

PROGRAMAÇÃO DE COMPUTADORES V - TCC- 00.323

Modulo 9: Você define como é o dado... Struct e typedef

Aura - Erick
aconci@ic.uff.br, erickr@id.uff.br

Roteiro

P1, Trabalho 5 e 6

Struct:

- Introdução;

- Criando um modelo de Estrutura;

- Referenciando elementos da estruturas;

- Atribuição de Estruturas;

- Matrizes de Estrutura;

- Exemplo ;

typedef

Exercícios

P1 (GABARITO) – 2015 / 2

1- (valor 1) O que é um fluxograma?
Desenhe e diga o significado de 4 símbolos usados nos fluxogramas.

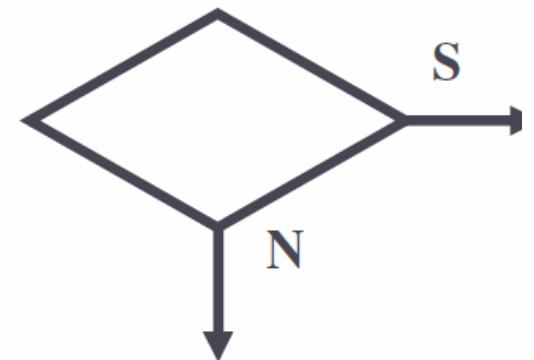
É a representação gráfica padronizada dos passos de um algoritmo. Alguns símbolos são: (valor 0,2)

INÍCIO e FIM do algoritmo (valor 0,2)

Processamento (ou ação, operação, execução) (valor 0,2)

Entrada ou saída de dados (valor 0,2)

Decisão (ou teste, controlo do laço, mudança do fluxo) (valor 0,2)



2- (valor 0,8)

Qual a diferença entre variáveis e constantes.

Dê 2 exemplos de como você pode definir constantes em C. **Variáveis** representam áreas de memória com um nome, cujo conteúdo pode ser alterado durante a execução do programa (valor 0,2).

Constante representa um **valor fixo** na memória, diferente das variáveis esse valor não vai mudar durante todo o programa, ou seja, tem o mesmo valor com validade até o término do programa (valor 0,2).

Exemplos: **#define cem 100** no cabeçalho do programa (valor 0,2), ou **const int cem = 100** (valor 0,2); (alguém pode ainda fazer uso direto do 100, mas não é uma boa opção!)



3 - (valor 1)

Sabendo que os operadores **Relacionais** ($>$, $<$, $>=$, $<=$, $==$, $!=$) relacionam dados, fornecendo como resultados : V ou F, e , que os operadores **Lógicos** ($\&\&$, $\|\|$, e $!$) Combinam resultados lógicos também resultando em V ou F; Para A , B e C constantes definidas em função de **seu número na lista de chamada**, de forma que A é o dígito das **unidades** , B o dígito das **dezenas** e C a **soma de ambos** diga se, cada uma das expressões é V ou F.

(A = B) AND (B > C) (valor 0,2)

(A != B) OR (B < C) (valor 0,2)

(A < B) AND (B > C) (valor 0,2)

(A >= B) OR (B = C) (valor 0,2)

NOT (A <= B) (valor 0,2)



4- (valor 0,2)

O que faz a Diretiva para o pré processador **#include** ?

A Diretiva **#include** permite incluir um cabeçalho de biblioteca.

As bibliotecas contêm funções pré-definidas, que podem ser utilizadas nos programas.

Exemplo:

#include <stdio.h>

inclui funções de entrada e saída



5- (valor total : 1,4)

Como você representaria em binário seu número de ordem na chamada?

(valor 0,1)

Qual o resultado da soma dele com o **decimal 64**, em binário?

(valor 0,2)

E deste resultado com o binário: 0100 0101 (valor 0,1)

Ou seja : Qual o resultado da soma destes 3 números em binários (i.e. 64, com seu número de ordem com o binário fornecido).

(valor 0,1)

Ainda qual o valor desta soma (ou seu resultado em decimal) se o número for considerada um número inteiro sem sinal (valor 0,2).

