

Curso: Ciência da Computação

Disciplina: Estrutura de Dados 1

Professor: Clayton Zambon

2. Introdução

2.3. Cadeia de Caracteres

2.4. Structs

2. Introdução

2.3. Cadeia de Caracteres

CADEIAS DE CARACTERES

- Um texto ou seu nome é representado por uma cadeia de caracteres;
- A Linguagem C não oferece o tipo `character`;
- Devemos utilizar o tipo `CHAR`;
 - O tipo `CHAR` possui o tamanho de 1 byte ou 8 bits.
- Esses caracteres são representados internamente por códigos numéricos:
 - Exemplo: o número 97 representa a letra “a”. Este número advém da tabela ASCII.
- Na linguagem C a diferença de caracteres inteiros está apenas na forma como são tratados.

2. Introdução

2.3. Cadeia de Caracteres

CADEIAS DE CARACTERES

```
Caracteres01.c
1  #include <stdio.h>
2
3  int main(int argc, char** argv)
4  {
5      char c = 97;
6
7      printf("%d %c", c, c);
8      return 0;
9  }
10
```

C:\Particular Zambon\aulas\Estrutura de Dados\src\Caracteres\Car

97 a
O Processo retornou 0 tempo de execução : 0.163 s
Pressione uma tecla para continuar...

```
Caracteres03.c
1  #include <stdio.h>
2
3  //Função que verifica se é letra ou número
4  int letra(char c)
5  {
6      if(c < '0' || c > '9')
7      {
8          return 1; //Aqui retorna 1 se é verdadeiro
9      }
10     return 0; //Aqui retorna 0 se é falso
11 }
12
13 int main(int argc, char** argv)
14 {
15     //Imprime e chama a função
16     if (letra('1') == 1)
17     {
18         printf("NAO e um numero\n");
19     }
20     else
21     {
22         printf("E um numero\n");
23     }
24     return 0;
25 }
```

2. Introdução

2.3. Cadeia de Caracteres

CADEIAS DE CARACTERES

- Strings em C são representadas por vetores do tipo CHAR terminadas obrigatoriamente por um caracter NULO representado por “\0” (barra zero);
- Se você deseja armazenar uma cadeia de caracteres, deve reservar uma posição adicional para o caracter de fim da cadeia, o “\0” (barra zero);
- O especificador de formato que deve ser utilizado é o “%s”;

2. Introdução

2.3. Cadeia de Caracteres

CADEIAS DE CARACTERES

```
Caracteres04.c
1  #include <stdio.h>
2
3  //Cadeia de caracteres
4  int main(int argc, char** argv)
5  {
6      //Imprimindo um nome na tela
7      char nome[] = "Clayton Zambon";
8      printf("Nome Completo: %s", nome);
9      return 0;
10 }
11
```

C:\Particular Zambon\aulas\Estrutura de Dados\src\Caracteres\Car

Nome Completo: Clayton Zambon
O Processo retornou 0 tempo de execução : 0.120 s
Pressione uma tecla para continuar...

```
Caracteres05.c
1  #include <stdio.h>
2  //Biblioteca para manipular Strings
3  #include <string.h>
4
5  //Cadeia de caracteres
6  int main(int argc, char** argv)
7  {
8      //Imprimindo um nome na tela
9      char nome[] = "Clayton Zambon";
10     //Função STRLEN retorna o tamanho da String
11     int tam = strlen(nome);
12     int i = 0;
13     //Imprime caracter por caracter
14     while(i < tam)
15     {
16         //Veja que utilizamos %c aqui
17         printf("%c", nome[i]);
18         i++;
19     }
20     return 0;
21 }
```

C:\Particular Zambon\aulas\Estrutura de Dados\src\Caracteres\Cara

Clayton Zambon
O Processo retornou 0 tempo de execução : 0.097 s
Pressione uma tecla para continuar...

