

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

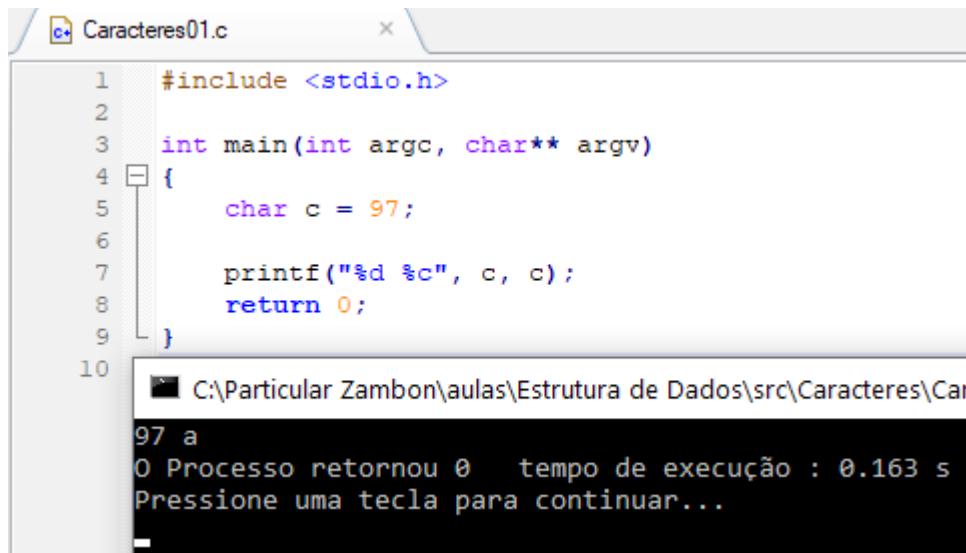
CADEIAS DE CARACTERES

- Um texto ou seu nome é representado por uma cadeia de caracteres;
- A Linguagem C não oferece o tipo caracter;
- Devemos utilizar o tipo “CHAR”;
 - O tipo CHAR possui o tamanho de 1 byte ou 8 bits.
- Esse 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.

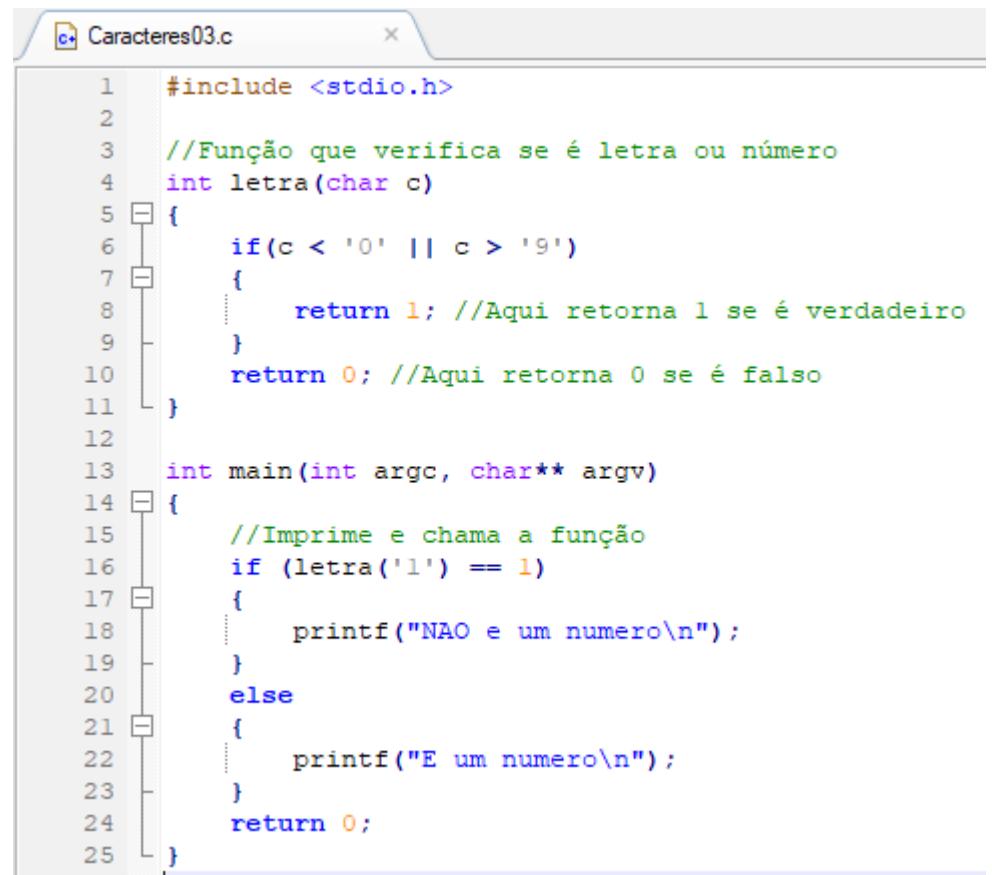
2. Introdução

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
0 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