Mostre como você poderia calcular o valor da soma se ela for considerada um número inteiro **sem** sinal. (valor 0,2)

Mostre como você poderia calcular o valor da soma se ela for considerada inteiro **com** sinal. (Dica: lembre da operação complemento de dois!) (valor 0,5)



Resposta:

(vamos supor que o Aluno tem ordem 10), ele em binário fica 1010

Ele somado com 64 resulta: 0100 1010

A soma destes dois números: 0100 0101 e 0100 1010 representados em binários é

$$0100\ 0101 + 0100\ 1010 = 1000\ 1111$$

Se for **unsigned int** o valor da soma é :

$$\begin{aligned} 1000\ 1111 &= 1 \times 2^7 + 0 \times 2^6 + 0 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 \\ &= 128 + 0 \times 64 + 0 \times 32 + 0 \times 16 + 1 \times 8 + 1 \times 4 + 1 \times 2 + 1 \times 1 = 128 + 15 = 143 \end{aligned}$$

Se for **signed int** o primeiro valor dela é indicativo de sinal logo -1:

Invertendo os bits teremos: 0111 0000

Esse número tem o valor:

$$= 0 \times 2^7 + 1 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 = 64 + 32 + 16 = 112$$

Somando 1: 113

Ou usando a definição de *complemento-de-dois* (ou *two's-complement*).

$$k - 2^{8s}$$

Para esse tamanho em bits $s=1$, e $2^8 = 256$

$$256 - 143 = 113.$$

Resposta : no caso deste aluno, o valor com sinal é -113 !!!



Quinto Trabalho - para 05/02

Faça um programa que pergunte a data de nascimento de uma pessoa.

Defina ou obtenha a data atual como você quiser (e.g., quando escrever o código, assuma que uma função `anoAtual()` retorna o ano e outra função `mesAtual()` retorna o mês)!

Depois calcule a idade da pessoa, em relação à data de hoje.

Se ela tiver menos de 2 anos escreva “você é um bebê!”.

Se ela tiver entre 2 e 13 anos escreva a mensagem na tela: “Você é uma criança!”.

Se ela tiver de 13 a 19 anos escreva a mensagem na tela: “Você é um adolescente”.

Se ela tiver de 20 a 65 anos escreva na tela: “Você é um adulto”.

Caso ela tenha mais de 65 anos escreva na tela: “Você está podendo estacionar em vagas para a terceira idade!”.

Não precisa testar na idade até os dias, mas não deixe de considerar os meses.

Depois, refaça o seu código anterior usando vetores com 4 posições para tratar as datas. Ou seja, considere que em uma dada posição você armazena os dias; em uma outra posição você armazena os meses. Em uma terceira posição os anos (atual e o de nascimento da pessoa). Finalmente em uma quarta posição deste vetor, você armazenará o cálculo da idade da pessoa.

O restante do programa tem a mesma forma anterior.

Refaça mais uma vez o seu código usando agora uma função para calcular a diferença entre as datas atuais e de nascimento da pessoa. O restante do programa pode ter a mesma forma anterior.

Entregue além do código em C o seu executáveis por e-mail para o "erickr@id.uff.br".

No *subject* da e-mail por "PROG V - TRAB 5"



Sexto Trabalho

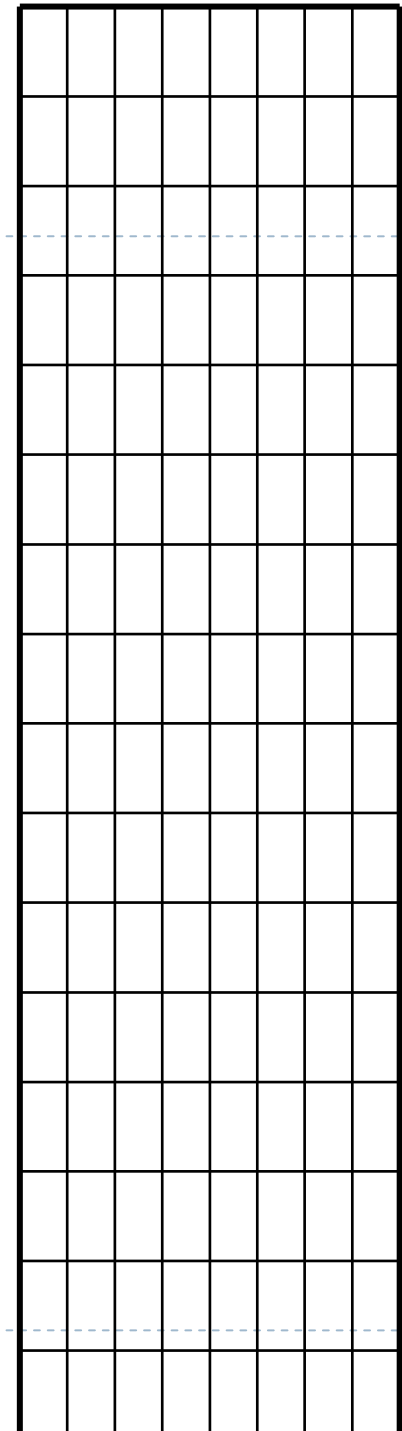
•Entrega:19 / 02 /2016

Escreva um programa que gere um arquivo que contenha uma matriz de 8 colunas por N linhas.

