

INF01202

Algoritmos e Programação

Modalidade Ensino a Distância

Linguagem C

Tópico 17: Estruturas – conceitos avançados...

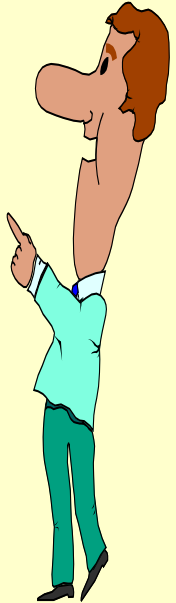
Material original desenvolvido por
Com acréscimos de material desenvolvido por

Revisado e expandido por

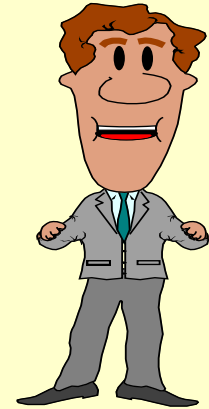
Mara Abel.
Maria Aparecida M. Souto
Magda Bercht
Maria Aparecida Castro Livi.

Até agora ...

- Trabalhamos com as diversas formas para:
 - Declarar uma estrutura em C
 - Declarar variáveis tipo estrutura
 - Inicializar uma estrutura
 - Referenciar os campos da estrutura individualmente
 - Preencher os campos com valores por leitura
 - Exibir os valores dos campos
 - Declarar e referenciar estruturas dentro de estruturas

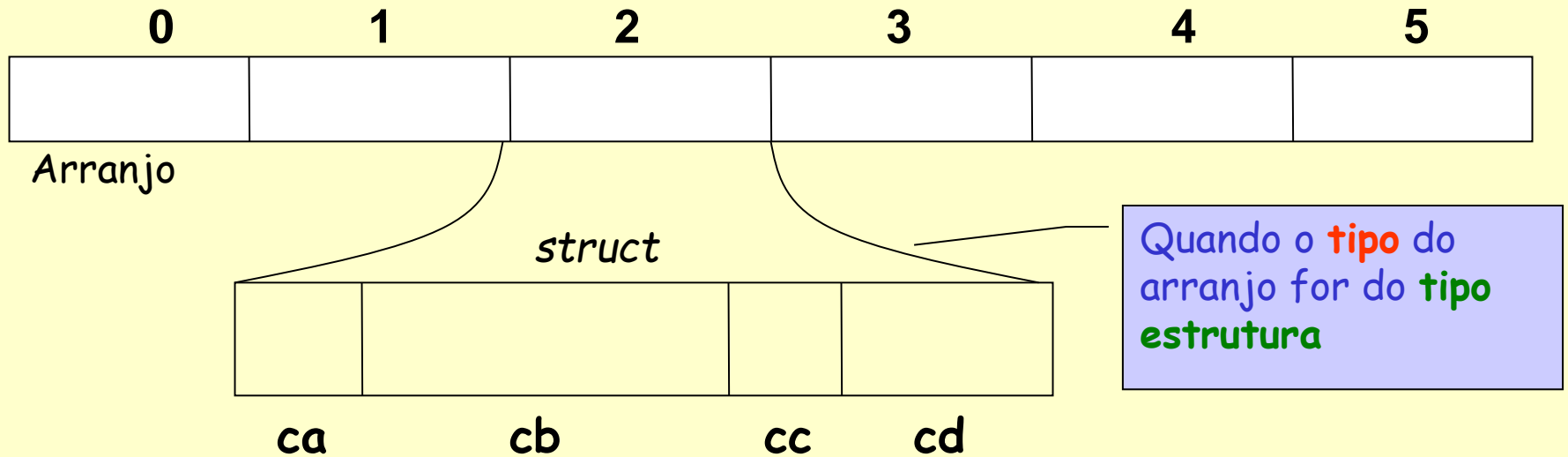


A seguir ...



- Vamos trabalhar com:
 - Arranjo de estruturas
 - Estruturas com campo tipo arranjo
 - Atribuição entre estruturas iguais
 - Passagem de estruturas entre funções:
 - Por valor
 - Por endereço
 - Os operadores seta \rightarrow e $*$ ()

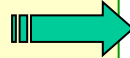
Arranjo de estruturas



- Para declarar um arranjo de estruturas, primeiro deve-se declarar a **estrutura**
- A seguir, deve-se declarar uma **variável** de arranjo daquele tipo

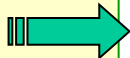
Exemplo 1: (cont.)