2. Introdução

2.3. Cadeia de Caracteres

CADEIAS DE CARACTERES

```
Caracteres06.c
1 #include <stdio.h>
2 //Biblioteca para manipular Strings
3 #include <string.h>
4
5 //Cadeia de caracteres
6 int main(int argc, char** argv)
7 {
8     //Imprimindo um nome da tela
9     //de uma forma diferente
10    char nome[] = "Clayton Zambon";
11    int i = 0;
12    //Imprime caracter por caracter
13    while(nome[i] != '\0')
14    {
15        //Veja que utilizamos %c aqui
16        printf("%c", nome[i]);
17        i++;
18    }
19    return 0;
20 }
```

C:\Particular Zambon\aulas\Estrutura de Dados\src\Caracteres\Ca
Clayton Zambon
O Processo retornou 0 tempo de execução : 0.163 s
Pressione uma tecla para continuar...

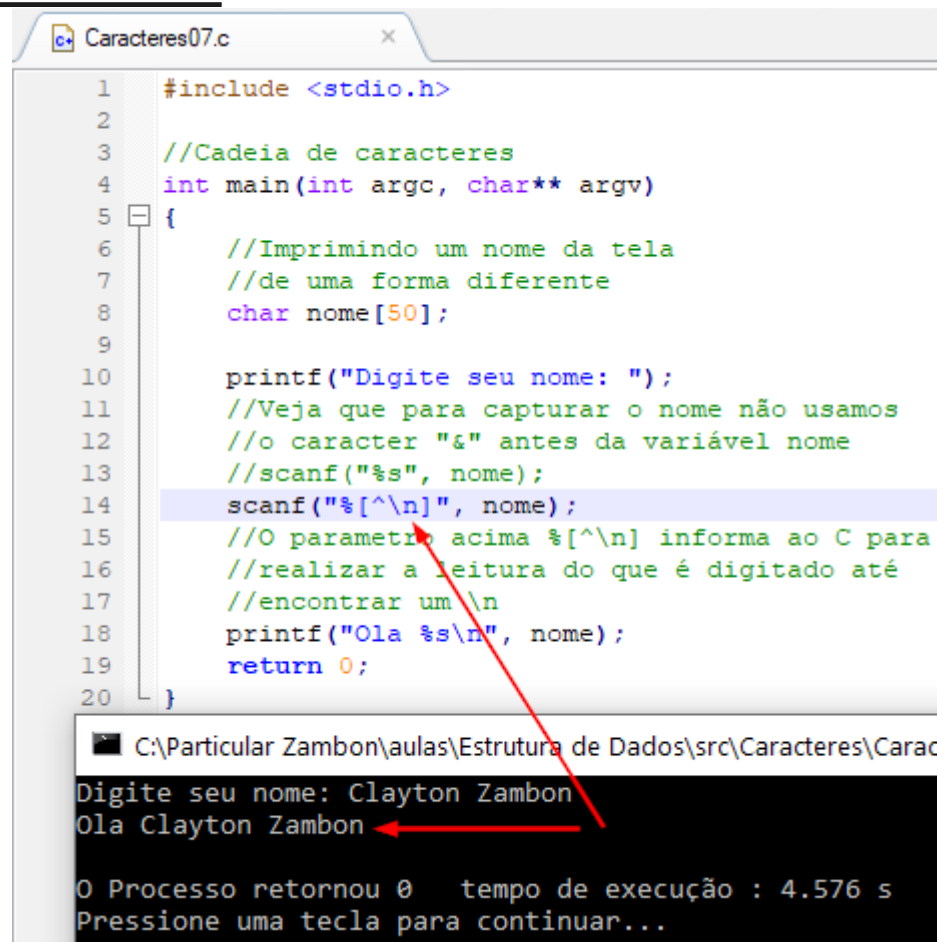
```
Caracteres07.c
1 #include <stdio.h>
2
3 //Cadeia de caracteres
4 int main(int argc, char** argv)
5 {
6     //Imprimindo um nome da tela
7     //de uma forma diferente
8     char nome[50];
9
10    printf("Digite seu nome: ");
11    //Veja que para capturar o nome não usamos
12    //o caracter "&" antes da variável nome
13    scanf("%s", nome);
14    //scanf("%[^\\n]", nome);
15    //O parametro acima %[^\n] informa ao C para
16    //realizar a leitura do que é digitado até
17    //encontrar um \n
18    printf("Ola %s\n", nome);
19    return 0;
20 }
```

