UNIVERSIDADE DE BRASÍLIA

Introdução à Ciência da Computação

Disciplina: 113913

Prof: Luiz Augusto F. Laranjeira

luiz.laranjeira@gmail.com

Universidade de Brasília – UnB

Campus Gama

14. ESTRUTURAS

5. Estruturas

- Uma estrutura é um tipo de dado composto formado por um conjunto de variáveis de tipos diferentes englobadas dentro de um mesmo nome.
- Estruturas agregam variáveis de tipos diferentes. Fazendo-se diferentes combinações pode-se gerar novos tipos de dados.
- Pode-se ter uma estrutura como membro de outra estrutura.
- Pode-se definir vetores de estruturas.
- Dada uma estrutura pode-se definir um ponteiro para ela.

5. Estruturas

```
struct  conta_t
{
    char nome[50];
    int telefone;
    float saldo;
     int  nconta;
};
```

- ✓ Definido o tipo, devemos agora declarar a variável que será deste tipo:

```
struct conta_t conta1;
```

5. Estruturas

```
struct conta_t
{
    char nome[50];
    int telefone;
    float saldo;
} conta1, conta2;
```

Equivalente a:

```
struct conta_t
{
    char nome[50];
    int telefone;
    float saldo;
} ;

struct  conta_t  conta1, conta2;
```

5. Estruturas

- **Utilização do tipo**

✓ Podemos fazer atribuição de estruturas do mesmo tipo conta2 = conta1, e os valores dos campos correspondentes serão idênticos.

✓ Para contar o número de caracteres de nome dentro da estrutura conta, podemos fazer:

```
for (i=0,conta1.nome[i],++i) ;
printf ("o nome tem -> %d letras \n",i);
```

5. Estruturas

- **Utilização de vetores de estruturas**

```
struct conta_t conta[100];

conta[1].telefone=2212324;
conta[1].nome=“João Carlos”;
conta[1].saldo=1245.89;

conta[5].telefone=2212888;
conta[5].nome=“Maria ”;
conta[5].saldo=6908.79;
```

5. Estruturas e Ponteiros

- **Notação de vetor ou de (aritmética de) ponteiros**

```
struct conta_t conta[100];
struct  conta_t  *p;

p = &conta[1];                  //  (p+4) = &conta[5];

(*p).telefone=2212324;          //   p->telefone = 2212324;
p[0].nome="João Carlos";        //   p->nome = "João Carlos";
p[0].saldo=1245.89;             //   p->saldo = 1245.89;

(*(p+4)).telefone=2212888;      //  (p+4)->telefone=2212888;
p[4]. nome="Maria ";            //  (p+4)->nome="Maria";
p[4].saldo=6908.79;             //  (p+4)->saldo=6908.79;
```

5. Estruturas e a Declaração de Tipo typedef

- **Utilização da declaração de tipo `typedef`**

```
typedef struct{
        char Primeiro[15];
        char Meio[15];
        char Sobrenome[15];
} NomeCompleto;

typedef struct{
        NomeCompleto nome;
        int  idade;
} Pessoa;
```

Pessoa diretor; // a variável diretor é do tipo Pessoa

Pessoa *p_pessoa; // ponteiro para Pessoa

FUNÇÕES DE LEITURA E ESCRITA DE CARACTERES

getchar()

Le um caracter do teclado e o ecoa na tela, avançando o cursor.

Só executa depois que o operador tecla <enter>.

putchar(c)

Escreve um caracter na tela e avança o cursor.

Exercício 5

Escreva um programa que faça o cadastro dos funcionários de uma empresa com as seguintes informações para cada funcionário: nome, sexo, salário, código (matrícula), endereço e cargo.

Defina um vetor de estruturas com 100 elementos. O programa deve ler o número de empregados que se deseja cadastrar (no máximo 100) e, em seguida, ler os dados de cada funcionário.

Finalmente o programa deverá escrever na tela os dados de cada funcionário da empresa.

Neste programa você deverá definir um tipo composto (estrutura) com a diretiva **typedef** e utilizar um vetor de estruturas.

Exercício 5 – Solução A

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

int main()
{
    // Declarações de variáveis
    typedef struct {
            char nome[51];
            char sexo;
            int matricula;
            int salario;
            char endereco[81];
            char cargo[21];
    } funcionario_t;

    int  num, i, j, mat;
    char c;
    funcionario_t funcionarios[100];
```

```
// Lendo o número de funcionários a serem
// cadastrados
do {
        printf("\nEntre o numero de funcionarios a
                                serem cadastrados: ");
        scanf("%d", &num);
        fflush(stdin);
 } while (num <= 0   ||   num > 100);
```

Exercício 5 – Solução A (cont.)