Declaração do tipo estrutura



```
...  
struct x  
{  
    int ca;  
    float cb;  
    char cc;  
    int cd;  
};
```

Declaração do arranjo tipo x



```
struct x arranjo[6];  
int ind;  
...
```

- O código acima cria 6 conjuntos de variáveis, organizadas conforme o tipo **x**
- Para acessar uma estrutura específica, é necessário indexar o nome da estrutura

Exemplo 1:

Declaração do tipo estrutura

```
struct x  
{  
    int ca;  
    float cb;  
    char cc;  
    int cd;  
};
```

Declaração do arranjo tipo x

```
x arranjo[6];  
int ind;
```

Referencia o campo **cb** da primeira estrutura de **arranjo**

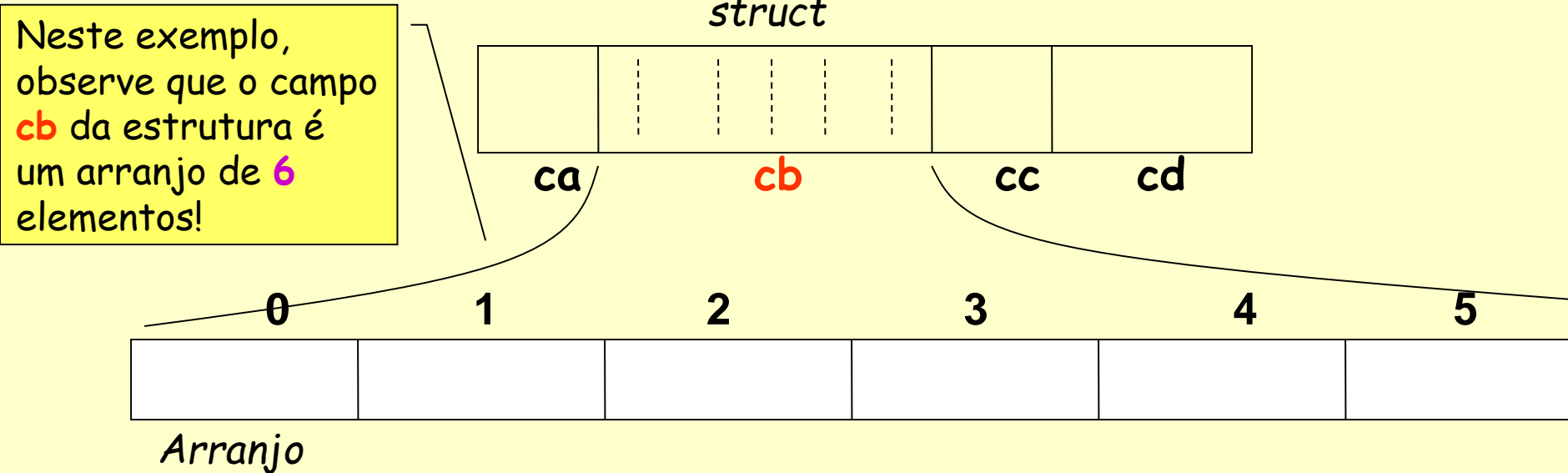
Referencia o campo **ca** da estrutura de **arranjo** indicada por **ind**

Referenciando os elementos do arranjo de estruturas:

```
scanf ("%d", &arranjo[0].cb );  
  
for (ind=0; ind < 6; ind++)  
    scanf ("%d", &arranjo[ind].ca);  
  
scanf ("%c", &arranjo[2].cc);  
scanf ("%f", &arranjo[2].cb);  
...
```

Referenciam **ca**, **cc** e **cb** da terceira estrutura de **arranjo**

Estruturas com campos arranjo



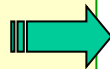
- No caso em que um dos campos da estrutura é um arranjo, então:
 - A declaração do campo é feita **normalmente** dentro da estrutura
 - Deve-se acrescentar um **índice** para referenciar cada elemento de tal arranjo

Exemplo 2:

Declaração do tipo estrutura 

```
typedef struct x
{
    int ca;
    float cb[6];
    char cc;
    int cd;
};
x var_reg;
int ind;
```

Referenciando os elementos
do campo arranjo:

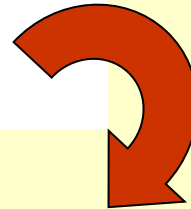


```
scanf ("%d", &var_reg.ca );

for (ind= 0; ind < 6;ind++)
    scanf ("%f", &var_reg.cb [ind]);
```


Atribuição entre estruturas iguais

Nome da variável *struct* pode ser utilizado sem os campos somente em comando de atribuição entre duas variáveis do mesmo tipo.



```
struct ficha_pessoal ficha1, ficha2;  
.....  
ficha1 = ficha2 ; // atribui todos os campos  
.....
```

