MC102 - Algoritmos e Progração de Computador

Prof. Alexandre Xavier Falcão

17° Aula: Funções (cont.)

1 Hierarquia entre funções

Como mencionado na aula anterior, uma função pode conter chamadas a outras funções. Após a execução de uma função, o programa sempre retorna para a posição imediatamente após a sua chamada. O exemplo abaixo demonstra leitura, ordenação e busca binária em uma lista de telefones de amigos.

```
#include <stdio.h>

#define N 5 /* tamanho da agenda de amigos, que na prática deve ser bem maior
               para justificar a busca binária. */

typedef enum {false,true} bool;

typedef struct _agenda {
  char nome[50];
  int  telefone;
} Agenda;

void      le_nome(char *nome); /* lê nome trocando \n por \0 */

void      le_dados(Agenda *amigo); /* lê nome e telefone de cada
                                      amigo, carregando esses dados
                                      em um vetor de registros. */
void      troca(Agenda *t1, Agenda *t2); /* troca dados do
                                            conteúdo de dois
                                            registros */
void      ordena(Agenda *amigo); /* ordena registros por seleção */
/* Realiza buscas binárias por nome, retornando false se o nome não
   for encontrado, e true no caso contrário. Neste caso, retorna o
   telefone encontrado no endereço apontado pela variável telefone. */
bool      busca(Agenda *amigo, char *nome, int *telefone);

/* Escopos das funções */

void le_nome(char *nome)
```

```
{
  int i,comp;

  fgets(nome,49,stdin);
  comp = strlen(nome);
  nome[comp-1]='\0'; /* sobrescreve \0 no lugar de \n */
}

void le_dados(Agenda *amigo)
{
  char linha[100],*p;
  int  i,comp;

  for (i=0; i < N; i++) {
    fgets(linha,99,stdin); /* lê linha */
    p = (char *)strchr(linha,'#'); /* procura posição do delimitador */
    comp = p-linha-1; /* acha o número de caracteres antes do delimitador */
    strncpy(amigo[i].nome,linha,comp); /* copia esses caracteres */
    amigo[i].nome[comp]='\0'; /* acrescenta o final de string */
    sscanf(p+1,"%d",&amigo[i].telefone); /* lê telefone a partir da
                                            posição do delimitador + 1 */
  }
}

void troca(Agenda *t1, Agenda *t2)
{
  Agenda t;

  t   = *t1;
  *t1 = *t2;
  *t2 = t;
}

void ordena(Agenda *amigo)
{
  int i,j,jm;

  for (j=0; j < N-1; j++){
  /* seleciona o menor nome */
    jm    = j;
    for (i=j+1; i < N; i++){
      if (strcmp(amigo[i].nome,amigo[jm].nome) < 0){
        jm    = i;
      }
    }
```

```
    /* troca */
    if (j != jm)
      troca(&amigo[j],&amigo[jm]);
  }
}

bool busca(Agenda *amigo, char *nome, int *telefone)
{
  int inicio, fim, pos;
  bool achou;


    inicio = 0;
    fim    = N-1;
    achou  = false;

    while ((inicio <= fim)&&(!achou)){
      pos = (inicio+fim)/2;
      if (strcmp(nome,amigo[pos].nome) < 0)
        fim = pos-1;
      else
        if (strcmp(nome,amigo[pos].nome) > 0)
          inicio = pos + 1;
        else{
          achou = true;
          *telefone = amigo[pos].telefone;
        }
    }
    return(achou);
}

int main()
{
  Agenda amigo[N];
  char nome[50];
  int telefone;

  le_dados(amigo);
  ordena(amigo);

  printf("Entre com um nome ou \nfim para sair do programa:\n");
  le_nome(nome);

  while(strcmp(nome,"fim")){
    if (busca(amigo,nome,&telefone))
      printf("O número do telefone de %s é: %d\n",nome,telefone);
```

```
    else
      printf("%s não está na lista de amigos\n",nome);

    printf("Entre com um nome ou\nfim para sair do programa\n");
    le_nome(nome);
  }
  return 0;
}
```