Sistema Fiep
FIEP SESI SENAI IEL

Design de Estruturas de Busca e Armazenamento

Prof. Elton Dal Bem Galvão

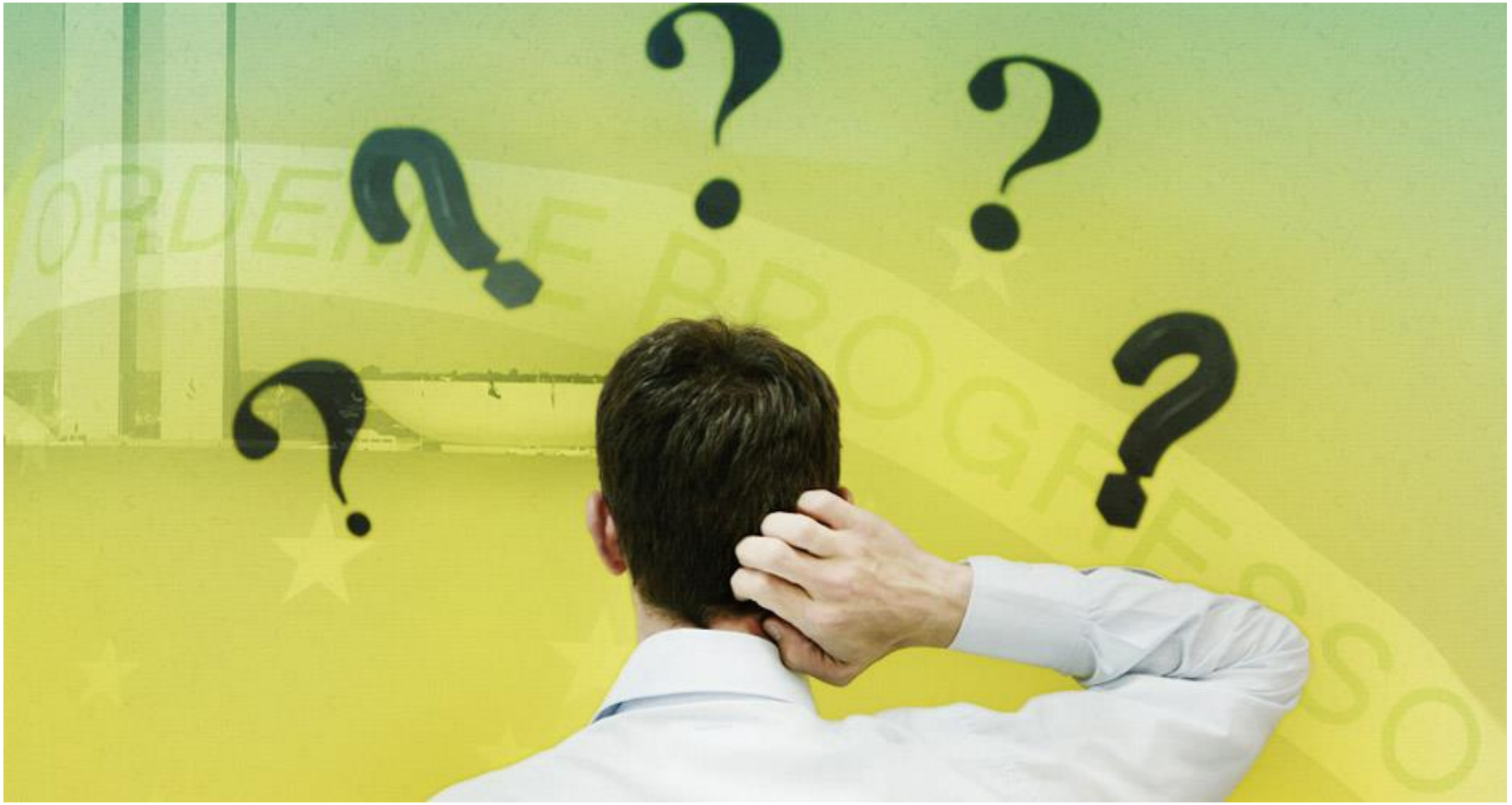
Agenda

- ✓ Apresentação
- ✓ Nossas aulas
- ✓ Ementa
- ✓ Estrutura da disciplina
- ✓ Revisão importante
- ✓ Exercícios

Ementa:

Conteúdo formativo:

O que você já sabe?



Design de Estruturas de Busca e Armazenamento

O que você já sabe?

