Curso: Ciência da Computação

Disciplina: Estrutura de Dados 1

Professor: Clayton Zambon

centro universitário unifacvest

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.

- 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

```c
#include <stdio.h>

int main(int argc, char** argv)
{
    char c = 97;

    printf("%d %c", c, c);
    return 0;
}
```

```
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

```c
#include <stdio.h>

//Função que verifica se é letra ou número
int letra(char c)
{
    if(c < '0' || c > '9')
    {
        return 1; //Aqui retorna 1 se é verdadeiro
    }
    return 0; //Aqui retorna 0 se é falso
}

int main(int argc, char** argv)
{
    //Imprime e chama a função
    if (letra('1') == 1)
    {
        printf("NAO e um numero\n");
    }
    else
    {
        printf("E um numero\n");
    }
    return 0;
}
```

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

```c
#include <stdio.h>

//Cadeia de caracteres
int main(int argc, char** argv)
{
    //Imprimindo um nome na tela
    char nome[] = "Clayton Zambon";
    printf("Nome Completo: %s", nome);
    return 0;
}
```

```
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

```c
#include <stdio.h>
//Biblioteca para manipular Strings
#include <string.h>

//Cadeia de caracteres
int main(int argc, char** argv)
{
    //Imprimindo um nome da tela
    char nome[] = "Clayton Zambon";
    //Função STRLEN retorna o tamanho da String
    int tam = strlen(nome);
    int i = 0;
    //Imprime caracter por caracter
    while(i < tam)
    {
        //Veja que utilizamos %c aqui
        printf("%c", nome[i]);
        i++;
    }
    return 0;
}
```

```
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

```c
#include <stdio.h>
//Biblioteca para manipular Strings
#include <string.h>

//Cadeia de caracteres
int main(int argc, char** argv)
{
    //Imprimindo um nome da tela
    //de uma forma diferente
    char nome[] = "Clayton Zambon";
    int i = 0;
    //Imprime caracter por caracter
    while(nome[i] != '\0')
    {
        //Veja que utilizamos %c aqui
        printf("%c", nome[i]);
        i++;
    }
    return 0;
}
```

```
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

```c
#include <stdio.h>

//Cadeia de caracteres
int main(int argc, char** argv)
{
    //Imprimindo um nome da tela
    //de uma forma diferente
    char nome[50];

    printf("Digite seu nome: ");
    //Veja que para capturar o nome não usamos
    //o caracter "&" antes da variável nome
    scanf("%s", nome);
    //scanf("%[^\n]", nome);
    //O parametro acima %[^\n] informa ao C para
    //realizar a leitura do que é digitado até
    //encontrar um \n
    printf("Ola %s\n", nome);
    return 0;
}
```

```
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

Caracteres07.c

```c
#include <stdio.h>

//Cadeia de caracteres
int main(int argc, char** argv)
{
    //Imprimindo um nome da tela
    //de uma forma diferente
    char nome[50];

    printf("Digite seu nome: ");
    //Veja que para capturar o nome não usamos
    //o caracter "&" antes da variável nome
    //scanf("%s", nome);
    scanf("%[^\n]", nome);
    //O parametro acima %[^\n] informa ao C para
    //realizar a leitura do que é digitado até
    //encontrar um \n
    printf("Ola %s\n", nome);
    return 0;
}
```

```
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...
```

2. Introdução
2.3. Cadeia de Caracteres

CADEIAS DE CARACTERES

- O primeiro argumento

É sempre o nome do

Programa.

Caracteres08.c

```c
#include <stdio.h>

//Um exemplo com os parametros
//da funcao MAIN "argc" e "argv"
int main(int argc, char** argv)
{
    /*O Parametro "argc" recebe o numero de argumentos
    passados para o programa quando ele e executado.
    Exemplo o comando de linha do S.O.
    O Parametro "argv[]" e um vetor de cadeia de
    caracteres que armazena os nomes passados como
    argumentos
    */
    int i;
    for(i = 0; i < argc; i++)
    {
        printf("%s\n", argv[i]);
    }
    return 0;
}
```

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

```c
//Exemplo de STRUCTS
#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;

    /* Veja que para acessar os campos da minha
    STRUCT pessoa eu digito o nome declarado seguido
    de um "." (ponto) ou do caracter "->" e o
    campo desejado*/
    chaves.idade = 15; //ou chaves->idade = 15;
    chaves.peso = 60.21; //ou chabes->peso = 60.21;

    printf("Idade: %d\nPeso: %.2f\n", chaves.idade, chaves.peso);
    return 0;
}
```

2. Introdução
2.4. Structs

STRUCTS – (Tipos Estruturados)

Structs02.c

```c
//Exemplo de STRUCTS
#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;

    printf("Informe a Idade: ");
    /*Veja que para capturar o valor digitado
    devemos usar o caracter "&" */
    scanf("%d", &chaves.idade);
    printf("Informe o Peso: ");
    scanf("%f", &chaves.peso);
    printf("Idade: %d\nPeso: %.2f", chaves.idade, chaves.peso);
    return 0;
}
```

```
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 oque é variáve

STRUCT e o que é

PONTEIRO. Veja no códig

ao lado um exemplo.

Structs03.c

```c
//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.

Structs04.c

```c
//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:

Structs05.c

```c
//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;

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

    /*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\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:

Structs05.c

```c
//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 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;
}
```

2. Introdução
2.4. Structs

STRUCTS

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

Structs06.c

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

typedef struct animal
{
    int idade;
}tanimal;
//Podemos ter Struct dentro de outra Struct
//Veja abaixo a struct animal dentro da struct pessoa
typedef struct pessoa
{
    int idade;
    float peso;
    tanimal animal;
}tpessoa;

int main(int argc, char** argv)
{
    tpessoa *flecha;

    flecha = (struct pessoa*)malloc(sizeof(struct pessoa));

    flecha->idade = 26;
    //Acessando a idade do animal
    (flecha->animal).idade = 110;
    flecha->peso = 78.20;
    printf("Idade da Pessoa: %d\n", flecha->idade);
    printf("Peso da Pessoa: %.2f\n", flecha->peso);
    //Exibindo a idade do animal
    printf("Idade do Animal: %d", (flecha->animal).idade);

    return 0;
}
```

```
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...
```

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!