C:\Particular Zambon\aulas\Estrutura de Dados\src\Caracteres\Carac
Digite seu nome: Clayton Zambon
Ola Clayton
O Processo retornou 0 tempo de execução : 5.053 s
Pressione uma tecla para continuar...

2. Introdução

2.3. Cadeia de Caracteres

CADEIAS DE CARACTERES



The image shows a code editor window titled 'Caracteres07.c' with the following C code:

```
1  #include <stdio.h>
2
3  //Cadeia de caracteres
4  int main(int argc, char** argv)
5  {
6      //Imprimindo um nome da tela
7      //de uma forma diferente
8      char nome[50];
9
10     printf("Digite seu nome: ");
11     //Veja que para capturar o nome não usamos
12     //o caracter "&" antes da variável nome
13     //scanf("%s", nome);
14     scanf("%[^\n]", nome);
15     //O parametro acima %[^\n] informa ao C para
16     //realizar a leitura do que é digitado até
17     //encontrar um \n
18     printf("Ola %s\n", nome);
19     return 0;
20 }
```

Below the code editor is a terminal window showing the execution of the program. The output is:

```
C:\Particular Zambon\aulas\Estrutura de Dados\src\Caracteres\Carac
Digite seu nome: Clayton Zambon
Ola Clayton Zambon
O Processo retornou 0 tempo de execução : 4.576 s
Pressione uma tecla para continuar...
```

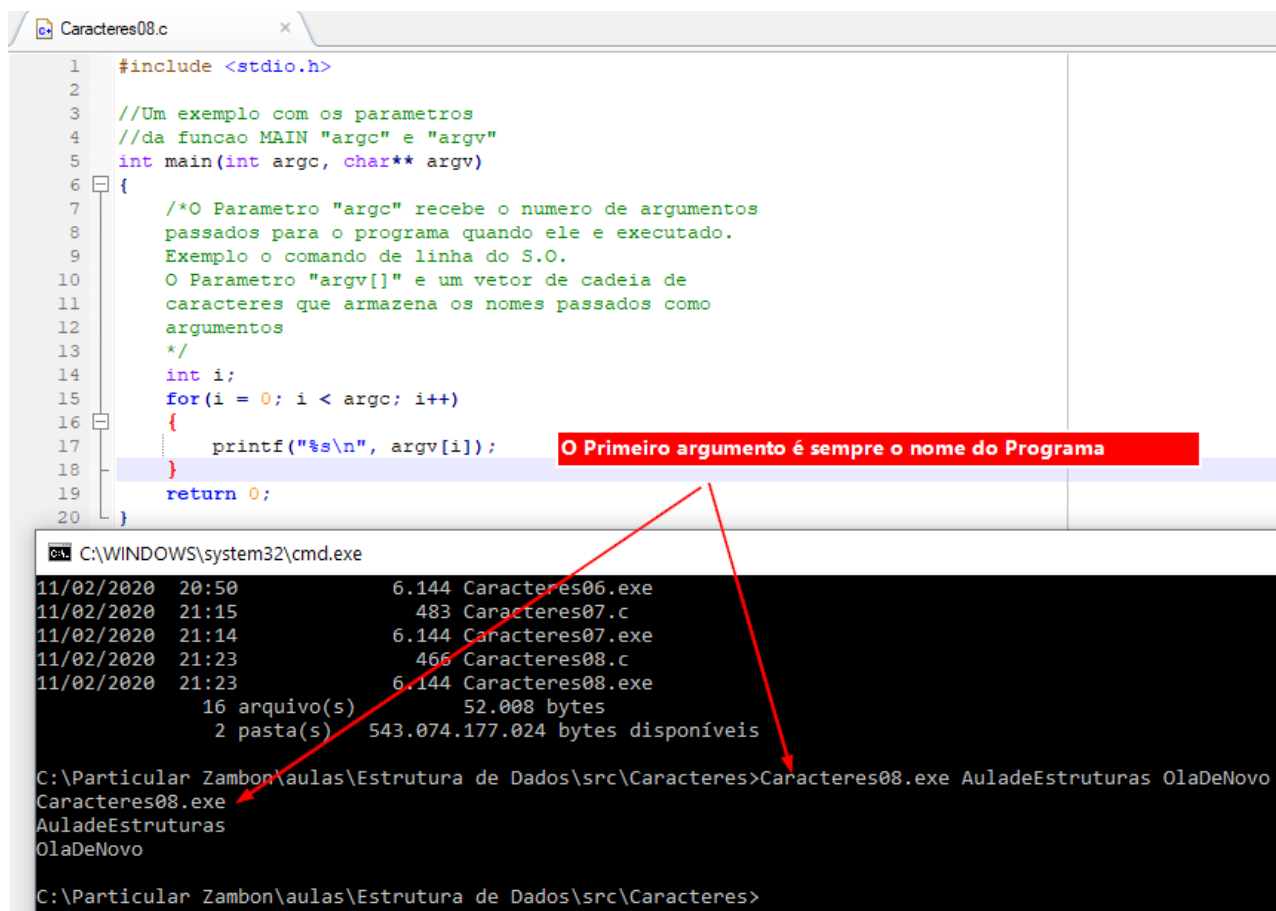
Two red arrows point from the code to the terminal output. One arrow points from the `scanf("%[^\n]", nome);` line (line 14) to the input "Clayton Zambon". The other arrow points from the `printf("Ola %s\n", nome);` line (line 18) to the output "Ola Clayton Zambon".