Modelo de um Computador

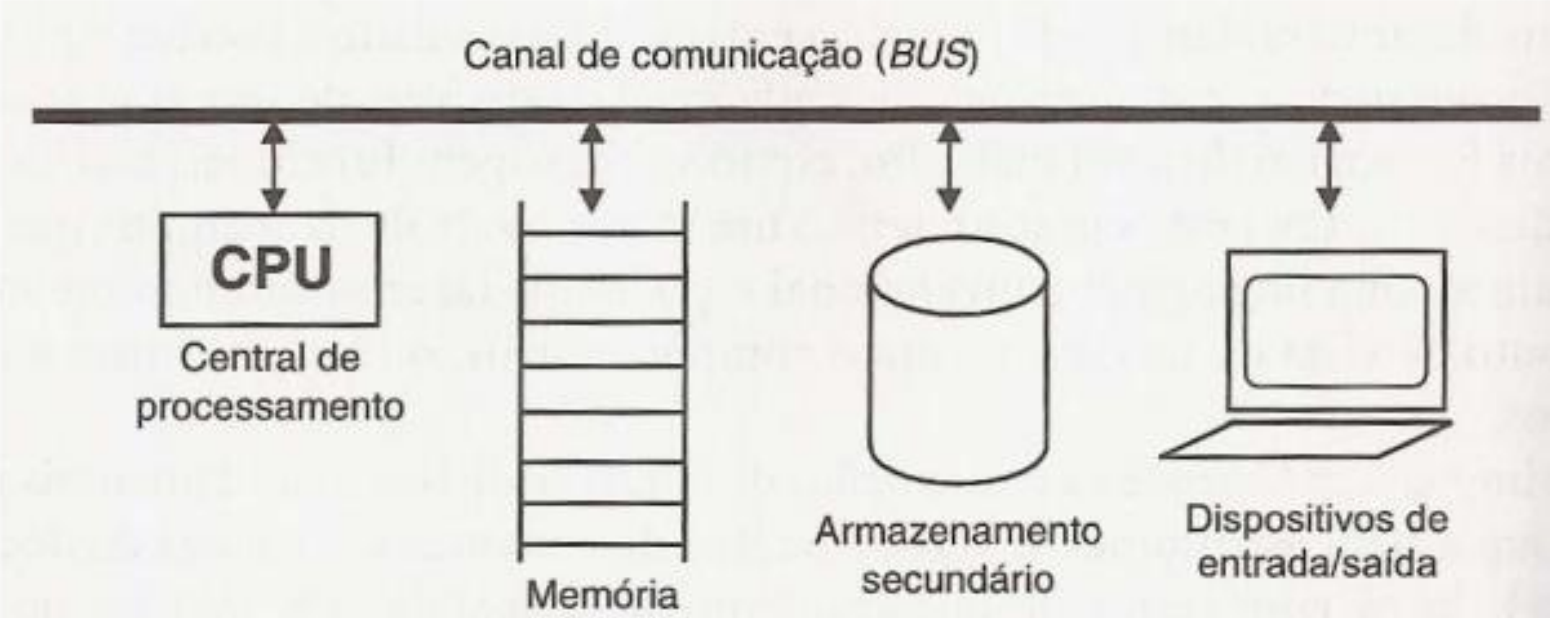


Figura 1.1 *Elementos básicos de um computador típico.*

Design de Estruturas de Busca e Armazenamento

O que você já sabe?

Armazenamento de Dados

- Organização da memória:
- Bits
  - Menor unidade
  - Valores 0 ou 1
- Bytes
  - Sequência de 8 bits
- Célula
  - Sequência de bits
  - Menor unidade de endereçamento
- Palavras
  - Sequência de bytes
  - Varia conforme arquitetura

		0	1	2	3	4	5	6	7
1	0	0	1	1	1	0	0	1	0
	1	1	1	0	0	1	1	1	0
	2	0	1	1	1	0	0	1	0
	3	0	0	0	0	0	0	0	0
2	0	1	1	1	0	1	0	1	0
	1	0	0	0	0	0	0	0	0
	2	1	1	1	1	1	1	1	1
	3	0	0	0	0	0	0	0	0

Sistema Fiep FIEP SESI SENAI IEL

Design de Estruturas de Busca e Armazenamento

O que você já sabe?