```
// Lendo os dados de cada funcionário
for (i=0; i < num; i++)
{
    // Pulando linhas para clareza na tela
    printf("\n\n");

    // Lendo o nome de um funcionário
    printf("Nome [%d] (max 50 chars): ", i+1);
    j = 0;
    do {
        c = getchar();
        if (c == '\n') c = '\0';
        if ((c>='a' && c<='z') || (c>='A' && c<='Z') ||
                        (c==‘ ‘)  ||  (c=='\0'))
        {
            funcionarios[i].nome[j] = c;
            if (c!='\0') j++;
        }
    } while ((j < 49) && ((c!= '\0') || (c=='\0‘  &&
                strlen(funcionarios[i].nome)==0)));

    if (c != '\0') funcionarios[i].nome[j] = '\0';
}
```

```
// Lendo o sexo de um funcionário
do {
        printf("Sexo [%d] ('f','F', 'm', 'M'): ", i+1);
        c = getchar();
        fflush(stdin);
        funcionarios[i].sexo = c;
} while (c!='f' && c!='F' && c!='m' && c!='M');

// Lendo a matrícula de um funcionário
do {
        printf("Matricula [%d]: ", i+1);
        scanf("%d", &mat);
        funcionarios[i].matricula = mat;
        fflush(stdin);
} while (mat < 0);
```

Exercício 5 – Solução A (cont.)

```
// Lendo o endereço de um funcionário
printf("Endereco [%d] (max 80 chars): ", i+1);
j = 0;
do {
      c = getchar();
      if (c == '\n') c = '\0';
      funcionarios[i].endereco[j] = c;
      if (c!='\0') j++;
} while ((j < 79) && ((c!= '\0') || (c=='\0' &&
                  strlen(funcionarios[i].endereco)==0)));
if (c != '\0') funcionarios[i].endereco[j] = '\0';

// Lendo o cargo de um funcionário
printf("Cargo [%d] (max 20 chars): ", i+1);
j = 0;
do {
      c = getchar();
      if (c == '\n') c = '\0';
     if ((c>='a' && c<='z') || (c>='A' && c<='Z') || (c=='\0'))  {
            funcionarios[i].cargo[j] = c;
            if (c!='\0') j++;
      }
} while ((j < 19) && ((c != '\0') || (c == '\0' &&
                    strlen(funcionarios[i].cargo)==0)));
      if (c != '\0') funcionarios[i].cargo[j] = '\0';
   }     // for loop que le os dados dos funcionarios

  // Escrevendo o número de funcionários cadastrados
   printf("FORAM CADASTRADOS %d FUNCIONARIOS\n",
                                                num);

   // Escrevendo os dados de cada funcionário
   for (i=0; i < num; i++)
   {
         printf("\n\nFUNCIONARIO [%d]\n", i+1);
         printf("     Nome  : %s\n", funcionarios[i].nome);
         printf("     Sexo  : %c\n", funcionarios[i].sexo);
         printf("     Matr  : %d\n", funcionarios[i].matricula);
         printf("     End   : %s\n", funcionarios[i].endereco);
         printf("     Cargo : %s\n", funcionarios[i].cargo);
   }

   // Colocando uma pausa na tela
   system("pause");
   return(0);

}  // end of main
```

Exercício 5 – Solução B

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

// Declarações de variáveis
typedef struct {
        char nome[51];
        char sexo;
        int matricula;
        float salario;
        char endereco[81];
        char cargo[21];
} funcionario_t;

void LerNome(funcionario_t *func, int i);
void LerSexo(funcionario_t *func, int i);
void LerMatricula(funcionario_t *func, int i);
void LerEndereco(funcionario_t *func, int i);
void LerCargo(funcionario_t *func, int i);
```

```
int main()
{
        int  num, i, j, mat=-1;
        char c;
        funcionario_t funcionarios[100];

    // Lendo o número de funcionários a serem
       cadastrados
       do {
                printf("\nEntre o numero de funcionarios a
                                        serem cadastrados: ");
                scanf("%d", &num);
                fflush(stdin);
       } while (num <= 0   ||   num > 100);
```

