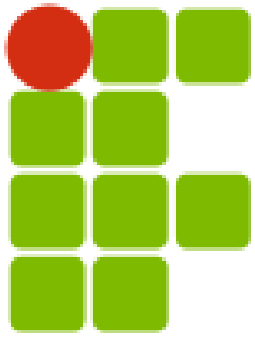


INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PARÁ

Estrutura de Dados

Ricardo José Cabeça de Souza
www.ricardojcsouza.com.br
ricardo.souza@ifpa.edu.br

Parte 4

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PARÁ

www.ricardojcsouza.com.br
ricardo.souza@ifpa.edu.br

FUNÇÕES

- **FUNÇÕES**
  - As funções dividem grandes tarefas de computação em tarefas menores
  - Criação de funções evita a repetição de código

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

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PARÁ

www.ricardojcsouza.com.br
ricardo.souza@ifpa.edu.br

FUNÇÕES

- Exemplo

```
/* programa que le um numero e imprime seu fatorial */

#include <stdio.h>

void fat (int n);

/* Função principal */
int main (void)
{
   int n;
   scanf("%d", &n);
   fat(n);
   return 0;
}

/* Função para imprimir o valor do fatorial */
void fat ( int n )
{
   int i;
   int f = 1;
   for (i = 1; i <= n; i++)
      f *= i;
   printf("Fatorial = %d\n", f);
}
```

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PARÁ

www.ricardojcsouza.com.br
ricardo.souza@ifpa.edu.br

FUNÇÕES

- **FUNÇÕES**
  - Quando uma função não tem parâmetros, colocamos a palavra reservada **void** entre os parênteses
  - Uma função pode ter um valor de retorno associado

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PARÁ

www.ricardojcsouza.com.br
ricardo.souza@ifpa.edu.br

FUNÇÕES

- **PROTÓTIPO**
  - O *protótipo* de uma função consiste na repetição da linha de sua definição seguida do caractere (;)

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

int main (void)
{
   . . .
}

void fat (int n)       /* obs: nao existe ; na definição */
{
   . . .
}
```

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PARÁ

www.ricardojcsouza.com.br
ricardo.souza@ifpa.edu.br

FUNÇÕES

- **FUNÇÕES**
  - Retorno

```
/* programa que le um numero e imprime seu fatorial (versão 2) */

#include <stdio.h>

int fat (int n);

int main (void)
{
   int n, r;
   scanf("%d", &n);
   r = fat(n);
   printf("Fatorial = %d\n", r);
   return 0;
}

/* funcao 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;
}
```

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PARÁ

www.ricardojcsouza.com.br
ricardo.souza@ifpa.edu.br

FUNÇÕES

- **PILHA DE EXECUÇÃO**
  - As variáveis locais definidas dentro do corpo de uma função (e isto inclui os parâmetros das funções) não existem fora da função
    - Escopo local
    - Escopo global

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PARÁ

www.ricardojcsouza.com.br
ricardo.souza@ifpa.edu.br

FUNÇÕES

- **PASSAGEM DE PARÂMETROS POR VALOR**
  - O valor passado é atribuído ao parâmetro da função chamada