C:\Particular Zambon\aulas\Estrutura de Dados\src\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\Caracteres04.c

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

C:\Particular Zambon\aulas\Estrutura de Dados\src\Caracteres\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 da 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 caractere por caractere
14    while(i < tam)
15    {
16        //Veja que utilizamos %c aqui
17        printf("%c", nome[i]);
18        i++;
19    }
20    return 0;
21}
```

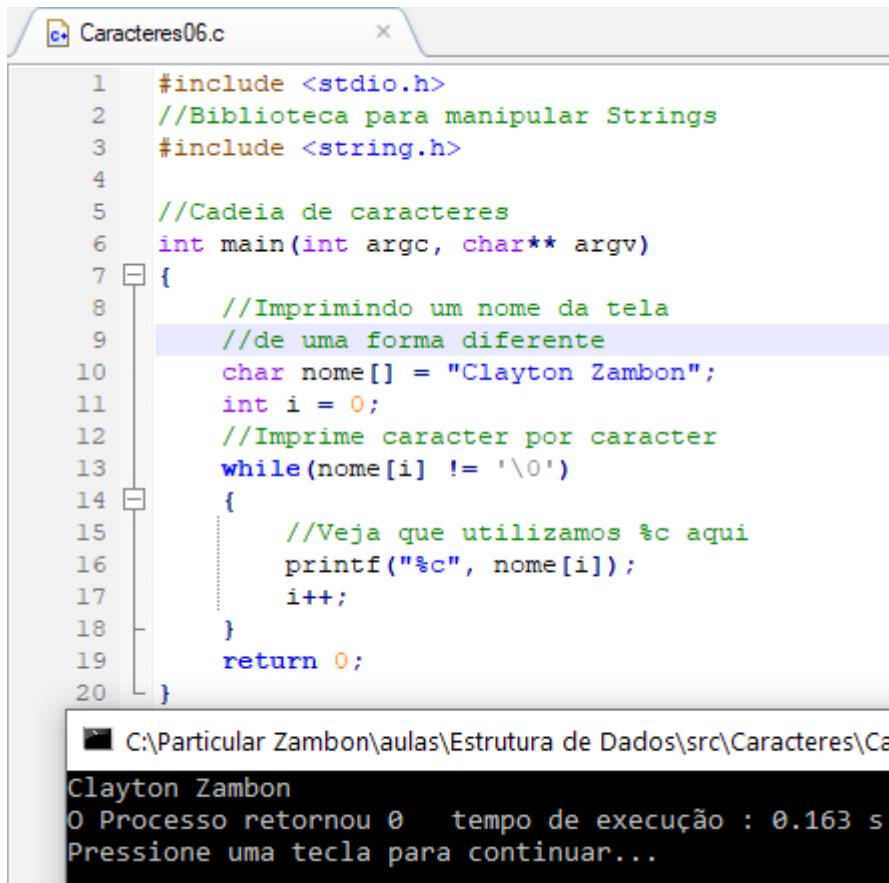
C:\Particular Zambon\aulas\Estrutura de Dados\src\Caracteres\Caracteres05.c

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

2. Introdução

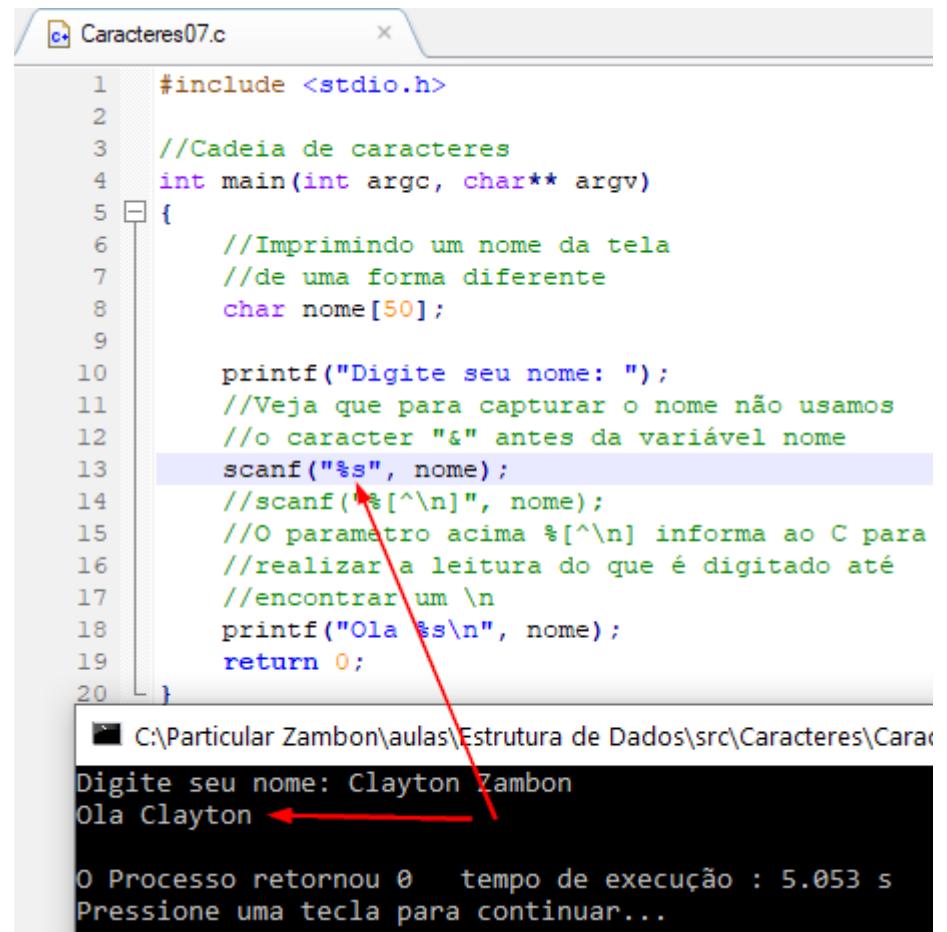
2.3. Cadeia de Caracteres

