Programação de Computadores II

Cap. 4 – Funções

Livro: Waldemar Celes, Renato Cerqueira, José Lucas Rangel. Introdução a Estruturas de Dados, Editora Campus (2004)

Slides adaptados dos originais dos profs.: Marco Antonio Casanova e Marcelo Gattass (PUC-Rio)

Referências

Waldemar Celes, Renato Cerqueira, José Lucas Rangel, *Introdução a Estruturas de Dados*, Editora Campus (2004)

Capítulo 4 – Funções

Tópicos

- Definição de funções
- Pilha de execução
- Variáveis globais e variáveis estáticas

Definição de Funções

- Comando para definição de função:

```
tipo_retornado  nome_da_função  ( lista de parâmetros... )
{
   corpo da função
}
```

```
/* programa que lê um número e imprime seu fatorial */

#include <stdio.h>
void fat (int n);
int main (void)
{  int n, r;
   printf(“Digite um número nao negativo:”
   scanf("%d", &n);
   fat(n);
   return 0;
}

/* função para calcular o valor do fatorial
void fat (int n)
{  int i;
   int f = 1;
   for (i = 1; i <= n; i++)
      f *= i;
   printf(“Fatorial = %d”, f);
}
```

“protótipo” da função:
deve ser incluído antes
da função ser chamada

chamada da função

“main” retorna um inteiro:
0 : execução OK
≠ 0 : execução ¬OK

declaração da função:
indica o tipo da saída e
o tipo e nome das entradas

```
void fat (int n);   /* obs: existe ; no protótipo */


void fat(int n)     /* obs: não existe ; na declaração */
{
}
```

```c
/* programa que lê um número e imprime seu fatorial (versão
2) */
#include <stdio.h>
int fat (int n);
int main (void)
{  int n, r;
   printf("Digite um número nao negativo:");
   scanf("%d", &n);
   r = fat(n);
   printf("Fatorial = %d\n", r);
   return 0;
}

/* função para calcular o valor do fatorial
int fat (int n)
{  int i;
   int f = 1;
   for (i = 1; i <= n; i++)
      f *= i;
   return f;
}
```

"protótipo" da função: deve ser incluído antes da função ser chamada

chamada da função

"main" retorna um inteiro:
0 : execução OK
≠ 0 : execução ¬OK

declaração da função: indica o tipo da saída e o tipo e nome das entradas

retorna o valor da função

Pilha de Execução

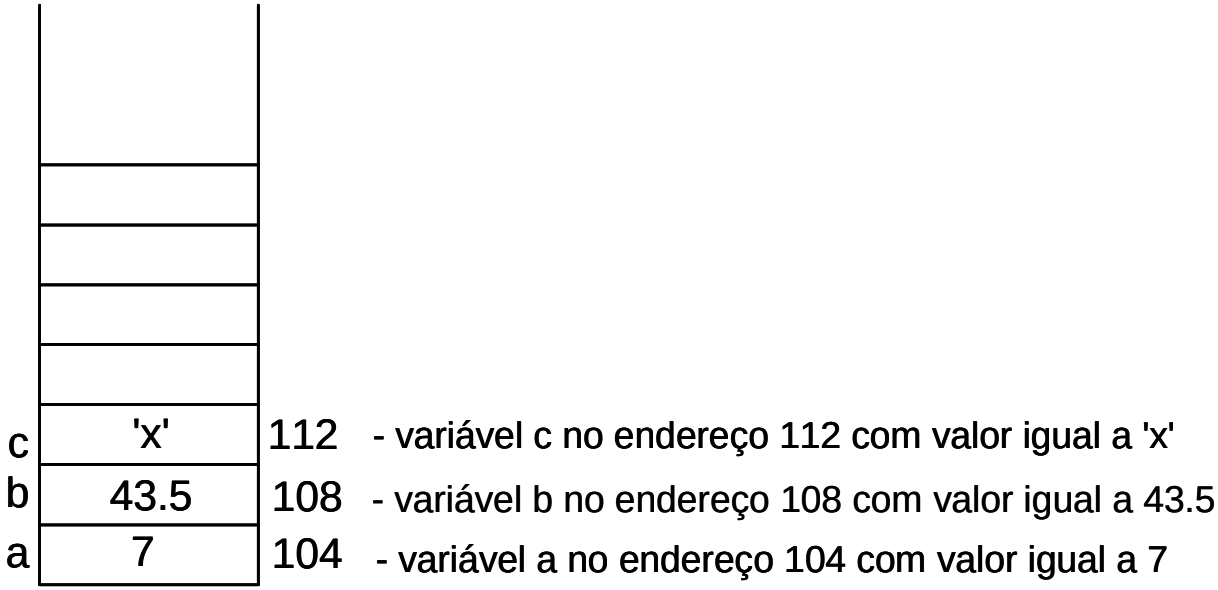
- Comunicação entre funções:
  - funções são independentes entre si
  - transferência de dados entre funções:
    - através dos parâmetros e do valor de retorno da função chamada
    - passagem de parâmetros é feita ***por valor***
  - variáveis locais a uma função:
    - definidas dentro do corpo da função (incluindo os parâmetros)
    - não existem fora da função
    - são criadas cada vez que a função é executada
    - deixam de existir quando a execução da função terminar

Pilha de Execução

- Comunicação entre funções (cont.):

  Pergunta: Como implementar a comunicação entre funções?