Exemplo 3:

```
...
struct data {
    int dia;
    char mes[3];
    int ano;
};
struct pessoa {
    char nome[10];
    struct data dia_admissao;
    float salario;
};
{
    struct data admissao; // apenas a data
    struct pessoa funcionario;

    scanf ("%s",funcionario.nome);
    admissao.dia = 1;
    admissao.mes = 'Jul';
    admissao.ano = 2001;
    funcionario.dia_admissao=admissao;

    printf("%d %s %d",funcionario.dia_admissao.dia,
        funcionario.dia_admissao.mes,
        funcionario.dia_admissao.ano);

    ...
}
```

Exemplo 4: arranjo de estrutura¹ com campo arranjo²

```
# include <stdio.h>
# include <stdlib.h>
# define MAXALUNOS 10
```

```
int main( )
```

```
{
```

```
    typedef struct al
```

```
    {
```

```
        char nome[20];
```

```
        float nota[5]; //2
```

```
        float media;
```

```
        char conceito;
```

```
    };
```

```
    al um_aluno; // dados de 1 aluno
```

```
    al alunos[MAXALUNOS]; //1 vetor com dados de MAXALUNOS
```

```
    int ind;
```

```
    float media = 0;
```

```
    system("color 70");
```

```
    printf("\nNome do ultimo aluno:");
```

```
    fflush(stdin);
```

```
    gets(alunos[MAXALUNOS - 1].nome);
```

```
    um_aluno = alunos[MAXALUNOS - 1];
```

```
    printf("\nNome do ultimo aluno = %s\n", um_aluno.nome);
```

```
    ...
```

Exemplo que lê e mostra o nome do último aluno do vetor alunos, onde cada elemento é a estrutura al.

Calcula e mostra a média das primeiras notas de todos alunos que é o elemento 1 do arranjo nota.

A última ocorrência de **alunos** é copiada p/ a estrutura **um_aluno**, que são iguais

cont.

```

. . .
// Leitura das notas da primeira prova de todos os alunos
for (ind=0;ind<MAXALUNOS;ind++)
{
    printf("\nNota da primeira prova do aluno %d: ", ind + 1);
    scanf ( "%f", &alunos [ind].nota[0] );
}
// Media da primeira prova dos alunos
// media declarada e inicializada com 0 no início do programa
for (ind = 0; ind < MAXALUNOS; ind++)
    media = media + alunos[ind].nota[0];
media = media / MAXALUNOS;
printf("\nMedia da primeira prova: %6.2f\n", media);
system("pause");
return 0;
}

```

Estruturas e funções

Passando elementos (ou campos) da estrutura para funções: **por valor**

- Passar um elemento de uma estrutura para uma função corresponde a passar o **valor** daquele elemento individual;
- Os campos de estruturas são variáveis, assim quando passados por valor, por parâmetros de funções, são os valores que são transferidos;
- Funciona como a passagem de valores de variáveis para funções;
- Lembre: o tipo do campo deve corresponder ao tipo do argumento esperado.

Exemplo 5: passagem de campos de estruturas p/ funções por valor

```
#include <stdio.h>
#include <stdlib.h>
void func1(char);
void func2(int);
void func3(float);
void func4(char [ ]);
void func5(char);
int main ( )
```

```
{
/*observar que neste exemplo a estrutura não possui nome.*/
```

```
    struct
    {
        char x;
        int y;
        float z;
        char s[10];
    } exemplo;
```

```
    func1 (exemplo.x);           // passa o valor do caractere em x
    func2 (exemplo.y);           // passa o valor do inteiro em y
    func3 (exemplo.z);           // passa o valor do float em z
    func4 (exemplo.s);           // passa o endereço da string em s
    func5 (exemplo.s[2]);        // passa o valor do caractere em s[2]
    system("pause");
    return 0;
```

```
}
```

```
void func1(char item1)
```

```
.....
```

Passagem de campos de estruturas para funções: por endereço

- na passagem de parâmetros por endereço, envolvendo campos de estruturas: o endereço do campo é passado no parâmetro;
- o tipo do endereço deve corresponder ao tipo do argumento esperado.
- Forma geral:

```
nome_função (&identificador_struct.elemento);
```