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\Caracteres06.c Clayton Zambon
O Processo retornou 0 tempo de execução : 0.163 s Pressione uma tecla para continuar...



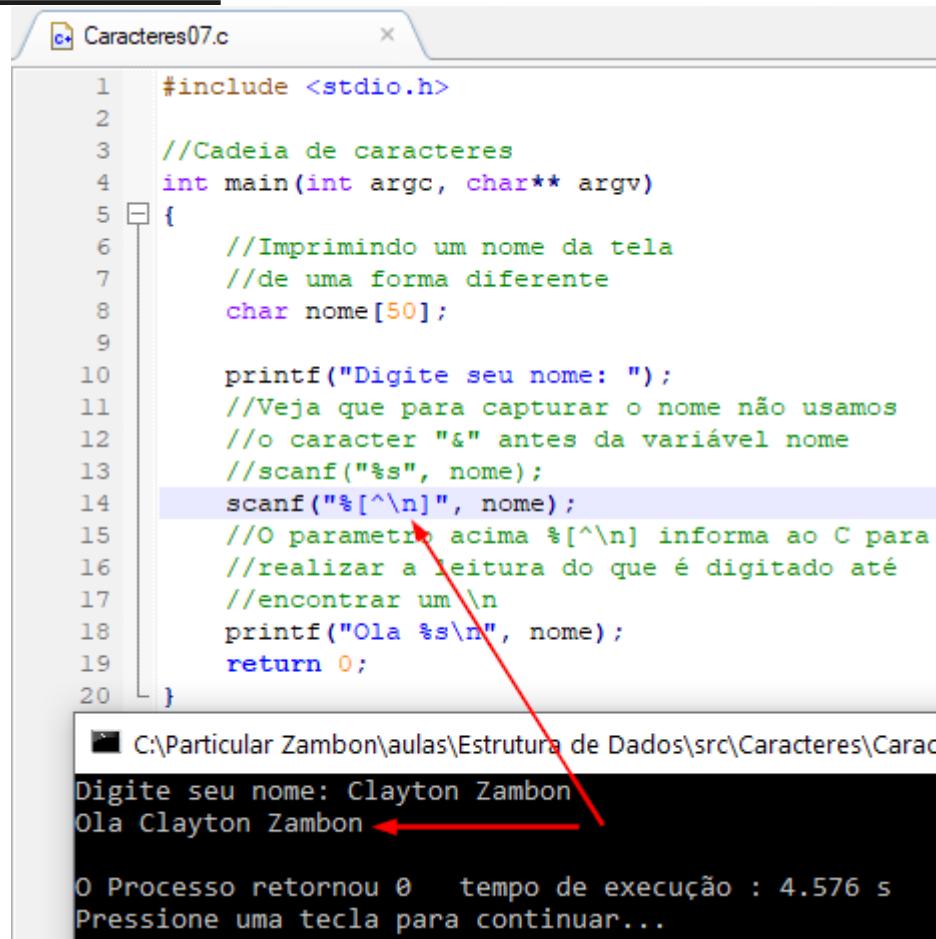
```
Caracteres07.c
1 #include <stdio.h>
2
3 //Cadeia de caracteres
4 int main(int argc, char** argv)
5 {
6     //Imprimindo um nome da tela
7     //de uma forma diferente
8     char nome[50];
9
10    printf("Digite seu nome: ");
11    //Veja que para capturar o nome não usamos
12    //o caracter "&" antes da variável nome
13    scanf("%s", nome);
14    //scanf("%[^\\n]", nome);
15    //O parâmetro 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
20 }
```

C:\Particular Zambon\aulas\Estrutura de Dados\src\Caracteres\Caracteres07.c
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