Armazenamento de Dados

- Espaço de memória finito

– Ex., espaço de 1 byte (8 bits), podemos representar apenas 2^8 (=256) valores distintos

- Podemos representar texto

– Associando valores aos caracteres

– Ex., A(65) e B (66)

- Representar um programa na memória

– Programas são executados em linguagem de máquina

– Programas executáveis são sequencias de instruções (códigos numéricos)

- 10110000 01100001 (MOV AL, 61h)

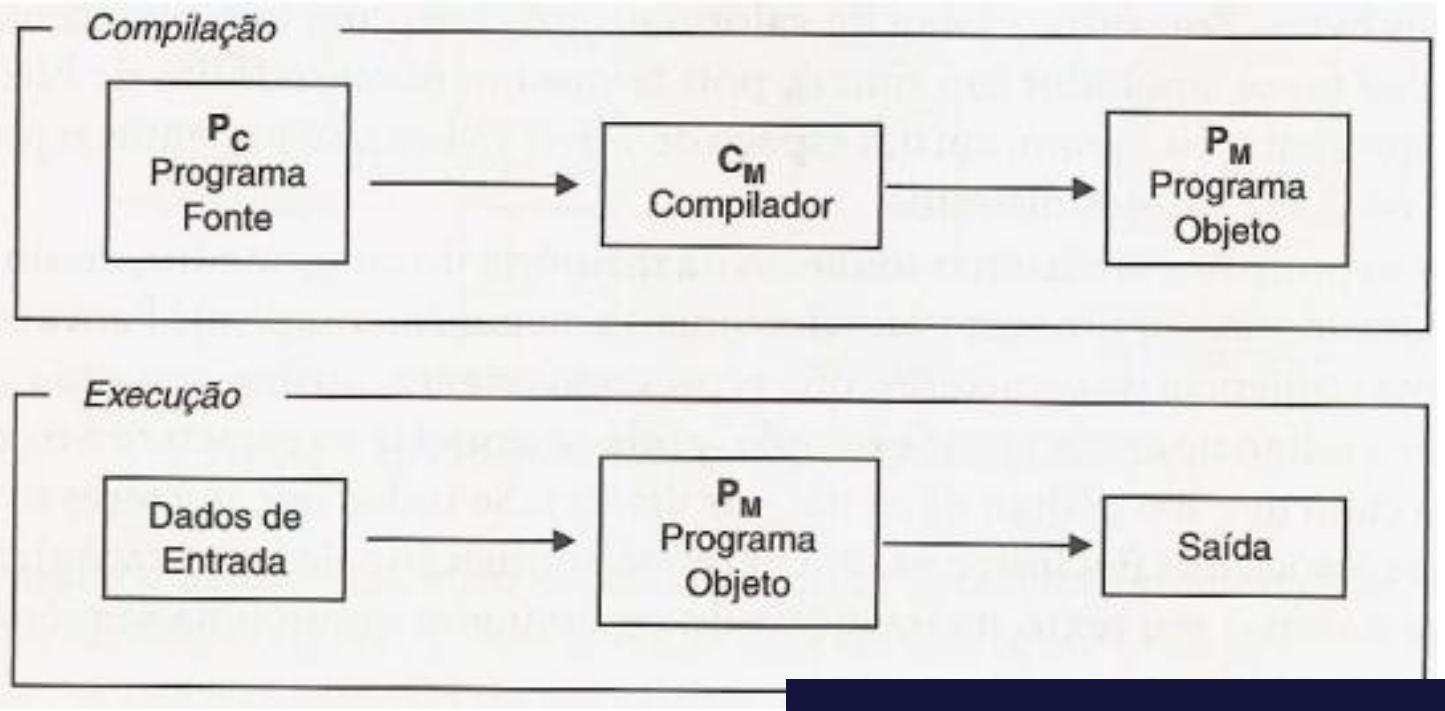
Design de Estruturas de Busca e Armazenamento

O que você já sabe?

Compilação de Programas

• Compilação: “tradução” de código fonte (Pc) para linguagem de máquina (M);

• Compilador (Cm), escrito em M: lê o programa Pc e traduz cada instrução para M, escrevendo o programa objeto (Pm).



Design de Estruturas de Busca e Armazenamento

O que você já sabe?