Exemplo 6: passagem de elementos (campos) por endereço

```
#include <stdio.h>
#include <stdlib.h>
void func1(char*);
void func2(int*);
void func3(float*);
void func4(char *);
void func5(char *);
int main ( )
{
    /*observar que neste exemplo a estrutura não possui nome.*/
    struct
    {
        char x;
        int y;
        float z;
        char s[10];
    }exemplo;

    func1 (&exemplo.x); // passa o endereço do caractere em x
    func2 (&exemplo.y); // passa o endereço do inteiro em y
    func3 (&exemplo.z); // passa o endereço do float em z
    func4 (exemplo.s); // passa o endereço da string em s
    func5 (&exemplo.s[2]); // passa o endereço do caractere em s[2]
    system("pause");
    return 0;
}
void func1(char *item1)
.....
```

Passagem de campos de estruturas para funções: por valor

Relembrando:

- todas as alterações realizadas na estrutura, internamente à função são perdidas ao encerrar-se a execução da função;
- uma cópia da estrutura deve ser armazenada internamente durante a execução da função e, dependendo do tamanho da estrutura, o seu armazenamento e manuseio podem aumentar o tempo de execução de forma indesejada.

Passagem de estruturas para funções: por valor

É correto passar para uma função uma estrutura **por valor**, se os dados da estrutura não vão sofrer alteração dentro da função.

Ex.: uma função que apresenta na tela os dados de uma estrutura.

No exemplo 7, a seguir, usou-se passagem por valor também para a função que carrega valores na estrutura, o que resultou em perda do trabalho feito na função de leitura.

Exemplo 7: passagem de estruturas por valor

Os dados de um aluno são lidos e em seguida são apresentados na tela.

A leitura e a escrita dos dados são realizadas por funções utilizando parâmetros por valor.

Na escrita nada é apresentado, porque os dados do aluno foram perdidos ao encerrar-se a função de leitura `lealunos`

```
# include <stdio.h>
# include <stdlib.h>
struct al
{
    char nome[20];
    float nota[5];
    float media;
    char conceito;
};
void escrevealunos(al, int);
void lealunos(al, int);
int main( )
{
    system("color 70");
    struct al aluno;
    lealunos(aluno, 5) ;
    escrevealunos(aluno,5) ;
    system("pause");
    return 0;
}
```

Declaração global de tipo de estrutura: autorizada!!!!

Só a declaração de tipo pode ser global!

Código com problema:

Leitura e escrita de estrutura com passagem por valor.

Dados são perdidos ao encerrar-se a função de leitura.

cont.

Exemplo 7:

```
void lealunos(al aluno, int numnotas)
{
    int j;
    fflush(stdin);
    printf("\nNome: ");
    fgets(aluno.nome, sizeof(aluno.nome), stdin);
    aluno.media = 0;
    for (j = 0; j < numnotas; j++)
    {
        printf("\nNota %d: ", j + 1);
        scanf ( "%f", &aluno.nota[j]);
        aluno.media += aluno.nota[j];
    }
    aluno.media /= numnotas;
    if (aluno.media > 8.9)
        aluno.conceito = 'A';
    else
        if (aluno.media > 7.5)
            aluno.conceito = 'B';
        else
            if (aluno.media > 5.9)
                aluno.conceito = 'C';
            else
                aluno.conceito = 'D';
}
```

Código com problema:

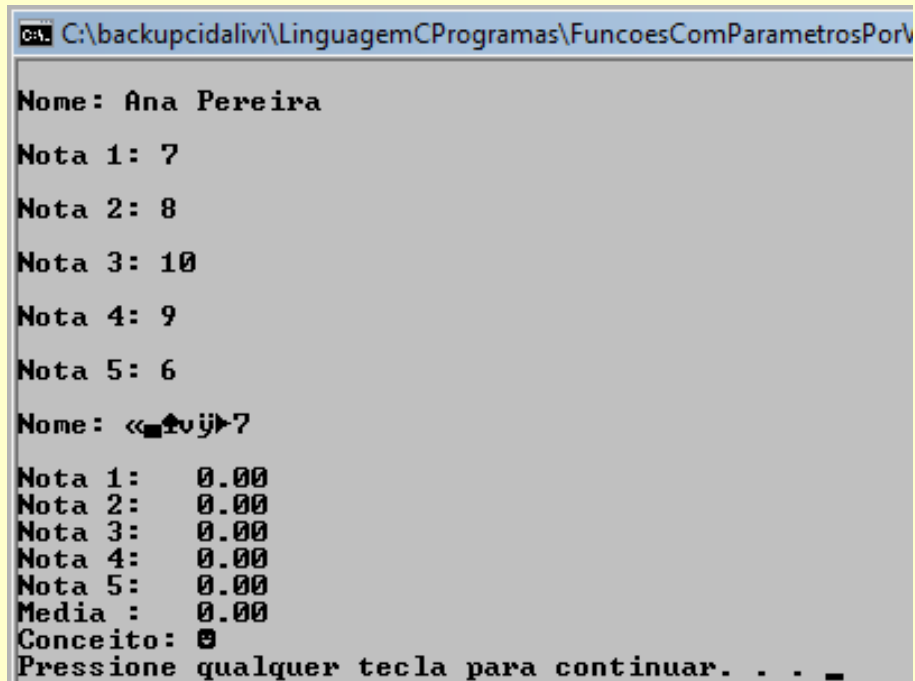
Leitura e escrita de estrutura com passagem por valor.