```
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("Olá %s\\n", nome);
19    return 0;
20 }
```

C:\Particular Zambon\aulas\Estrutura de Dados\src\Caracteres\Caracteres07.c

```
Digite seu nome: Clayton Zambon
Olá Clayton Zambon
```

O Processo retornou 0 tempo de execução : 4.576 s
Pressione uma tecla para continuar...

2. Introdução

2.3. Cadeia de Caracteres

CADEIAS DE CARACTERES

- O primeiro argumento

É sempre o nome do

Programa.

The screenshot shows a Windows command prompt window titled 'cmd' with the path 'C:\WINDOWS\system32\cmd.exe'. The window displays the following text:

```
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
AulaDeEstruturas
OlaDeNovo

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

A red arrow points from the text 'Caracteres08.exe' in the command line to the line '6.144 Caracteres08.exe' in the file list. Another red arrow points from the text 'AulaDeEstruturas' in the command line to the line '466 Caracteres08.c' in the file list. A third red arrow points from the text 'OlaDeNovo' in the command line to the line '543.074.177.024 bytes disponíveis' at the bottom of the file list.

Code editor window showing 'Caracteres08.c' with the following content:

```
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 }
```

A red box highlights the line 'printf("%s\n", argv[i]);' with the text 'O Primeiro argumento é sempre o nome do Programa'.

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

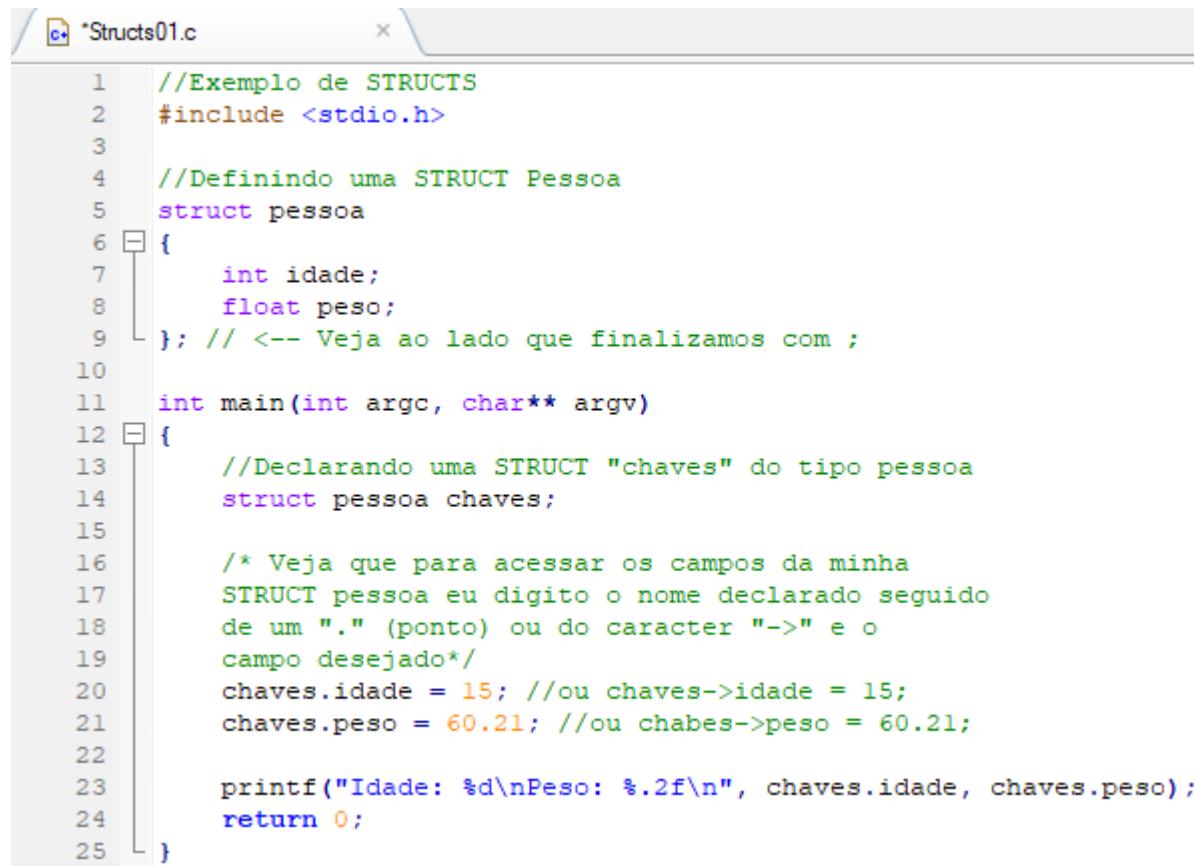
STRUCTS – (Tipos Estruturados)