  Resposta: Através de uma pilha



Exemplo: Fatorial iterativo

```
/* programa que lê um numero e imprime seu fatorial (versão 3) */
#include <stdio.h>
int fat (int n);
int main (void)
{  int n = 5;
   int r;
   r = fat ( n );
   printf("Fatorial de %d = %d \n", n, r);
   return 0;
}
int fat (int n)
{  int f = 1;
   while (n != 0) {
      f *= n;
      n--;
   }
   return f;
}
```

declaração das variáveis n e `r`, locais à função `main`

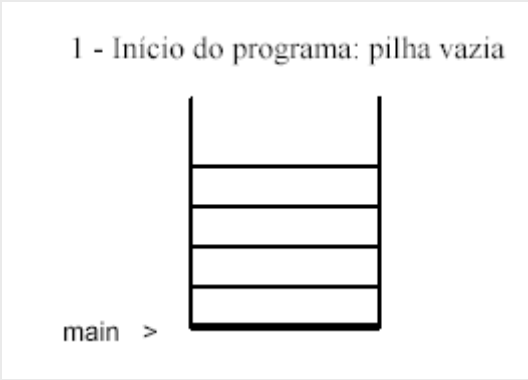
declaração das variáveis n e `f`, locais à função `fat`

alteração no valor de n em `fat` não altera o valor de n em `main`

simulação da chamada `fat(5)`:
a variável n possui valor 0 ao final da execução de `fat`,
mas o valor de n no programa principal ainda será 5

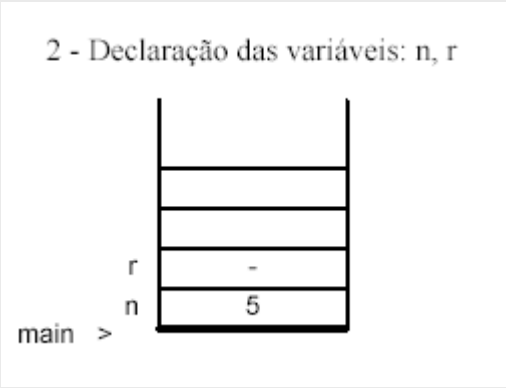
Exemplo: Início do programa

```
/* programa que lê um numero e imprime seu fatorial (versão 3) */
#include <stdio.h>
int fat (int n);
int main (void)
{  int n = 5;
   int r;
   r = fat ( n );
   printf("Fatorial de %d = %d \n", n, r);
   return 0;
}
int fat (int n)
{  int f = 1;
   while (n != 0) {
      f *= n;
      n--;
   }
   return f;
}
```



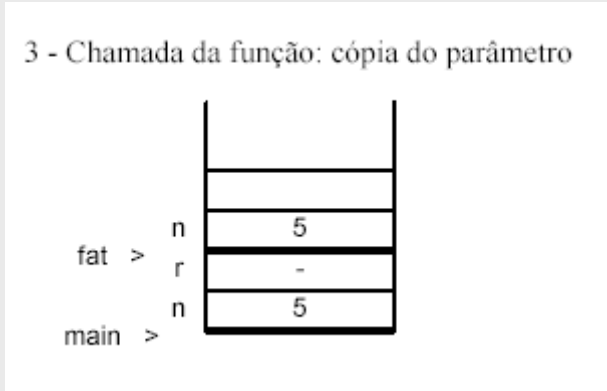
Exemplo: Declaração de `n` e `r` na `main()`

```
/* programa que lê um numero e imprime seu fatorial (versão 3) */
#include <stdio.h>
int fat (int n);
int main (void)
{  int n = 5;
   int r;
   r = fat ( n );
   printf("Fatorial de %d = %d \n", n, r);
   return 0;
}
int fat (int n)
{  int f = 1;
   while (n != 0) {
      f *= n;
      n--;
   }
   return f;
}
```



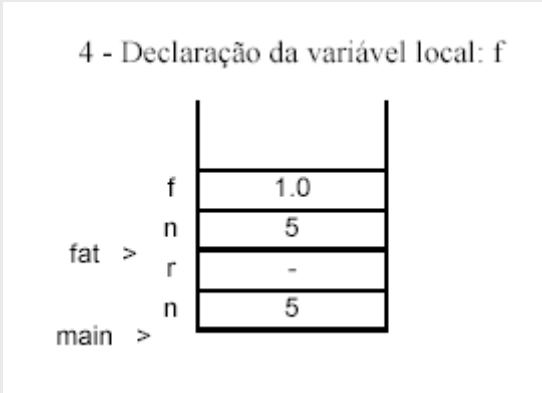
Exemplo: Declaração de `n` na `fat(int n)`

```
/* programa que lê um numero e imprime seu fatorial (versão 3) */
#include <stdio.h>
int fat (int n);
int main (void)
{  int n = 5;
   int r;
   r = fat ( n );
   printf("Fatorial de %d = %d \n", n, r);
   return 0;
}
int fat (int n)
{  int f = 1;
   while (n != 0) {
      f *= n;
      n--;
   }
   return f;
}
```