Exercício 5 – Solução B (cont.)

```
// Lendo os dados de cada funcionário
for (i=0; i < num; i++)
{
    // Pulando linhas para clareza na tela
    printf("\n\n");

    // Lendo o nome de um funcionário
    LerNome(&funcionarios[i], i);

    // Lendo o sexo de um funcionário
    LerSexo(&funcionarios[i], i);

    // Lendo a matrícula de um funcionário
    LerMatricula(&funcionarios[i], i);

    // Lendo o endereço de um funcionário
    LerEndereco(&funcionarios[i], i);

    // Lendo o cargo de um funcionário
    LerCargo(&funcionarios[i], i);

}   // For loop para ler os dados dos funcionários
```

```
    // Escrevendo o número de funcionários cadastrados
    printf("FORAM CADASTRADOS %d
                                FUNCIONARIOS!\n", num);

     // Escrevendo os dados de cada funcionário
    for (i=0; i < num; i++)
    {
        printf("\n\nFUNCIONARIO [%d]\n", i+1);
        printf("    Nome  : %s\n", funcionarios[i].nome);
        printf("    Sexo  : %c\n", funcionarios[i].sexo);
        printf("    Matr  : %d\n", funcionarios[i].matricula);
        printf("    End   : %s\n", funcionarios[i].endereco);
        printf("    Cargo : %s\n", funcionarios[i].cargo);
    }

    // Pulando linhas para clareza na tela
    printf("\n\n");

    // Colocando uma pausa na tela
    system("pause");
    return(0);

}  // end of main
```

Exercício 5 – Solução B (cont.)

```
void LerNome(funcionario_t *func, int i)
{
    int j = 0;
    char c;

    printf("Nome [%d] (max 50 chars): ", i+1);
    do {
        c = getchar();
        if (c == '\n') c = '\0';
        if ((c>='a' && c<='z') || (c>='A' && c<='Z') ||
                            (c==' ')  ||   (c=='\0'))   {
            func->nome[j] = c;
            if (c!='\0') j++;
        }
    } while ((j < 49) && ((c!= '\0') || (c=='\0' &&
                            strlen(func->nome)==0)));

    if (c != '\0') func->nome[j] = '\0';

    return;
}
```

```
void LerSexo(funcionario_t *func, int i)
{
    char c;
    do {
        printf("Sexo [%d] ('f','F', 'm', 'M'): ", i+1);
        c = getchar();
        fflush(stdin);
    } while (c!='f' && c!='F' && c!='m' && c!='M');
    func->sexo = c;
    return;
}

void LerMatricula(funcionario_t *func, int i)
{
    int mat;
    do {
        printf("Matricula [%d]: ", i+1);
        scanf("%d", &mat);
        func->matricula = mat;
        fflush(stdin);
    } while (mat < 0);

    return;
}
```

Exercício 5 – Solução B (cont.)

```
void LerEndereco(funcionario_t *func, int i)
{
    char c;
    int j = 0;

    printf("Endereco [%d] (max 80 chars): ", i+1);
    do {
        c = getchar();
        if (c == '\n') c = '\0';
        func->endereco[j] = c;
        if (c!='\0') j++;
    } while ((j < 79) && ((c!= '\0') || (c=='\0' &&
                    strlen(func->endereco)==0)));

    if (c != '\0') func->endereco[j] = '\0';

    return;
}
```

```
void LerCargo(funcionario_t *func, int i)
{
    char c;
    int j = 0;

    printf("Cargo [%d] (max 20 chars): ", i+1);
    do {
        c = getchar();
        if (c == '\n') c = '\0';
        if ((c>='a' && c<='z') || (c>='A' && c<='Z') ||
                                                (c=='\0'))
        {
            func->cargo[j] = c;
            if (c!='\0') j++;
        }
    } while ((j < 19) && ((c != '\0') || (c == '\0' &&
                        strlen(func->cargo)==0)));

    if (c != '\0') func->cargo[j] = '\0';

    return;
}
```

Exercício 6

Modifique o programa anterior de modo que ele não mais defina um vetor com número fixo de estruturas, mas sim um vetor de ponteiros para a estrutura desejada e que faça a alocação de memória para os dados de cada funcionário usando a função **malloc**, à medida em que o usuário for entrando estes dados.

Ao final do programa a função **free** deverá ser invocada de forma apropriada para liberar toda a memória alocada com a função malloc.

Exercício 6 – Solução

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

// Declarações de variáveis
typedef struct {
        char nome[51];
        char sexo;
        int matricula;
        float salario;
        char endereco[81];
        char cargo[21];
} funcionario_t;

void LerNome(funcionario_t *func, int i);
void LerSexo(funcionario_t *func, int i);
void LerMatricula(funcionario_t *func, int i);
void LerEndereco(funcionario_t *func, int i);
void LerCargo(funcionario_t *func, int i);
```

```
int main()
{
        int  num, i, j, mat=-1;
        char c;
        funcionario_t *funcionarios[100];

    // Lendo o número de funcionários a serem
       cadastrados
       do {
                printf("\nEntre o numero de funcionarios a
                                        serem cadastrados: ");
                scanf("%d", &num);
                fflush(stdin);
       } while (num <= 0   ||   num > 100);
```

Exercício 6 – Solução (cont.)