- 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)



```
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 screenshot shows a code editor window titled "Structs02.c" and a terminal window showing the program's output.

Code (Structs02.c):

```
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 }
```

Terminal Output:

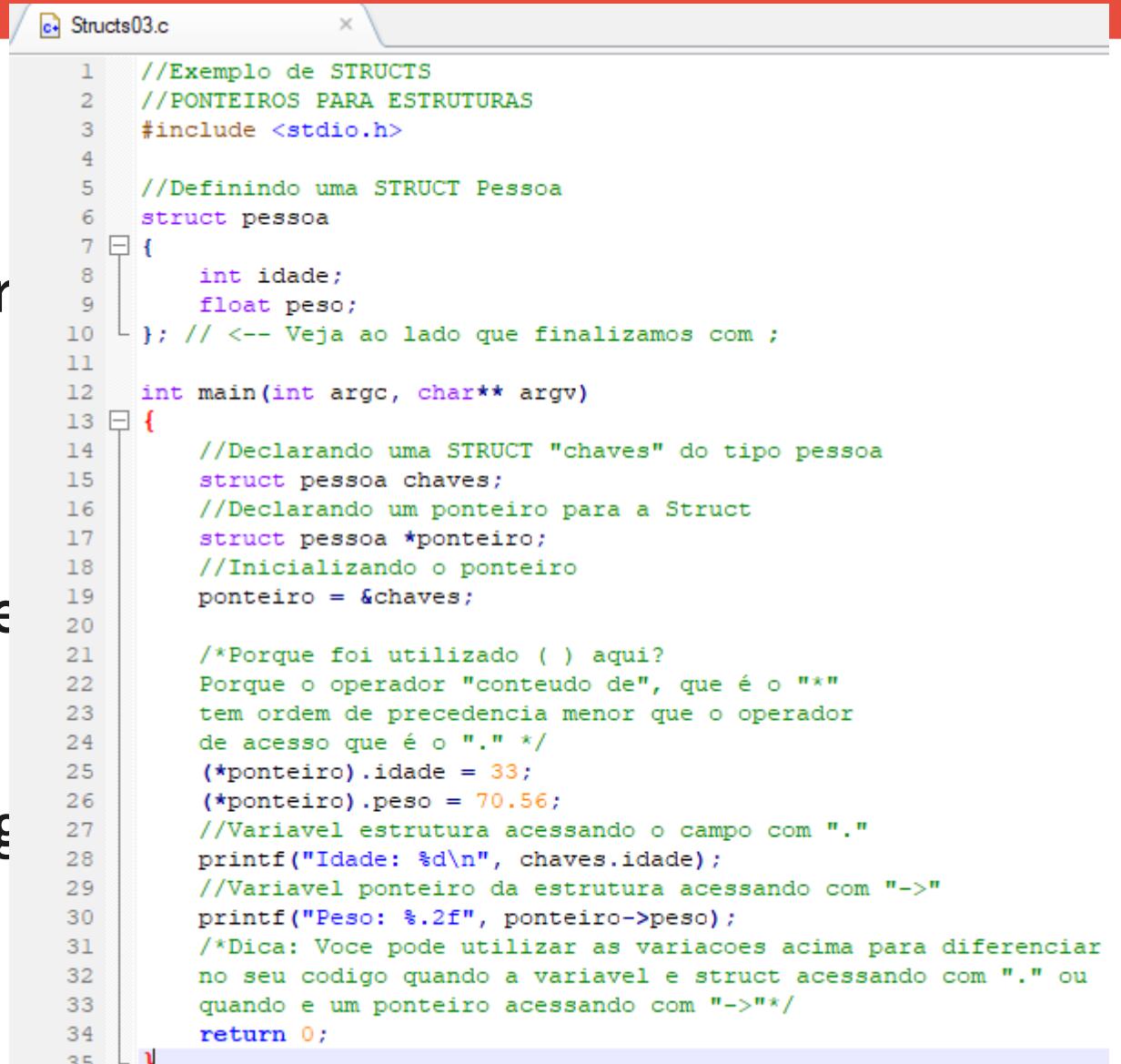
```
C:\Particular Zambon\aulas\Estrutura de Dados\src\Structs02
Informe a Idade: 15
Informe o Peso: 40.80
Idade: 15
Peso: 40.80
O Processo retornou 0 tempo de execução : 16.441 s
Pressione uma tecla para continuar...
```

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.



The image shows a screenshot of a code editor window titled "Structs03.c". The code is written in C and demonstrates the use of structures and pointers. It includes comments explaining the differences between accessing a variable directly and through a pointer. The code defines a structure "pessoa" with fields "idade" and "peso", and then uses a pointer "ponteiro" to access its members. It also shows how to print the values of these fields using both ".idade" and "->peso".