```
/* programa que le 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 = 1.0;
   while (n != 0)
   {
      f *= n;
      n--;
   }
   return f;
}
```

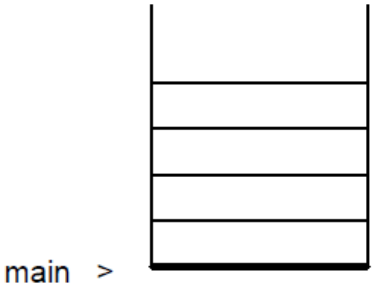
INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PARÁ

www.ricardojcsouza.com.br
ricardo.souza@ifpa.edu.br

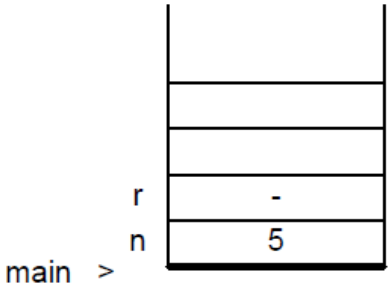
FUNÇÕES

- **MODELO DE PILHA**

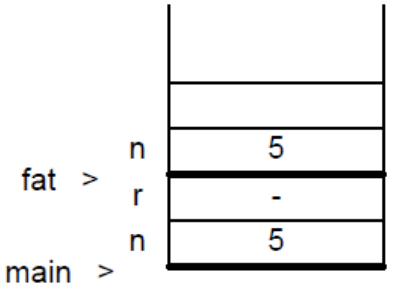
1 - Início do programa: pilha vazia



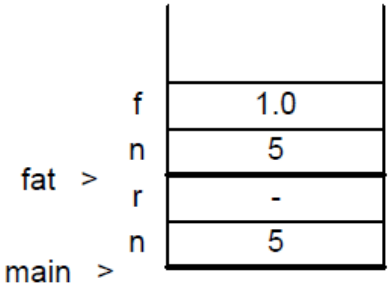
2 - Declaração das variáveis: n, r



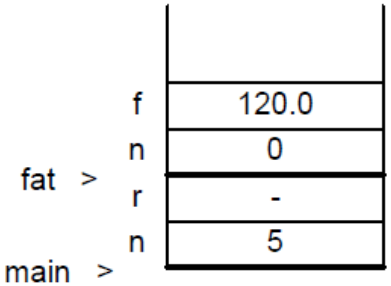
3 - Chamada da função: cópia do parâmetro



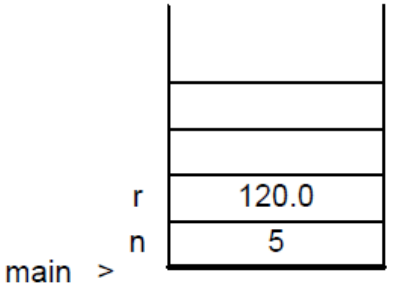
4 - Declaração da variável local: f



5 - Final do laço



6 - Retorno da função: desempilha



INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PARÁ

www.ricardojcsouza.com.br
ricardo.souza@ifpa.edu.br

FUNÇÕES

- **PONTEIRO DE VARIÁVEIS**
  - A linguagem C permite o armazenamento e a manipulação de valores de endereços de memória
  - Para cada tipo existente, há um tipo ponteiro que pode armazenar endereços de memória onde existem valores do tipo correspondente armazenados

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PARÁ

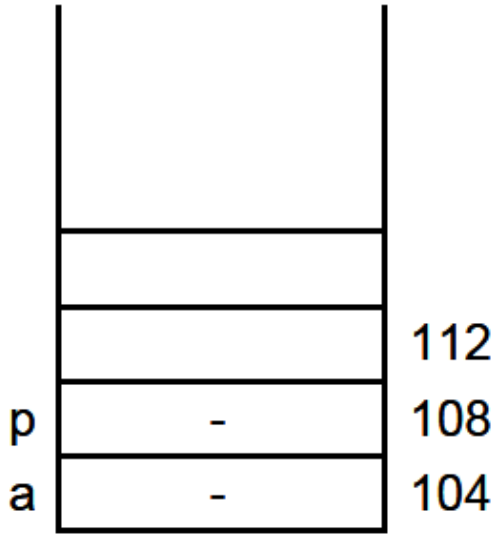
www.ricardojcsouza.com.br
ricardo.souza@ifpa.edu.br

FUNÇÕES

- **PONTEIRO DE VARIÁVEIS**
  - Após as declarações, ambas as variáveis, a e p, armazenam valores "lixo", pois não foram inicializadas

```
/*variável inteiro */
 int a;

/*variável ponteiro p/ inteiro */
 int *p;
```



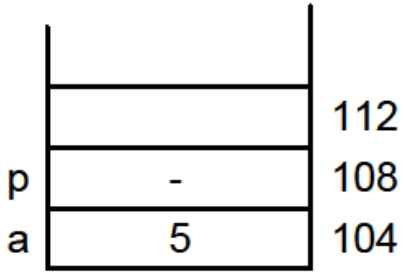
INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PARÁ

www.ricardojcsouza.com.br
ricardo.souza@ifpa.edu.br

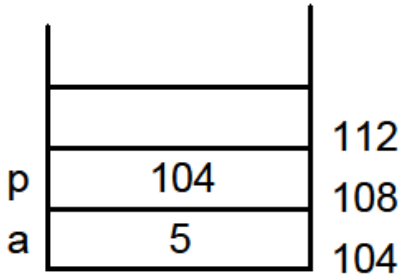
FUNÇÕES

- **PONTEIRO DE VARIÁVEIS**

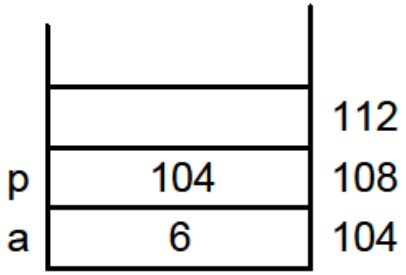
```
/* a recebe o valor 5 */
a = 5;
```