2. Introdução

2.3. Cadeia de Caracteres

CADEIAS DE CARACTERES

- O primeiro argumento
É sempre o nome do
Programa.



The image shows a C program named 'Caracteres08.c' and its execution output in a command prompt. The program is designed to demonstrate how command-line arguments are passed to a C program. It includes a header file <stdio.h> and a main function that takes two arguments: 'argc' (the number of arguments) and 'argv' (an array of pointers to the arguments). The program iterates through the 'argv' array, printing each argument on a new line. A red box highlights the first argument, 'Caracteres08.exe', with the text 'O Primeiro argumento é sempre o nome do Programa'. The command prompt shows the program being executed with three arguments: 'AuladeEstruturas', 'OlaDeNovo', and 'Caracteres08.exe'.

```
1 #include <stdio.h>
2
3 //Um exemplo com os parametros
4 //da funcao MAIN "argc" e "argv"
5 int main(int argc, char** argv)
6 {
7     /*O Parametro "argc" recebe o numero de argumentos
8     passados para o programa quando ele e executado.
9     Exemplo o comando de linha do S.O.
10    O Parametro "argv[]" e um vetor de cadeia de
11    caracteres que armazena os nomes passados como
12    argumentos
13    */
14    int i;
15    for(i = 0; i < argc; i++)
16    {
17        printf("%s\n", argv[i]);
18    }
19    return 0;
20 }
```

O Primeiro argumento é sempre o nome do Programa

```
C:\WINDOWS\system32\cmd.exe
11/02/2020 20:50 6.144 Caracteres06.exe
11/02/2020 21:15 483 Caracteres07.c
11/02/2020 21:14 6.144 Caracteres07.exe
11/02/2020 21:23 466 Caracteres08.c
11/02/2020 21:23 6.144 Caracteres08.exe
16 arquivo(s) 52.008 bytes
2 pasta(s) 543.074.177.024 bytes disponíveis

C:\Particular Zambon\aulas\Estrutura de Dados\src\Caracteres>Caracteres08.exe AuladeEstruturas OlaDeNovo
Caracteres08.exe
AuladeEstruturas
OlaDeNovo

C:\Particular Zambon\aulas\Estrutura de Dados\src\Caracteres>
```