Dados são perdidos ao encerrar-se a função de leitura.

cont.

Exemplo 7:

```
void escrevealunos(al aluno, int numnotas)
{
    int j;
    printf("\nNome: " );
    puts(aluno.nome);
    for (j = 0; j < numnotas; j++)
        printf("\nNota %d: %6.2f", j + 1, aluno.nota[j]);
    printf("\nMedia : %6.2f", aluno.media);
    printf("\nConceito: %c\n", aluno.conceito);
}
```



```
C:\backupcidalivi\LinguagemCProgramas\FuncoesComParametrosPorV
Nome: Ana Pereira
Nota 1: 7
Nota 2: 8
Nota 3: 10
Nota 4: 9
Nota 5: 6
Nome: «»7
Nota 1: 0.00
Nota 2: 0.00
Nota 3: 0.00
Nota 4: 0.00
Nota 5: 0.00
Media : 0.00
Conceito: E
Pressione qualquer tecla para continuar. . . _
```

Código com problema:

**Leitura e escrita de estrutura com passagem por valor.
Dados são perdidos ao encerrar-se a função de leitura.**

cont.

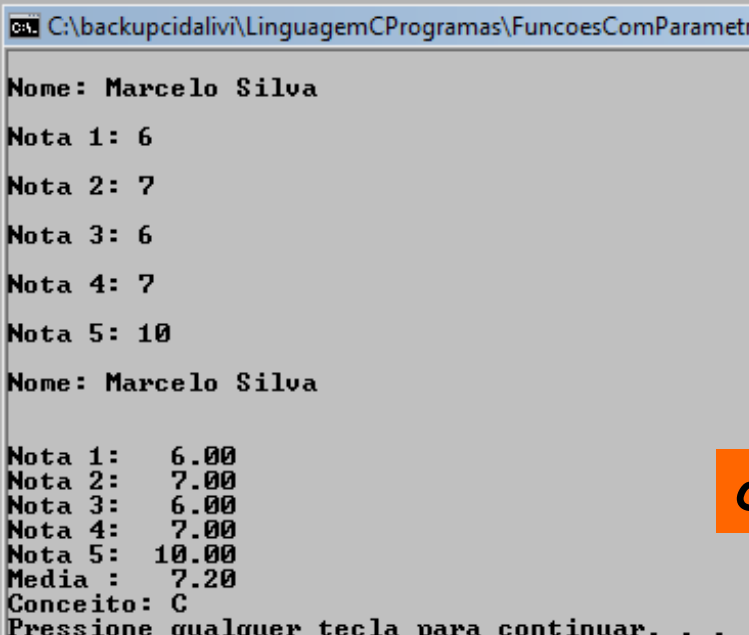
Exemplo 7:

```
void lealunos(al aluno, int numnotas)
{
    int j;
    fflush(stdin);
    printf("\nNome: ");
    fgets(aluno.nome, sizeof(aluno.nome), stdin);
    aluno.media = 0;
    for (j = 0; j < numnotas; j++)
    {
        printf("\nNota %d: ", j + 1);
        scanf ( "%f", &aluno.nota[j]);
        aluno.media += aluno.nota[j];
    }
    aluno.media /= numnotas;
    if (aluno.media > 8.9)
        aluno.conceito = 'A';
    else
        if (aluno.media > 7.5)
            aluno.conceito = 'B';
        else
            if (aluno.media > 5.9)
                aluno.conceito = 'C';
            else
                aluno.conceito = 'D';
    escrevealunos(aluno, 5);
}
```

Correção do código

Solução 1:

Se a função **lealunos** chamar a função **escrevealunos**, então os dados do aluno lido serão apresentados normalmente. Enquanto **lealunos** estiver ativa, a variável **aluno** existirá e seu conteúdo poderá ser usado sem problemas.



```
C:\backupcidalivi\LinguagemCProgramas\FuncoesComParametr
Nome: Marcelo Silva
Nota 1: 6
Nota 2: 7
Nota 3: 6
Nota 4: 7
Nota 5: 10
Nome: Marcelo Silva
Nota 1: 6.00
Nota 2: 7.00
Nota 3: 6.00
Nota 4: 7.00
Nota 5: 10.00
Media : 7.20
Conceito: C
Pressione qualquer tecla para continuar. . .
```

cont.