```
/* p recebe o endereço de a
   (diz-se p aponta para a) */
p = &a;
```




```
/* conteúdo de p recebe o valor 6 */
*p = 6;
```

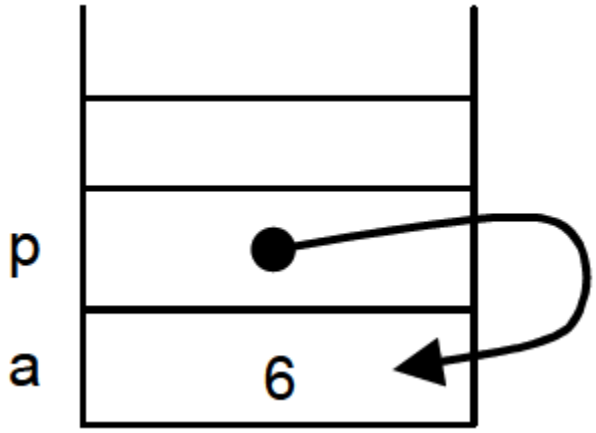


INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PARÁ

www.ricardojcsouza.com.br
ricardo.souza@ifpa.edu.br

FUNÇÕES

- **PONTEIRO DE VARIÁVEIS**
  - A variável **a** recebe, indiretamente, o valor 6
  - Acessar **a** é equivalente a acessar ***p**, pois **p** armazena o endereço de **a**
  - Dizemos que **p** *aponta* para **a**, daí o nome ***ponteiro***



INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PARÁ

www.ricardojcsouza.com.br
ricardo.souza@ifpa.edu.br

FUNÇÕES

- **PONTEIRO DE VARIÁVEIS**
  - Exemplo

```
int main ( void )
{
    int a;
    int *p;
    p = &a;
    *p = 2;
    printf(" %d ", a);
    return;
}
```

imprime o valor 2.

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PARÁ

www.ricardojcsouza.com.br
ricardo.souza@ifpa.edu.br

FUNÇÕES

- **PASSANDO PONTEIROS PARA FUNÇÕES**
  - Ponteiros oferecem meios de alterarmos valores de variáveis acessando-as indiretamente
  - Passagem de parâmetros por **referência**

www.ricardojcsouza.com.br
ricardo.souza@ifpa.edu.br

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA PARÁ

FUNÇÕES

- **PASSANDO PONTEIROS PARA FUNÇÕES**

```c
/* funcao troca (versao ERRADA) */

#include <stdio.h>

void troca (int x, int y )
{
   int temp;
   temp = x;
   x = y;
   y = temp;
}

int main ( void )
{
   int a = 5, b = 7;
   troca(a, b);
   printf("%d %d \n", a, b);
   return 0;
}
```

```c
/* funcao troca (versao CORRETA) */
#include <stdio.h>

void troca (int *px, int *py )
{
   int temp;
   temp = *px;
   *px = *py;
   *py = temp;
}