2. Introdução

2.4. Structs

STRUCTS – (Tipos Estruturados)

- Até o momento utilizamos os tipos básico da Linguagem C como “CHAR”, “INT”, “FLOAT”;
- Para desenvolver programas mais complexos iremos precisar trabalhar de forma mais abstrata para representar os dados;
- Imagine que queremos representar um dado chamado PESSOA:
 - Uma pessoa tem RG, Nome, Idade, Peso;
- A Linguagem C possui formas para estruturar dados complexos nos quais as informações são compostas por diversos campos e tipos como o exemplo acima.

2. Introdução

2.4. Structs

STRUCTS – (Tipos Estruturados)

- Podemos criar os Tipos Estruturados que podem ser usados para representar informações do nosso exemplo PESSOA;
- Este tipo que criamos chamamos de STRUCT ou estrutura;
- Podemos definir um STRUCT cujos campos são compostos por vários valores de tipos mais simples como no nosso exemplo Pessoa que possui RG(int), Nome(char), idade(int) Peso (float);
- Vejamos um exemplo implementado no código:

2. Introdução

2.4. Structs

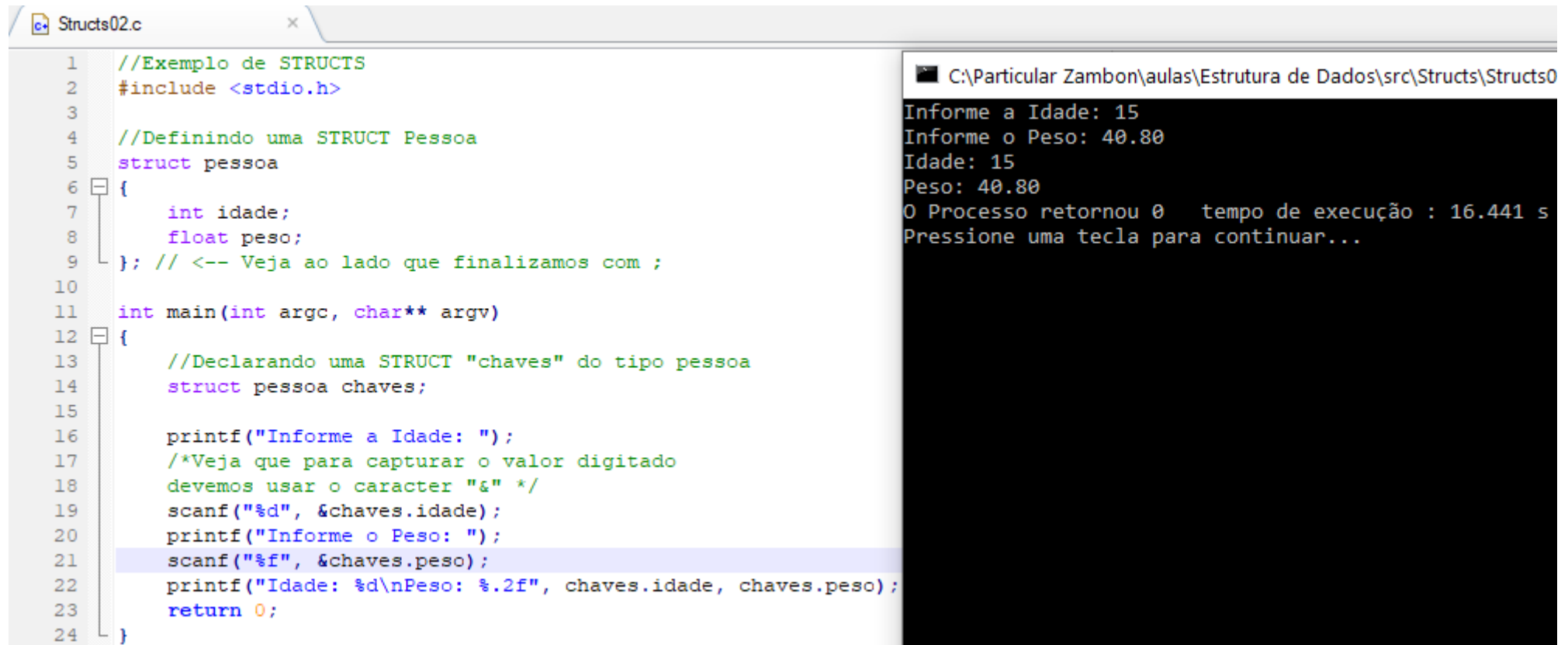
STRUCTS – (Tipos Estruturados)

```
*Structs01.c x
1 //Exemplo de STRUCTS
2 #include <stdio.h>
3
4 //Definindo uma STRUCT Pessoa
5 struct pessoa
6 {
7     int idade;
8     float peso;
9 }; // <-- Veja ao lado que finalizamos com ;
10
11 int main(int argc, char** argv)
12 {
13     //Declarando uma STRUCT "chaves" do tipo pessoa
14     struct pessoa chaves;
15
16     /* Veja que para acessar os campos da minha
17     STRUCT pessoa eu digito o nome declarado seguido
18     de um "." (ponto) ou do caracter "->" e o
19     campo desejado*/
20     chaves.idade = 15; //ou chaves->idade = 15;
21     chaves.peso = 60.21; //ou chaves->peso = 60.21;
22
23     printf("Idade: %d\nPeso: %.2f\n", chaves.idade, chaves.peso);
24     return 0;
25 }
```