Cada elemento desta matriz deve conter o símbolo de “space” (o de código número 32) da tabela AISCII ou a primeira letra do seu nome em Maiúsculas.

Esses símbolos devem ser dispostos de modo a que você escreva nesta matriz um dos seus sobrenomes, na forma de uma letra sobre a outra.

Cada letra com mesma altura.





sexto trabalho – cont.

Por exemplo, você pode decidir que vai escolher para altura ter cada letra com 12 linhas.

Entre uma letra e outra deve haver uma linha inteira com apenas espaços (ou com o código de numero 32 da tabela AISCII, aparecendo na linha inteira).

Escolha também um dos seus sobrenomes que não tenha outro na sala. Por exemplo, não escolha o “Oliveira”, o “Silva” ou o “Santos”, pois há mais de um aluno com esses sobrenomes.

Explicando melhor, se você fosse sua profa., deveria gerar algo como a matriz ao lado:

```
AAAAAAAAA
A
A
A
A
A
A
A
AAAAAAAAA
```

```
AAAAAAAAA
A      A
A      A
A      A
A      A
A      A
A      A
AAAAAAAAA
```

```
A      A
AA     A
A A    A
A A    A
A  A   A
A   A  A
A    AA
A     A
```

```
AAAAAAAAA
A
A
A
A
A
A
A
AAAAAAAAA
```

```
AAAAAAA
A
A
A
A
A
A
A
AAAAAAA
```



Introdução;

Uma **estrutura** é uma coleção de variáveis referenciadas por um nome ;

Útil quando se deseja agrupar informações ;

Uma definição de estrutura forma um modelo que pode ser usado para criar variáveis de estrutura;

As variáveis que formam a estrutura são chamados membros (ou campos ou elementos) da estrutura.



STRUCT

Assim, uma *struct* é uma variável especial que contém diversas outras variáveis normalmente de tipos diferentes.

As variáveis internas contidas pela **struct** são denominadas membros da **struct**.

Sintaxe:

```
struct <identificador>
{
  <listagem dos tipos e membros>;
}
```

```
struct <identificador> <variavel>;
```



■ Criando um Modelo de Estrutura (lista postal)

```
struct end ← Especificador  
             de tipo  
{  
    char nome[30];  
    char rua[40];  
    char cidade[20];  
    char estado[3];  
    char cep[9];  
}; ← Termina com  
      ponto-e-vírgula
```

Elementos

- No trecho acima, nenhuma variável foi de fato declarada. Apenas a forma dos dados foi definida.
- Para declarar uma variável do tipo end, escrever:

```
struct end info;
```

operador “.” (ponto)

Referenciando elementos da estruturas;

Após ser declarada uma variável "x" do tipo struct "y", onde "y" é o nome da estrutura, ela é acessada pelo operador “.” (ponto)

Ou seja para se acessar os elementos usa-se a seguinte sintaxe:

x.elemento

No exemplo seguinte, é criada uma estrutura composta por 3 elementos: um inteiro, e dois vetores de caracteres de tamanhos diferentes.



Struct

```
struct Pessoa
{
    char nome[64]; // vetor de 64 caracteres para o nome
    unsigned int idade;
    char cpf[13];
};
int main()
{
    struct Pessoa aluno = {"Nina Thoni", 19, "00.000.000-00"}; // declaração variável "aluno"

    printf("Nome: %s\n", aluno .nome);
    printf("Idade: %u\n", aluno .idade);
    printf("CPF: %s\n", aluno .cpf);

    getchar(); // desnecessário, mas comumente utilizado para "segurar" a tela aberta
    return 0;
}
```



Exemplo de declaração de uma outra struct

```
struct ficha_de_aluno
```

```
{  
  char nome[50];  
  char disciplina[30];  
  float nota_prova1;  
  float nota_prova2;
```

```
};
```

```
(..... )
```

```
struct ficha_de_aluno aluno;
```

Neste exemplo criamos a **struct ficha_de_aluno**.

Depois de criar a struct precisamos criar/declarar a variável que vai utilizá-la

Para isso que criamos a variável **aluno**, que será do tipo `ficha_de_aluno`:

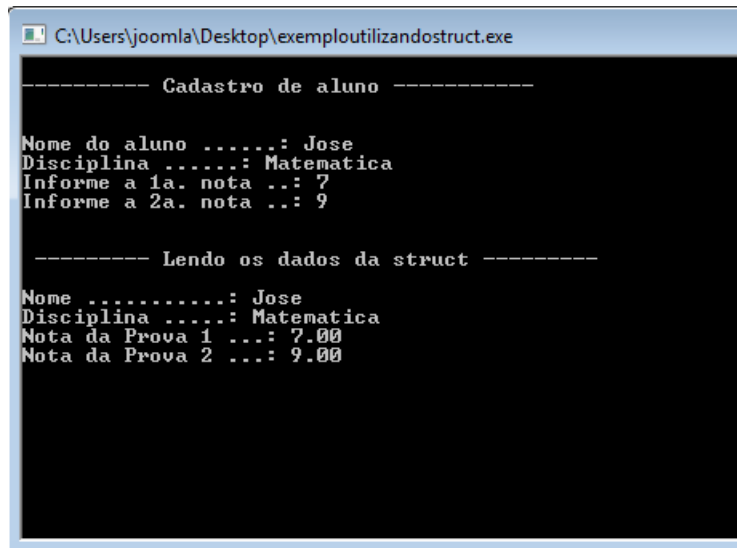
```
struct ficha_de_aluno aluno;
```



Outro exemplo de struct

```
#include <stdio.h>
#include <conio.h>
int main(void) {
    /*Criando a struct */
    struct ficha_de_aluno {
        char nome[50];
        char disciplina[30];
        float nota_prova1, nota_prova2; };
    /*Criando a variável aluno que será do tipo struct ficha_de_aluno */
    struct ficha_de_aluno aluno;
    printf("\n----- Cadastro de aluno ----- \n\n\n");
    printf("Nome do aluno .....: ");
    fgets(aluno.nome, 40, stdin);
    printf("Disciplina .....: ");
    fgets(aluno.disciplina, 40, stdin);
    printf("Informe a 1a. nota ..: ");
    scanf("%f", &aluno.nota_prova1);
    printf("Informe a 2a. nota ..: ");
    scanf("%f", &aluno.nota_prova2);
    printf("\n\n ----- Lendo os dados da struct ----- \n\n");
    printf("Nome .....: %s", aluno.nome);
    printf("Disciplina .....: %s", aluno.disciplina);
    printf("Nota da Prova 1 ...: %.2f\n" , aluno.nota_prova1);
    printf("Nota da Prova 2 ...: %.2f\n" , aluno.nota_prova2);
    getch();
    return(0); }
```

Tela de execução



```
C:\Users\joomla\Desktop\exemploutilizandostruct.exe

----- Cadastro de aluno -----

Nome do aluno .....: Jose
Disciplina .....: Matematica
Informe a 1a. nota ..: 7
Informe a 2a. nota ..: 9

----- Lendo os dados da struct -----

Nome .....: Jose
Disciplina .....: Matematica
Nota da Prova 1 ...: 7.00
Nota da Prova 2 ...: 9.00
```



Leia mais em:

<http://cursonline.no.sapo.pt/p7.htm>

<http://www.ime.usp.br/~pf/algoritmos/aulas/stru.html>

Pesquise na internet outros textos de cursos com exemplos de

Atribuição de valores a Estruturas;

Matrizes de Estrutura;

Implemente os exemplos que achar mais interessante.



Typedef

Em C e C++ podemos redefinir um tipo de dado dando-lhe um novo nome.

Essa forma de programação ajuda em **dois sentidos**:

Fica mais simples entender para que serve tal tipo de dado;

É a única forma de conseguirmos referenciar uma estrutura de dados dentro de outra struct dentro de struct).

Para redefinirmos o nome de um tipo de dado usamos o comando typedef = type definition.

Typedef faz o compilador assumir que o novo nome é um certo tipo de dado, e então, passamos a usar o novo nome da mesma forma que usaríamos o antigo.

Por exemplo, podemos definir que, ao invés de usarmos **int**, agora usaremos **inteiro** ou, ao invés de usarmos **float**, usaremos **real**.

Typedef deve sempre vir **antes de qualquer programação** (protótipo de funções, funções, função main, structs, etc.) e sua sintaxe base é:

typedef nome_antigo nome_novo;



```
#include <iostream>
#include <cstdlib>
```

```
typedef int inteiro;
typedef float real;
```

```
int main ()
{
    inteiro x = 1;
    real y = 1.5;
    printf ("%d , %f , x, y );
    system ("pause");
    return 1;
}
```



Typedef também pode ser usado para estruturas .

```
# include <stdio.h>
typedef struct Pessoa
{
    char nome[64]; // vetor de 64 chars para o nome
    unsigned int idade;
    char CPF[13];
} Pessoa;

int main()
{
    Pessoa aluno={"Felipe Santos", 26, "00.000.000-00"}; //declaração da variável "aluno "
    printf("Nome: %s\n", aluno.nome);
    printf("Idade: %u\n", aluno.idade);
    printf("CPF: %s\n", aluno.cpf);
    getchar();
    return 0;
}
```



Resumo:

As estruturas são grupos de variáveis organizadas arbitrariamente pelo programador.

Podem ser tratados como um novo um tipo de dados se utilizadas com: typedef