```
// Lendo os dados de cada funcionário
for (i=0; i < num; i++)
{
    // Alocando memória para um funcionário
    funcionarios[i] =  (funcionario_t *)
                    malloc(sizeof(funcionario_t));

    // Pulando linhas para clareza na tela
    printf("\n\n");

    // Lendo dados de um funcionário
    LerNome(funcionarios[i], i);
    LerSexo(funcionarios[i], i);
    LerMatricula(funcionarios[i], i);
    LerEndereco(funcionarios[i], i);
    LerCargo(funcionarios[i], i);

}   // For loop para ler os dados dos funcionários

// Escrevendo o número de funcionários
// cadastrados
printf("FORAM CADASTRADOS %d
                        FUNCIONARIOS!\n", num);
```

```
    // Escrevendo os dados de cada funcionário e
    // liberando a memória correspondente (um a um)
    for (i=0; i < num; i++)
    {
        printf("\n\nFUNCIONARIO [%d]\n", i+1);
        printf("     Nome  : %s\n", funcionarios[i]->nome);
        printf("     Sexo  : %c\n", funcionarios[i]->sexo);
        printf("     Matr  : %d\n", funcionarios[i]->matricula);
        printf("     End   : %s\n", funcionarios[i]->endereco);
        printf("     Cargo : %s\n", funcionarios[i]->cargo);

        free(funcionarios[i]);
    }

    // Pulando linhas para clareza na tela
    printf("\n\n");

    // Colocando uma pausa na tela
    system("pause");
    return(0);

}  // end of main
```

Exercício 7

Modifique o programa anterior de modo que ele não mais defina um vetor com número fixo de estruturas, mas defina apenas um (somente um) ponteiro para o tipo da estrutura desejada e faça, usando a função **malloc**, a alocação de toda memória necessária para os dados dos funcionários após ler o número de empregados que se deseja cadastrar.

Ao final do programa a função **free** deverá ser invocada de forma apropriada para liberar a memória alocada com a função malloc.

O acesso às várias estruturas correspondentes aos dados dos empregados poderá ser feito com notação de vetor ou de aritmética de ponteiros.

Exercício 7 – Solução

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

// Declarações de variáveis
typedef struct {
        char nome[51];
        char sexo;
        int matricula;
        float salario;
        char endereco[81];
        char cargo[21];
} funcionario_t;

void LerNome(funcionario_t *func, int i);
void LerSexo(funcionario_t *func, int i);
void LerMatricula(funcionario_t *func, int i);
void LerEndereco(funcionario_t *func, int i);
void LerCargo(funcionario_t *func, int i);
```

```
int main()
{
        int  num, i, j, mat=-1;
        char c;
        funcionario_t *funcionarios;

    // Lendo o número de funcionários a serem
       cadastrados
       do {
                printf("\nEntre o numero de funcionarios a
                                        serem cadastrados: ");
                scanf("%d", &num);
                fflush(stdin);
       } while (num <= 0   ||   num > 100);

 // Alocando memória para todos os funcionários
 funcionarios=  (funcionario_t *)
                  malloc(num * sizeof(funcionario_t));
```

Exercício 7 – Solução (cont.)

```
// Lendo os dados de cada funcionário
for (i=0; i < num; i++)
{
    // Pulando linhas para clareza na tela
    printf("\n\n");

    // Lendo dados de um funcionário
    LerNome(funcionarios + i, i);
    LerSexo(funcionarios + i, i);
    LerMatricula(funcionarios + i, i);
    LerEndereco(funcionarios + i, i);
    LerCargo(funcionarios + i, i);

}   // For loop para ler os dados dos funcionários

// Escrevendo o número de funcionários
// cadastrados
printf("FORAM CADASTRADOS %d
                        FUNCIONARIOS!\n", num);
```

```
// Escrevendo os dados de cada funcionário
for (i=0; i < num; i++)
{
    printf("\n\nFUNCIONARIO [%d]\n", i+1);
    printf(" Nome  : %s\n", (funcionarios + i)->nome);
    printf(" Sexo  : %c\n", funcionarios + i)->sexo);
    printf(" Matr  : %d\n", (funcionarios + i)->matricula);
    printf(" End   : %s\n", (funcionarios + i)->endereco);
    printf(" Cargo : %s\n", (funcionarios + i)->cargo);
}

// Liberando a memória para todos os funcionarios
free(funcionarios);

// Pulando linhas para clareza na tela
printf("\n\n");

// Colocando uma pausa na tela
system("pause");
return(0);

}  // end of main
```