Passagem de estruturas para funções: por endereço

Se os dados da estrutura vão sofrer alteração dentro da função, a estrutura deverá ser passada por endereço.

Ex.: uma função que carrega dados em uma estrutura, ou uma função que altera os dados de uma estrutura.

Exemplo 7: (cont.)

```
void lealunos(al *aluno, int numnotas)
{
```

```
    int j;
    fflush(stdin);
    printf("\nNome: ");
    fgets((*)aluno).nome, sizeof((*)aluno).nome, stdin);
    (*aluno).media = 0;
    for (j = 0; j < numnotas; j++)
    {
        printf("\nNota %d: ", j + 1);
        scanf ( "%f", &(*aluno).nota[j]);
        (*aluno).media += (*aluno).nota[j];
    }
    aluno->media /= numnotas;
    if ( (*aluno).media > 8.9)
        (*aluno).conceito = 'A';
    else
        if ( (*aluno).media > 7.5)
            (*aluno).conceito = 'B';
        else
            if ( (*aluno).media > 5.9)
                (*aluno).conceito = 'C';
            else
                (*aluno).conceito = 'D';
}
```

Notação que indica o acesso por ponteiro a um campo de estrutura.

Correção do código

Solução 2:

Passar a estrutura por endereço para a função lealunos.

Neste caso, tanto o protótipo quanto a chamada de lealunos devem ser alterados.

Protótipo:

void lealunos(al*, int);

chamada:

lealunos(&aluno, 5);

Passando estruturas para funções: por endereço ...

➤ Lembre que:

- Todas as alterações realizadas na estrutura, internamente à função, **permanecem** acessáveis mesmo após o término da função

➤ Interessante: o ponteiro da estrutura que é passado para a função ocupa pouco espaço!

Ponteiros para estruturas: notações (*) e ->

Notação 1: (*)

Na função recém vista, `lealunos`, `aluno` é um ponteiro para uma estrutura.

O acesso a campos da estrutura foi indicado como:

`&(*aluno).nota[j]` , `(*aluno).media` ,
`(*aluno).conceito` .

O parênteses que cerca `*aluno` é necessário porque o `.(ponto)` tem prioridade maior que o `*`. Sem os parênteses, a leitura da referência ficaria incorreta.

Notação 2: ->

Notação mais frequente para indicar o acesso a um campo de uma estrutura por meio de um ponteiro para essa estrutura.

Escrever `aluno->media` é o mesmo que escrever `(*aluno).media`.

Exemplo 7:

```
void lealunos(al *aluno, int numnotas)
{
    int j;
    fflush(stdin);
    printf("\nNome: ");
    fgets(aluno->nome, sizeof(aluno->nome), stdin);
    aluno->media = 0;
    for (j = 0; j < numnotas; j++)
    {
        printf("\nNota %d: ", j + 1);
        scanf ("%f", &aluno->nota[j]);
        aluno->media += aluno->nota[j];
    }
    aluno->media /= numnotas;
    if (aluno->media > 8.9)
        aluno->conceito = 'A';
    else
        if (aluno->media > 7.5)
            aluno->conceito = 'B';
        else
            if (aluno->media > 5.9)
                aluno->conceito = 'C';
            else
                aluno->conceito = 'D';
}
```

Versão da função
lealunos com
notação ->

Arranjo de estruturas com campo arranjo

Exemplo 8: Os dados de vários alunos são lidos e armazenados em um arranjo e após são todos apresentados na tela.

A leitura e a escrita do arranjo com dados dos alunos são realizadas por funções utilizando parâmetros por endereço.

Na sequência são apresentadas duas soluções
(e suas implementações).

Observar que a passagem do parâmetro estrutura ocorre da mesma forma nas duas soluções.

Exemplo 8: Arranjo de estruturas com campo arranjo

Solução 1:

Laços de varredura do arranjo na função *main*.

Demais funções recebem uma ocorrência do arranjo a cada vez.

Exemplo 8 solução 1:

//Leitura e escrita de arranjo de estrutura com passagem por endereço
//laços de varredura do arranjo na main

```
# include <stdio.h>
```

```
# include <stdlib.h>
```

```
#define MAXALU 5
```

```
#define MAXNOTAS 5
```

```
struct al
```

```
{
```

```
    char nome[20];
```

```
    float nota[MAXNOTAS];
```

```
    float media;
```

```
    char conceito;
```

```
};
```

```
void lealunos(al *, int);
```

```
void escrevealunos(struct al *, int);
```

```
int main( )
```

```
{
```

```
    int i;
```

```
    struct al aluno, turma[MAXALU];
```

```
    system("color 70");
```

```
    for (i = 0; i < MAXALU; i++)
```

```
        lealunos(&turma[i], MAXNOTAS);
```

```
    for (i = 0; i < MAXALU; i++)
```

```
        escrevealunos(&turma[i], MAXNOTAS) ;
```

```
    system("pause");
```

```
    return 0;
```

```
}
```

cont.