```
//Exemplo de STRUCTS
//PONTEIROS PARA ESTRUTURAS
#include <stdio.h>

//Definindo uma STRUCT Pessoa
struct pessoa
{
    int idade;
    float peso;
}; // <-- Veja ao lado que finalizamos com ;

int main(int argc, char** argv)
{
    //Declarando uma STRUCT "chaves" do tipo pessoa
    struct pessoa chaves;
    //Declarando um ponteiro para a Struct
    struct pessoa *ponteiro;
    //Inicializando o ponteiro
    ponteiro = &chaves;

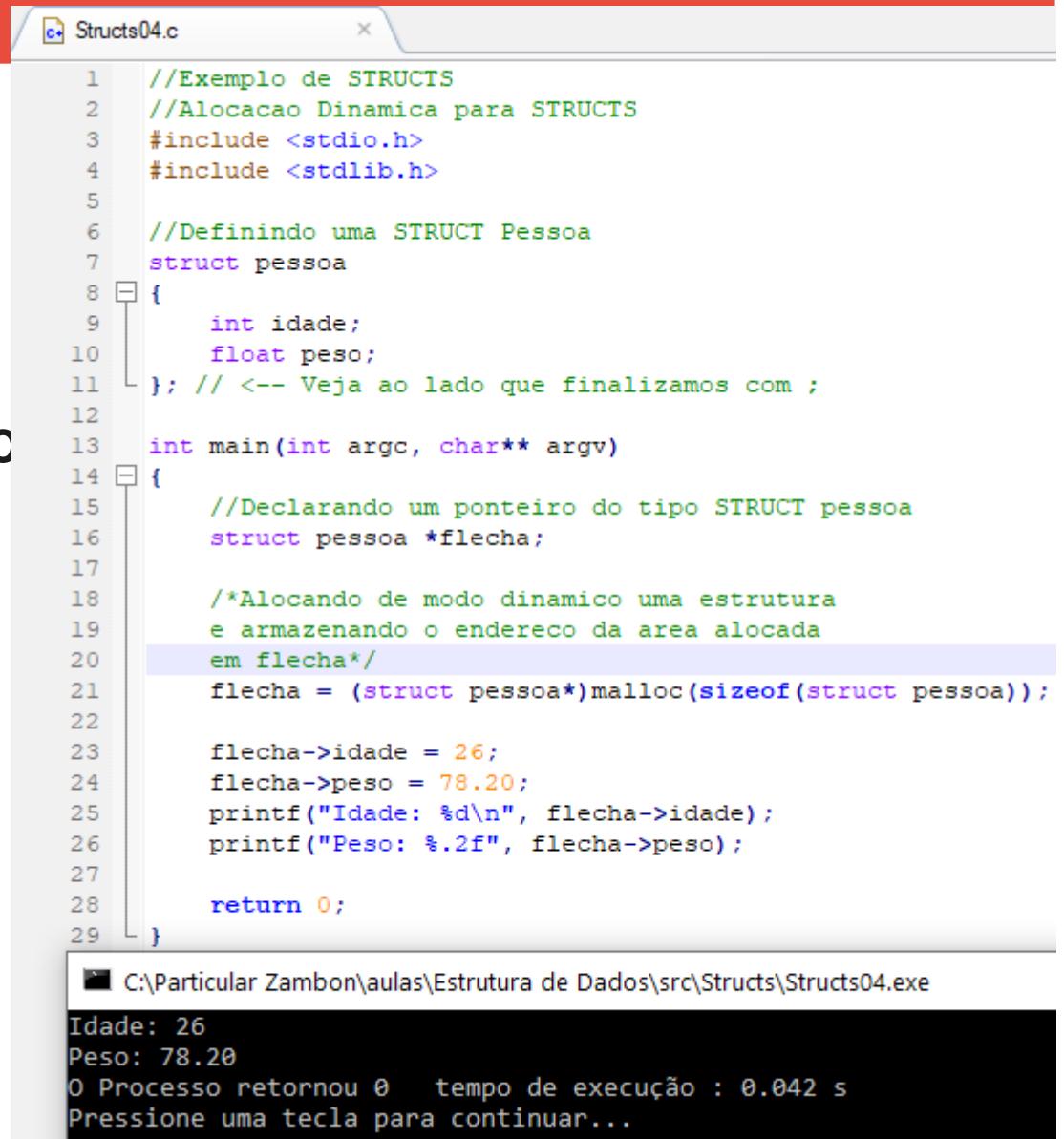
    /*Porque foi utilizado ( ) aqui?
    Porque o operador "conteudo de", que é o "*"
    tem ordem de precedencia menor que o operador
    de acesso que é o ". " */
    (*ponteiro).idade = 33;
    (*ponteiro).peso = 70.56;
    //Variavel estrutura acessando o campo com "."
    printf("Idade: %d\n", chaves.idade);
    //Variavel ponteiro da estrutura acessando com "->"
    printf("Peso: %.2f", ponteiro->peso);
    /*Dica: Voce pode utilizar as variacoes acima para diferenciar
    no seu codigo quando a variavel e struct acessando com " ." ou
    quando e um ponteiro acessando com "->"*/
    return 0;
}
```

2. Introdução

2.4. Structs

STRUCTS

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