int main ( void )
{
   int a = 5, b = 7;
   troca(&a, &b);     /* passamos os endereços das variáveis */
   printf("%d %d \n", a, b);
   return 0;
}
```

Obs: Alguns compiladores usam “&“como representação do ponteiro na passagem de parâmetros.

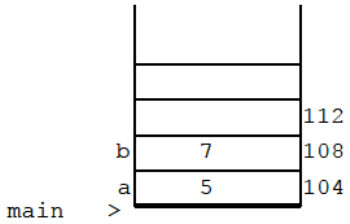
INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PARÁ

www.ricardojcsouza.com.br
ricardo.souza@ifpa.edu.br

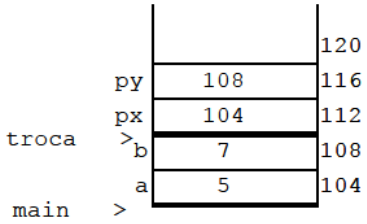
FUNÇÕES

- **PASSANDO PONTEIROS PARA FUNÇÕES**

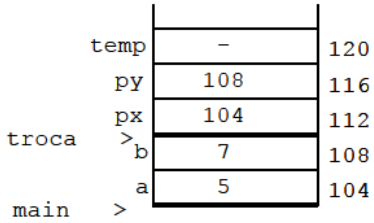
1 -Declaração das variáveis: a, b



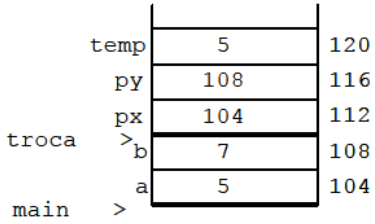
2 - Chamada da função: passa endereços



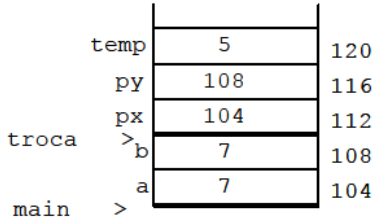
3 - Declaração da variável local: temp



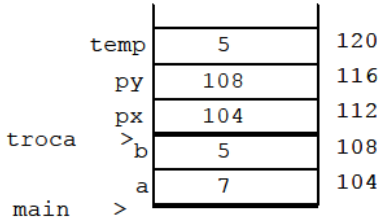
4 - temp recebe conteúdo de px



5 -Conteúdo de px recebe conteúdo de py



6 -Conteúdo de py recebe temp



INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PARÁ

www.ricardojcsouza.com.br
ricardo.souza@ifpa.edu.br

FUNÇÕES

- **RECURSIVIDADE**
  - As funções podem ser chamadas recursivamente, isto é, dentro do corpo de uma função podemos chamar novamente a própria função
  - Se uma função **A** chama a própria função **A**, dizemos que ocorre uma **recursão direta**
  - Se uma função **A** chama uma função **B** que, por sua vez, chama **A**, temos uma **recursão indireta**

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PARÁ

www.ricardojcsouza.com.br
ricardo.souza@ifpa.edu.br

FUNÇÕES

- **RECURSIVIDADE**
  - Exemplo

```
/* Função recursiva para calculo do fatorial */

int fat (int n)
{
   if (n==0)
      return 1;
   else
      return n*fat(n-1);
}
```

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PARÁ

www.ricardojcsouza.com.br
ricardo.souza@ifpa.edu.br

FUNÇÕES

- **PRÉ-PROCESSADOR**
  - O pré-processador de C reconhece determinadas diretivas e altera o código para, então, enviá-lo ao compilador
  - Uma das diretivas reconhecidas pelo pré-processador: **#include**
  - É seguida por um nome de arquivo e o pré-processador a substitui pelo corpo do arquivo especificado
  - Quando o nome do arquivo a ser incluído é envolto por aspas ("*arquivo*"), o pré-processador procura primeiro o arquivo no diretório atual
  - Se o arquivo é colocado entre os sinais de menor e maior (<*arquivo*>), o pré-processador procura nos diretórios de ***include*** especificados para compilação

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PARÁ

www.ricardojcsouza.com.br
ricardo.souza@ifpa.edu.br

FUNÇÕES

- **PRÉ-PROCESSADOR**
  - O pré-processador de C reconhece determinadas diretivas e altera o código para, então, enviá-lo ao compilador
  - Outra diretiva reconhecidas pelo pré-processador: **#define**
  - Diretiva de definição para representarmos constantes simbólicas

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PARÁ

www.ricardojcsouza.com.br
ricardo.souza@ifpa.edu.br

FUNÇÕES

- **PRÉ-PROCESSADOR**
  - Exemplo diretiva #define

```
#define PI  3.14159

float area (float r)
{
   float a = PI * r * r;
   return a;
}
```

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PARÁ

www.ricardojcsouza.com.br
ricardo.souza@ifpa.edu.br

Estrutura de Dados

- **REFERÊNCIAS**
- Tenenbaum, Aaron M. Langsam, Yedidyah, Augenstein, Moshe J. **Estruturas de dados usando C**. São Paulo : MAKRON *Books,* 1995.
- Veloso, Paulo. et. al. **Estrutura de dados**. Rio de Janeiro: Campus, 1986.
- Moraes, Celso Roberto. **Estrutura de dados e algoritmos**. 2. ed. São Paulo: Futura, 2003.
- Celes, W. Rangel, J. L. **Curso de Estrutura de Dados**. PUC-Rio, 2002.
- W. Celes, R. Cerqueira, J.L. Rangel. **Introdução a Estruturas de Dados - com técnicas de programação em C**. Rio de Janeiro: Campus, 2004.