Exemplo 8 solução 1:

```
void lealunos(al *aluno, int numnotas)
```

```
{
    int j;
    fflush(stdin);
    printf("\nNome: ");
    fgets(aluno->nome, sizeof(aluno -> nome), stdin);
    aluno ->media = 0;
    for (j = 0; j < numnotas; j++)
    {
        printf("\nNota %d: ", j + 1);
        scanf ( "%f", &(aluno ->nota[j]));
        aluno->media += aluno ->nota[j];
    }
    aluno ->media /= numnotas;
    if (aluno ->media > 8.9)
        aluno ->conceito = 'A';
    else
        if (aluno ->media > 7.5)
            aluno ->conceito = 'B';
        else
            if (aluno ->media > 5.9)
                aluno->conceito = 'C';
            else
                aluno ->conceito = 'D';
}
```

cont.

Exemplo 8 solução 1:

```
void escrevealunos( al *pal, int numnotas)
{
    int j;
    printf("\nNome: " );
    puts(pal ->nome);
    for (j = 0; j < numnotas; j++)
        printf("\nNota %d: %6.2f", j + 1, pal ->nota[j]);
    printf("\nMedia : %6.2f", pal ->media);
    printf("\nConceito: %c\n", pal ->conceito);
}
```

Solução 2:

**Arranjo passado como
parâmetro para as
funções.**

**Laços de varredura do
arranjo interno às
funções.**

Versão 1:

com operador - >

Exemplo 8 solução 2 versão 1:

*//Leitura e escrita de arranjo de estrutura com passagem por endereço
//laços de varredura do arranjo internos às funções (não na main)*

```
# include <stdio.h>
# include <stdlib.h>
#define MAXALU 5
#define MAXNOTAS 5
struct al
{
    char nome[20];
    float nota[MAXNOTAS];
    float media;
    char conceito;
};
void lealunos(al *, int, int);
void escrevealunos(struct al *, int, int);
int main( )
{
    int i;
    struct al aluno, turma[MAXALU];
    system("color 70");
    lealunos(turma, MAXALU, MAXNOTAS) ;
    escrevealunos(turma, MAXALU, MAXNOTAS) ;
    system("pause");
    return 0;
}
```

cont.

Exemplo 8 solução 2 versão 1:

```
void lealunos(al *aluno, int numalu, int numnotas)
{
    int i, j;
    for (i = 0; i < numalu; i++)
    {
        fflush(stdin);
        printf("\nNome: ");
        fgets((aluno + i)->nome, sizeof((aluno + i)->nome), stdin);
        (aluno + i)->media = 0;
        for (j = 0; j < numnotas; j++)
        {
            printf("\nNota %d: ", j + 1);
            scanf ( "%f", &((aluno + i)->nota[j]));
            (aluno + i)->media += (aluno + i)->nota[j];
        }
        (aluno + i)->media /= numnotas;
        if ((aluno + i)->media > 8.9)
            (aluno + i)->conceito = 'A';
        else
            if ((aluno + i)->media > 7.5)
                (aluno + i)->conceito = 'B';
            else
                if ((aluno + i)->media > 5.9)
                    (aluno + i)->conceito = 'C';
                else
                    (aluno + i)->conceito = 'D';
    }
}
```

cont.

Exemplo 8 solução 2 versão 1:

```
void escrevealunos( al *pal, int numalu, int numnotas)
{
    int i, j;
    for (i = 0; i < numalu; i++)
    {
        printf("\nNome: " );
        puts((pal + i)->nome);
        for (j = 0; j < numnotas; j++)
            printf("\nNota %d: %6.2f", j + 1, (pal + i)->nota[j]);
        printf("\nMedia : %6.2f", (pal + i)->media);
        printf("\nConceito: %c\n", (pal + i)->conceito);
    }
}
```

Versão 2:

com notação de arranjo

Exemplo 8 solução 2 versão 2:

```
//Leitura dos dados de varios alunos
//e escrita posterior desses dados
#include <stdio.h>
#include <stdlib.h>
#define MAXALUNOS 2
#define MAXNOTAS 5
struct al
{
    char nome[20];
    float nota[MAXNOTAS];
    float media;
    char conceito;
};

void lealunos(al *, int, int);
void escrevealunos(struct al *, int, int);
int main( )
{
    system("color 70");
    struct al aluno, turma[MAXALUNOS];
    lealunos(turma, MAXNOTAS, MAXALUNOS) ;
    escrevealunos(turma, MAXNOTAS, MAXALUNOS) ;
    system("pause");
    return 0;
}
```

cont.