```
//Exemplo de STRUCTS
//Alocacao Dinamica para STRUCTS
#include <stdio.h>
#include <stdlib.h>

//Definindo uma STRUCT Pessoa
struct pessoa
{
    int idade;
    float peso;
}; // <-- Veja ao lado que finalizamos com ;

int main(int argc, char** argv)
{
    //Declarando um ponteiro do tipo STRUCT pessoa
    struct pessoa *flecha;

    /*Alocando de modo dinamico uma estrutura
    e armazenando o endereco da area alocada
    em flecha*/
    flecha = (struct pessoa*)malloc(sizeof(struct pessoa));

    flecha->idade = 26;
    flecha->peso = 78.20;
    printf("Idade: %d\n", flecha->idade);
    printf("Peso: %.2f", flecha->peso);

    return 0;
}
```

C:\Particular Zambon\aulas\Estrutura de Dados\src\Structs\Structs04.exe
Idade: 26
Peso: 78.20
O Processo retornou 0 tempo de execução : 0.042 s
Pressione uma tecla para continuar...

2. Introdução

2.4. Structs

STRUCTS

- A linguagem C permite criarmos tipos, veja:

The screenshot shows a code editor window titled "Structs05.c". The code defines a struct "pessoa" with fields "idade" and "peso", and then declares a pointer "tpessoa *flecha". The code is as follows:

```
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 }
```

The line "tpessoa *flecha;" is highlighted with a light blue background and has three red arrows pointing to it from the left margin. The output window below shows the program's execution and results:

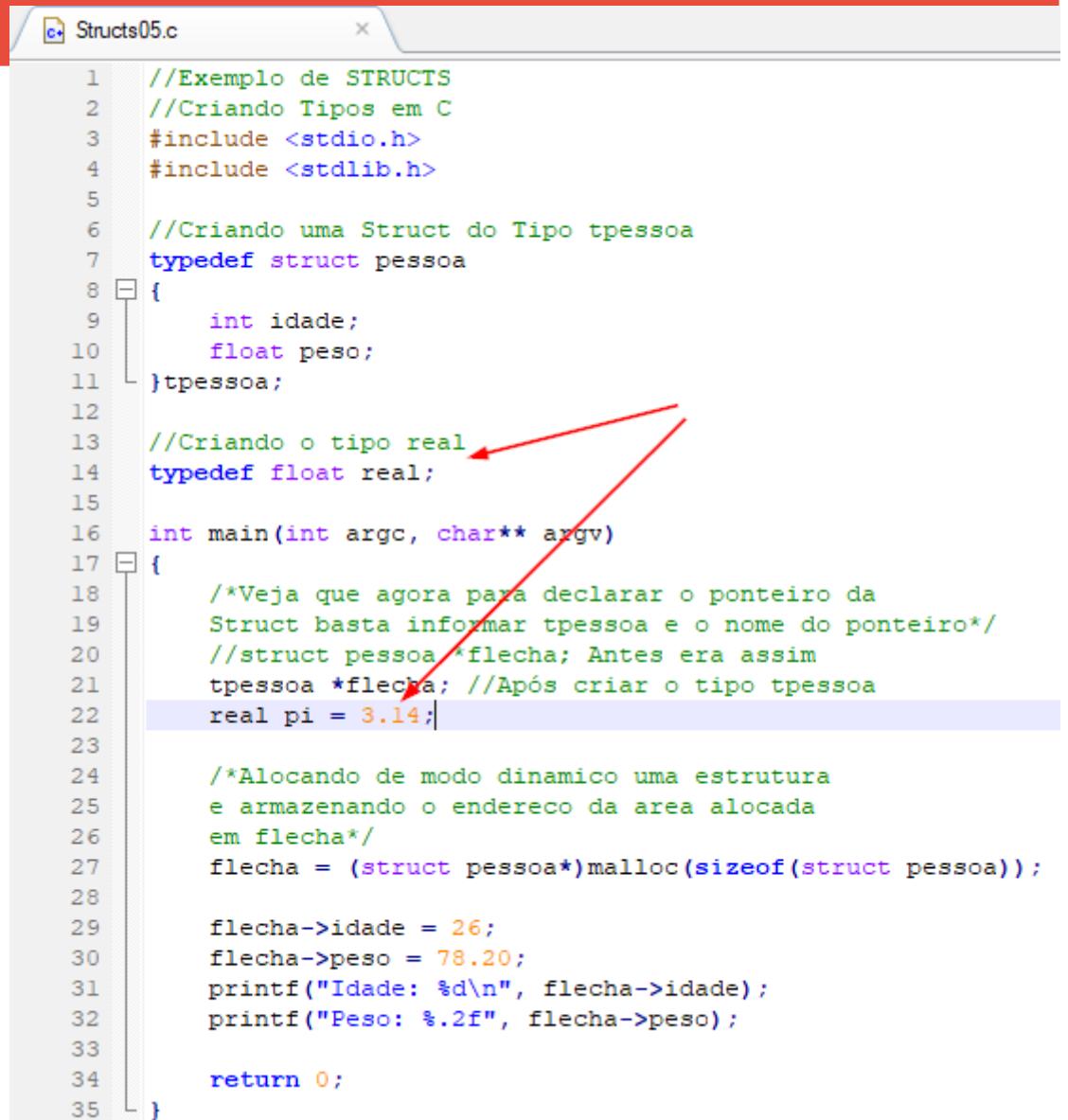
```
C:\Particular Zambon\aulas\Estrutura de Dados\src\Structs\Structs05.exe
Idade: 26
Peso: 78.20
O Processo retornou 0 tempo de execução : 0.069 s
Pressione uma tecla para continuar...
```

2. Introdução

2.4. Structs

STRUCTS

- A linguagem C permite criarmos tipos, veja:



```
//Exemplo de STRUCTS
//Criando Tipos em C
#include <stdio.h>
#include <stdlib.h>

//Criando uma Struct do Tipo tpessoa
typedef struct pessoa
{
    int idade;
    float peso;
}tpessoa;

//Criando o tipo real
typedef float real;

int main(int argc, char** argv)
{
    /*Veja que agora para declarar o ponteiro da
    Struct basta informar tpessoa e o nome do ponteiro*/
    //struct pessoa *flecha; Antes era assim
    tpessoa *flecha; //Após criar o tipo tpessoa
    real pi = 3.14;

    /*Alocando de modo dinamico uma estrutura
    e armazenando o endereço da area alocada
    em flecha*/
    flecha = (struct pessoa*)malloc(sizeof(struct pessoa));

    flecha->idade = 26;
    flecha->peso = 78.20;
    printf("Idade: %d\n", flecha->idade);
    printf("Peso: %.2f", flecha->peso);

    return 0;
}
```

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...
```

```
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 }
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 Iago. 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!