2. Introdução

2.4. Structs

STRUCTS – (Tipos Estruturados)



The image shows a C program named 'Structs02.c' in a code editor. The program defines a struct 'pessoa' with 'idade' (int) and 'peso' (float) fields. In the 'main' function, it declares a variable 'chaves' of type 'pessoa', prompts the user for age and weight, reads the input, and prints the values. The execution output on the right shows the user entering '15' for age and '40.80' for weight, with the program printing these values and a message to press a key to continue.

```
1 //Exemplo de STRUCTS
2 #include <stdio.h>
3
4 //Definindo uma STRUCT Pessoa
5 struct pessoa
6 {
7     int idade;
8     float peso;
9 }; // <-- Veja ao lado que finalizamos com ;
10
11 int main(int argc, char** argv)
12 {
13     //Declarando uma STRUCT "chaves" do tipo pessoa
14     struct pessoa chaves;
15
16     printf("Informe a Idade: ");
17     /*Veja que para capturar o valor digitado
18     devemos usar o caracter "&" */
19     scanf("%d", &chaves.idade);
20     printf("Informe o Peso: ");
21     scanf("%f", &chaves.peso);
22     printf("Idade: %d\nPeso: %.2f", chaves.idade, chaves.peso);
23     return 0;
24 }
```

C:\Particular Zambon\aulas\Estrutura de Dados\src\Structs\Structs0
Informe a Idade: 15
Informe o Peso: 40.80
Idade: 15
Peso: 40.80
O Processo retornou 0 tempo de execução : 16.441 s
Pressione uma tecla para continuar...