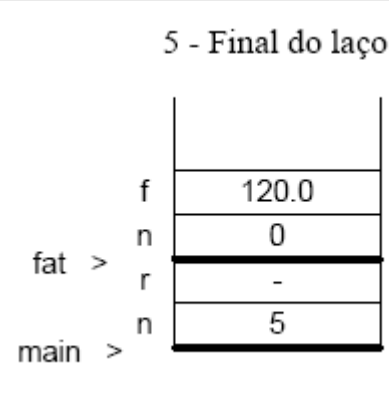
Exemplo: Declaração de `n` e `f` na `fat(int n)`

```
/* programa que lê um numero e imprime seu fatorial (versão 3) */
#include <stdio.h>
int fat (int n);
int main (void)
{  int n = 5;
   int r;
   r = fat ( n );
   printf("Fatorial de %d = %d \n", n, r);
   return 0;
}
int fat (int n)
{  int f = 1;
   while (n != 0) {
      f *= n;
      n--;
   }
   return f;
}
```



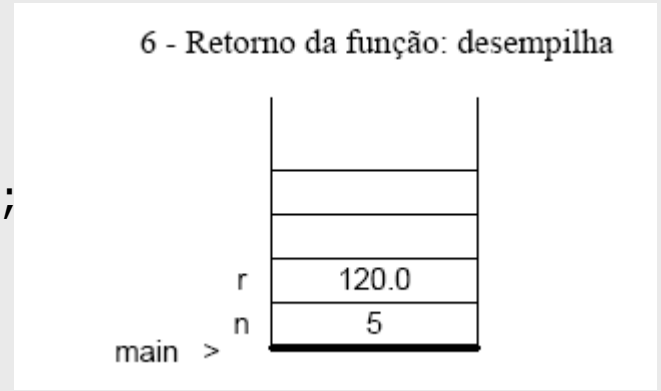
Exemplo: no final do laço

```
/* programa que lê um numero e imprime seu fatorial (versão 3) */
#include <stdio.h>
int fat (int n);
int main (void)
{  int n = 5;
   int r;
   r = fat ( n );
   printf("Fatorial de %d = %d \n", n, r);
   return 0;
}
int fat (int n)
{  int f = 1;
   while (n != 0) {
      f *= n;
      n--;
   }
   return f;
}
```



Exemplo: no retorno

```
/* programa que lê um numero e imprime seu fatorial (versão 3) */
#include <stdio.h>
int fat (int n);
int main (void)
{  int n = 5;
   int r;
   r = fat ( n );
   printf("Fatorial de %d = %d \n", n, r);
   return 0;
}
int fat (int n)
{  int f = 1;
   while (n != 0) {
      f *= n;
      n--;
   }
   return f;
}
```



Variáveis Globais

- Variável global:
  - declarada fora do corpo das funções:
    - visível por todas as funções subseqüentes
  - não é armazenada na pilha de execução:
    - não deixa de existir quando a execução de uma função termina
    - existe enquanto o programa estiver sendo executado
  - utilização de variáveis globais:
    - deve ser feito com critério
    - pode-se criar um alto grau de interdependência entre as funções
    - dificulta o entendimento e o reuso do código

Variáveis Globais

```
#include <stdio.h>

int s, p;  /* variáveis globais */

void somaprod (int a, int b)
{
   s = a + b;
   p = a * b;
}

int main (void)
{
   int x, y;
   scanf("%d %d", &x, &y);
   somaprod(x,y);
   printf("Soma = %d  produto = %d\n", s, p);
   return 0;
}
```

Variáveis Estáticas

- Variável estática:
  - declarada no corpo de uma função:
    - visível apenas dentro da função em que foi declarada
  - não é armazenada na pilha de execução:
    - armazenada em uma área de memória estática
    - continua existindo antes ou depois de a função ser executada
  - utilização de variáveis estáticas:
    - quando for necessário recuperar o valor de uma variável atribuída na última vez que a função foi executada

Variáveis Estáticas

- Exemplo:

```
void imprime ( float a )
{
   static int n = 1;
   printf(" %f   ", a);
   if ((n % 5) == 0) printf(" \n ");
   n++;
}
```

–função para imprimir números reais:

•imprime um número por vez, separando-os por espaços em branco e colocando, no máximo, cinco números por linha

Comentários

- variáveis estáticas e variáveis globais:
  - são inicializadas com zero,
    se não forem explicitamente inicializadas
- variáveis globais estáticas:
  - são visíveis para todas as funções subseqüentes
  - não podem ser acessadas por funções definidas
    em outros arquivos
- funções estáticas:
  - não podem ser chamadas por funções definidas
    em outros arquivos