Programação Procedimental
GBC014 – 2015/1
Prof. Renan Cattelan – www.facom.ufu.br/~renan

Prática 10

Estruturas e alocação dinâmica

Exercício

- Crie um programa para armazenar uma agenda de contatos pessoais usando estrutura.
- Faça funções para
  1. listar contatos,
  2. pesquisar por nome,
  3. pesquisar por telefone.
- Um contato deve ser composto por nome, endereço, telefone e e-mail. Use alocação dinâmica para o armazenamento do nome do contato. Para definir e armazenar a lista de contatos, use um vetor de estrutura, também alocado dinamicamente.

Estruturas

- Como visto na prática 8, uma ***estrutura*** é uma estrutura de dados criada a partir de tipos básicos
- Na prática, funciona como uma coleção de variáveis, possivelmente de tipos diferentes

```
struct nome_estrutura {
    tipo_1   nome_campo_1;
    tipo_2   nome_campo_2;
    ...
    tipo_n nome_campo_n;
} variaveis_estrutura;
```

```
struct aluno{
        char nome[40];
        int numero;
        float media_notas;
} a1, a2;
```

Exercício

- Crie um programa para armazenar uma agenda de contatos pessoais usando estrutura.
- Faça funções para
  1. listar contatos,
  2. pesquisar por nome,
  3. pesquisar por telefone.
- Um contato deve ser composto por nome, endereço, telefone e e-mail. Use alocação dinâmica para o armazenamento do nome do contato. Para definir e armazenar a lista de contatos, use um vetor de estrutura, também alocado dinamicamente.

```c
struct contato {
        char *nome;
        char endereco[50];
        int telefone;
        char email[30];
};
typedef struct contato Contato;
```

Exercício

- Crie um programa para armazenar uma agenda de contatos pessoais usando estrutura.
- Faça funções para
  1. listar contatos,
  2. pesquisar por nome,
  3. pesquisar por telefone.
- Um contato deve ser composto por nome, endereço, telefone e e-mail. Use alocação dinâmica para o armazenamento do nome do contato. Para definir e armazenar a lista de contatos, use um vetor de estrutura, também alocado dinamicamente.

Alocação de memória

- Estática (antes da execução do programa)

```
int opcao;
char nome[50];
int telefone;
```

- Dinâmica (durante a execução do programa)

```
char *ptr = malloc(1);
scanf( "%c", ptr);

char *nome = (char *) malloc(sizeof(char)*50);
```

Alocação de memória

Função malloc():

- **abreviatura de *memory allocation***
- **aloca um bloco de bytes consecutivos na memória do computador e devolve o endereço desse bloco**
- **o número de bytes é especificado no argumento da função**
- **o endereço devolvido por malloc é do tipo "genérico" (void *)**
- **neste exemplo, malloc aloca 1 byte, espaço suficiente para armazenar 1 caractere**

```
char *ptr = malloc(1);
scanf( "%c", ptr);

char *nome = (char *) malloc(sizeof(char)*50);
```

Alocação de memória

- Estática (antes da execução do programa)

```
int opcao;
char nome[50];
int telefone;
```

- Dinâmica (durante programa)

```
char *ptr = malloc(1);
scanf( "%c", ptr);

char *nome = (char *) malloc(sizeof(char)*50);
```

- é possível recorrer ao operador sizeof, que informa quantos bytes um tipo de dado especifico tem
- no exemplo, malloc aloca uma string de 50 caracteres

```c
int main(void)
{
    Contato *vContatos; // vetor de contatos
    int quant; // tamanho do vetor de contatos

    printf("\nQuantos contatos deseja cadastrar? ");
    scanf("%d", &quant);

    // alocacao dinamica do vetor
    vContatos = (Contato *) malloc(sizeof(Contato)*quant);
    // cadastro dos contatos
    int i;
    for (i=0; i<quant; i++) {
        printf("\nContato %d - digite nome: ", i+1);
        // alocacao dinamica do nome
        vContatos[i].nome = (char *) malloc(sizeof(char)*50);
        scanf(" %50[^\n]%*c", vContatos[i].nome);

        printf("\nContato %d - digite endereco: ", i+1);
        scanf(" %50[^\n]%*c", vContatos[i].endereco);

        printf("\nContato %d - digite telefone: ", i+1);
        scanf("%d", &(vContatos[i].telefone));

        printf("\nContato %d - digite email: ", i+1);
        scanf(" %30[^\n]%*c", vContatos[i].email);
    }
```

```c
int main(void)
{
    Contato *vContatos; // vetor de contatos
    int quant; // tamanho do vetor de contatos

    printf("\nQuantos contatos deseja cadastrar? ");
    scanf("%d", &quant);

    // alocacao dinamica do vetor
    vContatos = (Contato *) malloc(sizeof(Contato)*quant);
    // cadastro dos contatos
    int i;
    for (i=0; i<quant; i++) {
        printf("\nContato %d - digite nome: ", i+1);
        // alocacao dinamica do nome
        vContatos[i].nome = (char *) malloc(sizeof(char)*50);
        scanf(" %50[^\n]%*c", vContatos[i].nome);

        printf("\nContato %d - digite endereco: ", i+1);
        scanf(" %50[^\n]%*c", vContatos[i].endereco);

        printf("\nContato %d - digite telefone: ", i+1);
        scanf("%d", &(vContatos[i].telefone));

        printf("\nContato %d - digite email: ", i+1);
        scanf(" %30[^\n]%*c", vContatos[i].email);
    }
```