2. Introdução

2.4. Structs

STRUCTS

- Dica: Você pode utilizar as variações de acesso à STRUCT para definir no código o que é variável STRUCT e o que é PONTEIRO. Veja no código ao lado um exemplo.

```
Structs03.c
1 //Exemplo de STRUCTS
2 //PONTEIROS PARA ESTRUTURAS
3 #include <stdio.h>
4
5 //Definindo uma STRUCT Pessoa
6 struct pessoa
7 {
8     int idade;
9     float peso;
10 }; // <-- Veja ao lado que finalizamos com ;
11
12 int main(int argc, char** argv)
13 {
14     //Declarando uma STRUCT "chaves" do tipo pessoa
15     struct pessoa chaves;
16     //Declarando um ponteiro para a Struct
17     struct pessoa *ponteiro;
18     //Inicializando o ponteiro
19     ponteiro = &chaves;
20
21     /*Porque foi utilizado ( ) aqui?
22     Porque o operador "conteudo de", que é o "*"
23     tem ordem de precedencia menor que o operador
24     de acesso que é o "." */
25     (*ponteiro).idade = 33;
26     (*ponteiro).peso = 70.56;
27     //Variavel estrutura acessando o campo com "."
28     printf("Idade: %d\n", chaves.idade);
29     //Variavel ponteiro da estrutura acessando com "->"
30     printf("Peso: %.2f", ponteiro->peso);
31     /*Dica: Voce pode utilizar as variacoes acima para diferenciar
32     no seu codigo quando a variavel e struct acessando com "." ou
33     quando e um ponteiro acessando com "->"/>
34     return 0;
35 }
```

2. Introdução

2.4. Structs

STRUCTS

- O exemplo ao lado mostra como trabalhar com alocação dinâmica de STRUCTS em ponteiros.

```
Struts04.c
1 //Exemplo de STRUCTS
2 //Alocacao Dinamica para STRUCTS
3 #include <stdio.h>
4 #include <stdlib.h>
5
6 //Definindo uma STRUCT Pessoa
7 struct pessoa
8 {
9     int idade;
10    float peso;
11 }; // <-- Veja ao lado que finalizamos com ;
12
13 int main(int argc, char** argv)
14 {
15     //Declarando um ponteiro do tipo STRUCT pessoa
16     struct pessoa *flecha;
17
18     /*Alocando de modo dinamico uma estrutura
19     e armazenando o endereco da area alocada
20     em flecha*/
21     flecha = (struct pessoa*)malloc(sizeof(struct pessoa));
22
23     flecha->idade = 26;
24     flecha->peso = 78.20;
25     printf("Idade: %d\n", flecha->idade);
26     printf("Peso: %.2f", flecha->peso);
27
28     return 0;
29 }
```

C:\Particular Zambon\aulas\Estrutura de Dados\src\Structs\Structs04.exe

Idade: 26
Peso: 78.20
O Processo retornou 0 tempo de execucao : 0.042 s
Pressione uma tecla para continuar...

2. Introdução

2.4. Structs

STRUCTS

- A linguagem C permite criarmos tipos, veja:

```
Structs05.c
1 //Exemplo de STRUCTS
2 //Criando Tipos em C
3 #include <stdio.h>
4 #include <stdlib.h>
5
6 //Criando uma Struct do Tipo tpessoa
7 typedef struct pessoa
8 {
9     int idade;
10    float peso;
11 }tpessoa;
12
13 int main(int argc, char** argv)
14 {
15     /*Veja que agora para declarar o ponteiro da
16     Struct basta informar tpessoa e o nome do ponteiro*/
17     //struct pessoa *flecha; Antes era assim
18     tpessoa *flecha; //Após criar o tipo tpessoa
19
20     /*Alocando de modo dinamico uma estrutura
21     e armazenando o endereco da area alocada
22     em flecha*/
23     flecha = (struct pessoa*)malloc(sizeof(struct pessoa));
24
25     flecha->idade = 26;
26     flecha->peso = 78.20;
27     printf("Idade: %d\n", flecha->idade);
28     printf("Peso: %.2f", flecha->peso);
29
30     return 0;
31 }
```

C:\Particular Zambon\aulas\Estrutura de Dados\src\Structs\Structs05.exe

Idade: 26
Peso: 78.20
O Processo retornou 0 tempo de execução : 0.069 s
Pressione uma tecla para continuar...