Exemplo 8 solução 2 versão 2:

```
void lealunos(al *aluno, int numnotas, int maxalunos)
{
    int j, i;
    for (i = 0; i < maxalunos; i++)
    {
        fflush(stdin);
        printf("\nNome: ");
        gets(aluno[i].nome);
        aluno[i].media = 0;
        for (j = 0; j < numnotas; j++)
        {
            printf("\nNota %d: ", j + 1);
            scanf ( "%f", &(aluno[i].nota[j]));
            aluno[i].media += aluno[i].nota[j];
        }
        aluno[i].media /= numnotas;
        if (aluno[i].media > 8.9)
            aluno[i].conceito = 'A';
        else
            if (aluno[i].media > 7.5)
                aluno[i].conceito = 'B';
            else
                if (aluno[i].media > 5.9)
                    aluno[i].conceito = 'C';
                else
                    aluno[i].conceito = 'D';
    }
}
```

cont.

Exemplo 8 solução 2 versão 2:

```
void escrevealunos( al *pal, int numnotas, int maxalunos)
{
    int j, i;
    for (i = 0; i < maxalunos; i++)
    {
        printf("\nNome: " );
        puts(pal[i].nome);
        for (j = 0; j < numnotas; j++)
            printf("\nNota %d: %6.2f",  j + 1, pal[i].nota[j]);
        printf("\nMedia : %6.2f",    pal[i].media);
        printf("\nConceito: %c\n",    pal[i].conceito);
    }
}
```

Arranjo de estruturas com campo arranjo

Exemplo 9: Os dados de vários alunos são lidos e armazenados em um arranjo e após são apresentados na tela apenas os dados do primeiro aluno. A leitura e a escrita do arranjo com dados dos alunos são realizadas por funções utilizando parâmetros por endereço.

**Apenas uma ocorrência do arranjo
é processada.**

Uso da notação *()

Exemplo 9:

```
//Leitura de arranjo de estrutura com passagem por endereco
//escrita so dos dados do primeiro aluno
#include <stdio.h>
#include <stdlib.h>
#define MAXALUNOS 2
#define MAXNOTAS 5
struct al
{
    char nome[20];
    float nota[MAXNOTAS];
    float media;
    char conceito;
};
void lealunos(al *, int, int);
void escrevealunos(struct al *, int, int);
int main( )
{
    system("color 70");
    struct al aluno, turma[MAXALUNOS], aux;
    lealunos(turma, MAXNOTAS, MAXALUNOS) ;
    aux = turma[0];
    printf("\n%6.2f\n", aux.media);
    escrevealunos(&aux, MAXNOTAS, MAXALUNOS) ;
    system("pause");
    return 0;
}
```

cont.

Exemplo 9:

```
void lealunos(al *aluno, int numnotas, int maxalunos)
{
    int j, i;
    for (i = 0; i < maxalunos; i++)
    {
        fflush(stdin);
        printf("\nNome: ");
        fgets((*aluno + i).nome, sizeof ((*aluno + i).nome), stdin);
        (*aluno + i).media = 0;
        for (j = 0; j < numnotas; j++)
        {
            printf("\nNota %d: ", j + 1);
            scanf ( "%f", &((*aluno + i)).nota[j]);
            (*aluno + i).media += (*aluno + i).nota[j];
        }
        (*aluno + i).media /= numnotas;
        if ((*aluno + i).media > 8.9)
            (*aluno + i).conceito = 'A';
        else
            if ((*aluno + i).media > 7.5)
                (*aluno + i).conceito = 'B';
            else
                if ((*aluno + i).media > 5.9)
                    (*aluno + i).conceito = 'C';
                else
                    (*aluno + i).conceito = 'D';
    }
}
```

cont.

Exemplo 9:

```
void escrevealunos( al *pal, int numnotas, int maxalunos)
{
    int j, i;
    printf("\nNome: " );
    puts((*pal).nome);
    for (j = 0; j < numnotas; j++)
        printf("\nNota %d: %6.2f", j + 1, (*pal).nota[j]);
    printf("\nMedia : %6.2f",    (*pal).media);
    printf("\nConceito: %c\n",    (*pal).conceito);
}
```