Alocação dinâmica do vetor de contatos

Alocação dinâmica do nome do contato

```c
int main(void)
{
    Contato *vContatos; // vetor de contatos
    int
    p
    s
    /
    v
    /
    i
    fo
        // alocac          ca do nome
        vContatos[i]    e = (char *) malloc(sizeof(char)*50);
        scanf(" %50[^\n]%*c", vContatos[i].nome);

        printf("\nContato %d - digite endereco: ", i+1);
        scanf(" %50[^\n]%*c", vContatos[i].endereco);

        printf("\nContato %d - digite telefone: ", i+1);
        scanf("%d", &(vContatos[i].telefone));

        printf("\nContato %d - digite email: ", i+1);
        scanf(" %30[^\n]%*c", vContatos[i].email);
    }
```

- o espaço inicial (' ') consome qualquer \n que já esteja na stream stdin
- 50 é o limite para o número de caracteres lido
- [^\n] aceita qualquer caracter até a entrada de um \n
- finalmente, %*c consome qualquer \n pressionado após a entrada da string

Exercício

- Crie um programa para armazenar uma agenda de contatos pessoais usando estrutura.
- Faça funções para
  1. listar contatos,
  2. pesquisar por nome,
  3. pesquisar por telefone.
- Um contato deve ser composto por nome, endereço, telefone e e-mail. Use alocação dinâmica para o armazenamento do nome do contato. Para definir e armazenar a lista de contatos, use um vetor de estrutura, também alocado dinamicamente.

```c
void listar_contatos(Contato *v, int q) {
    int i;
    for (i=0; i<q; i++) {
        printf("\nContato %d => %s|%s|%d|%s \n", i+1,
            v[i].nome,v[i].endereco,v[i].telefone,v[i].email);
    }
}
```

Exercício

- Crie um programa para armazenar uma agenda de contatos pessoais usando estrutura.
- Faça funções para
  1. listar contatos,
  2. pesquisar por nome,
  3. pesquisar por telefone.
- Um contato deve ser composto por nome, endereço, telefone e e-mail. Use alocação dinâmica para o armazenamento do nome do contato. Para definir e armazenar a lista de contatos, use um vetor de estrutura, também alocado dinamicamente.

```c
int pesquisar_nome(Contato *v, int q, char *nome) {
    int i;
    for (i=0; i<q; i++) {
        if (strcmp(v[i].nome,nome)==0) {
            printf("\nEncontrado: Contato %d => %s|%s|%d|%s \n", i+1,
                v[i].nome,v[i].endereco,v[i].telefone,v[i].email);
            return 0;
        }
    }
    printf("\nContato nao encontrado!\n");
    return 1;
}
```

Exercício

- Crie um programa para armazenar uma agenda de contatos pessoais usando estrutura.
- Faça funções para
  1. listar contatos,
  2. pesquisar por nome,
  3. pesquisar por telefone.
- Um contato deve ser composto por nome, endereço, telefone e e-mail. Use alocação dinâmica para o armazenamento do nome do contato. Para definir e armazenar a lista de contatos, use um vetor de estrutura, também alocado dinamicamente.

```c
int pesquisar_telefone(Contato *v, int q, int telefone) {
    int i;
    for (i=0; i<q; i++) {
        if (v[i].telefone == telefone) {
            printf("\nEncontrado: Contato %d => %s|%s|%d|%s \n", i+1,
                v[i].nome,v[i].endereco,v[i].telefone,v[i].email);
            return 0;
        }
    }
    printf("\nContato nao encontrado!\n");
    return 1;
}
```

```c
int opcao;
char nome[50];
int telefone;
do {
    printf("\nEscolha uma opcao:");
    printf("\n1 - Listar contatos");
    printf("\n2 - Pesquisar por nome");
    printf("\n3 - Pesquisar por telefone");
    printf("\n4 - Sair\n=> ");
    scanf("%d", &opcao);
    switch (opcao) {
        case 1:
            listar_contatos(vContatos, quant);
            break;
        case 2:
            printf("\nDigite o nome desejado: ");
            scanf(" %50[^\n]%*c", nome);
            pesquisar_nome(vContatos, quant, nome);
            break;
        case 3:
            printf("\nDigite o telefone desejado: ");
            scanf("%d", &telefone);
            pesquisar_telefone(vContatos, quant, telefone);
            break;
        case 4:
            for (i=0; i<quant; i++)
                free(vContatos[i].nome);
            free(vContatos);
            printf("\nTchau!\n");
            break;
        default:
            printf("\nOpcao invalida. Tente novamente.\n");
    }
} while (opcao != 4);
```