2. Introdução

2.4. Structs

STRUCTS

- A linguagem C permite criarmos tipos, veja:

```
Struts05.c
1 //Exemplo de STRUCTS
2 //Criando Tipos em C
3 #include <stdio.h>
4 #include <stdlib.h>
5
6 //Criando uma Struct do Tipo tpessoa
7 typedef struct pessoa
8 {
9     int idade;
10    float peso;
11 }tpessoa;
12
13 //Criando o tipo real
14 typedef float real;
15
16 int main(int argc, char** argv)
17 {
18     /*Veja que agora para declarar o ponteiro da
19     Struct basta informar tpessoa e o nome do ponteiro*/
20     //struct pessoa *flecha; Antes era assim
21     tpessoa *flecha; //Após criar o tipo tpessoa
22     real pi = 3.14;
23
24     /*Alocando de modo dinamico uma estrutura
25     e armazenando o endereco da area alocada
26     em flecha*/
27     flecha = (struct pessoa*)malloc(sizeof(struct pessoa));
28
29     flecha->idade = 26;
30     flecha->peso = 78.20;
31     printf("Idade: %d\n", flecha->idade);
32     printf("Peso: %.2f", flecha->peso);
33
34     return 0;
35 }
```

2. Introdução

2.4. Structs

STRUCTS

- A linguagem C permite
criamos tipos Structs
aninhadas, veja:

```
C:\Particular Zambon\aulas\Estrutura de Dados\src\Structs\Structs
Idade da Pessoa: 26
Peso da Pessoa: 78.20
Idade do Animal: 110
O Processo retornou 0 tempo de execução : 0.170 s
Pressione uma tecla para continuar...
```

```
Structs06.c
1 //Exemplo de STRUCTS
2 //Criando Tipos em C
3 #include <stdio.h>
4 #include <stdlib.h>
5
6 typedef struct animal
7 {
8     int idade;
9 } tanimal;
10 //Podemos ter Struct dentro de outra Struct
11 //Veja abaixo a struct animal dentro da struct pessoa
12 typedef struct pessoa
13 {
14     int idade;
15     float peso;
16     tanimal animal;
17 } tpessoa;
18
19 int main(int argc, char** argv)
20 {
21     tpessoa *flecha;
22
23     flecha = (struct pessoa*)malloc(sizeof(struct pessoa));
24
25     flecha->idade = 26;
26     //Acessando a idade do animal
27     (flecha->animal).idade = 110;
28     flecha->peso = 78.20;
29     printf("Idade da Pessoa: %d\n", flecha->idade);
30     printf("Peso da Pessoa: %.2f\n", flecha->peso);
31     //Exibindo a idade do animal
32     printf("Idade do Animal: %d", (flecha->animal).idade);
33
34     return 0;
35 }
```

Referências

EDELWEISS, Nina; GALANTE, Renata. Estruturas de Dados. Porto Alegre, BOOKMAN, 2009.

HEINZLE, Roberto. Estruturas de Dados: implementações com C e Pascal. Blumenau, DIRETIVA, 2006.

TENENBAUM, Aron M. Estrutura de Dados usando C. São Paulo, Makron Books, 1995.

FORBELLONE, André Luiz Villar; EBERSPÄCHER, Henri Frederico. Lógica de Programação: a construção de algoritmos e estruturas de dados. 3. ed. São Paulo, PRENTICE HALL, 2005.

KOFFMAN, Elliot B.; WOLFGANG, Paul A. T. Objetos, abstração, estruturas de dados e projeto usando C++. Rio de Janeiro, LTC, 2008.

PEREIRA, Silvio do lago. Estruturas de dados fundamentais: conceitos e aplicações. São Paulo, Érica, 1996.

VILLAS, Marcos Viana et al. Estruturas de dados – Conceitos e técnicas de implementação. Rio de Janeiro, Campus, 1993.

VELOSO, Paulo et al. Estrutura de dados. Rio de Janeiro, Campus, 1996.

Obrigado!