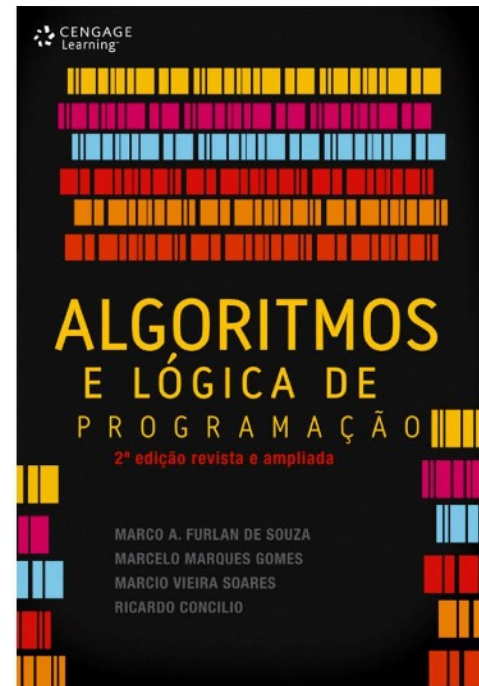


- 1** Objetivo
 - 2** Registros
 - 3** Como usar
 - 4** Exercícios
 - 5** Referências
- 

Referências



[Schildt, 1997]



[de Souza et al, 2011]

Referências

- [Schildt, 1997] SCHILDT, H. **C Completo e Total**. 3. ed. São Paulo: Pearson, 1997.
- [de Souza et al, 2011] DE SOUZA, M. A. F. et al. **Algoritmos e lógica de programação**. 2. ed. São Paulo: Cengage Learning, 2011.

Perguntas?

Prof. Rafael G. **Mantovani**

rafaelmantovani@utfpr.edu.br