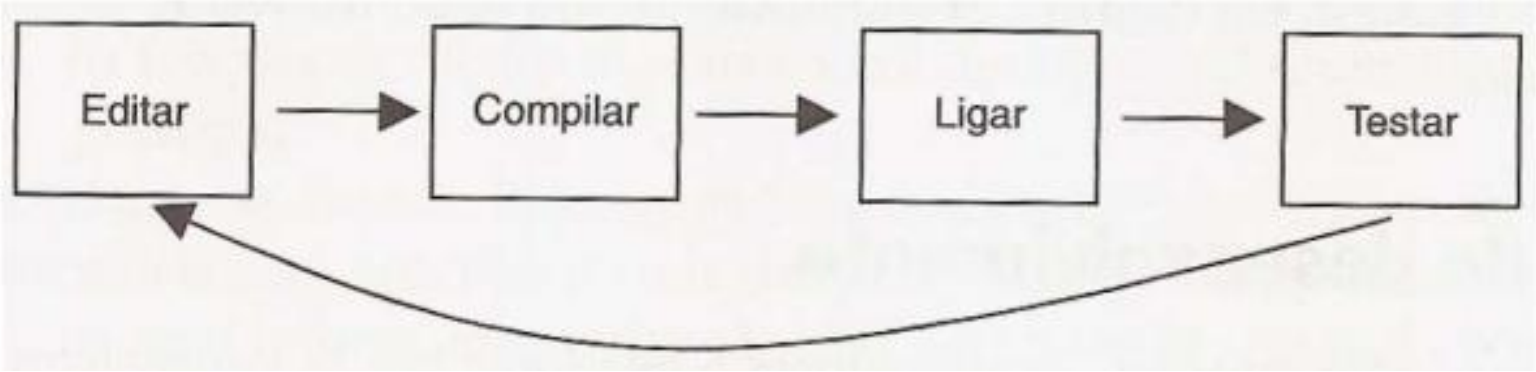
Ciclo de Desenvolvimento

- Programas em C geralmente são divididos em vários arquivos;
- Cada arquivo pode ser compilado separadamente;
- Para gerar um executável, precisamos reunir os códigos dos arquivos separados (juntamente com as bibliotecas usadas) : ***Ligador***
- Bibliotecas: permitem que funções de interesse geral sejam usadas por vários programas;
- O ligador pode ser usado automaticamente pelo compilador (biblioteca padrão), ou deve ser explicitamente acionado.

Design de Estruturas de Busca e Armazenamento

O que você já sabe?

Ciclo de Desenvolvimento (cont.)



Design de Estruturas de Busca e Armazenamento

O que você já sabe?

Tipos de Dados

- Definição

Um tipo de dado nada mais que é algo do mundo real que pode ser representado computacionalmente. Por exemplo, os números que pertencem ao conjunto dos números inteiros, os números que pertencem ao conjunto dos números reais, letras, caracteres especiais, acentuação, pontuação, palavras, etc.

Design de Estruturas de Busca e Armazenamento

O que você já sabe?

Tipos de Dados

- O objetivo principal de qualquer computador é a resolução de problemas através da manipulação de dados, que podem ser de vários tipos.

- Em computação, dado é algo que pode ser representado, armazenado e manipulado.

Design de Estruturas de Busca e Armazenamento

O que você já sabe?

Tipos de Dados

- Os dados podem ser de tipo primitivo ou derivado

- Tipos primitivos: tipos de dados básicos utilizados na construção de algoritmos. São utilizados no algoritmo, porque já vem definido na linguagem. A maioria das linguagens utilizam os quatro tipos básico de dados.

- Tipos derivados: tipos de dados que derivam dos tipos de dados básicos (primitivos).

Design de Estruturas de Busca e Armazenamento

O que você já sabe?

Tipos de Dados

Tipo	Tamanho	Menor valor	Maior valor
char	1 byte	-128	+127
unsigned char	1 byte	0	+255
short int (short)	2 bytes	-32.768	+32.767
unsigned short int	2 bytes	0	+65.535
int (*)	4 bytes	-2.147.483.648	+2.147.483.647
long int (long)	4 bytes	-2.147.483.648	+2.147.483.647
unsigned long int	4 bytes	0	+4.294.967.295
float	4 bytes	-10^{38}	$+10^{38}$
double	8 bytes	-10^{308}	$+10^{308}$

Sistema Fiep FIEP SESI SENAI IEL

Design de Estruturas de Busca e Armazenamento

O que você já sabe?

Operadores de Incremento e Decremento

- ++ e --

– Incrementa ou decrementa o valor de uma variável de uma unidade

– O incremento/decremento pode ser antes ou depois da variável ser usada

- N++, incrementa n depois de ser usado
- ++N, incrementa n antes de ser usado.

```
n = 5;
x = n++;          /* x recebe 5; n é incrementada para 6 */
x = ++n;          /* n é incrementada para 6; x recebe 6 */
a = 3;
b = a++ * 2;      / b termina com o valor 6 e a com o valor 4 */
```

O que você já sabe?

Entrada e saída

- São feitas com uso de funções
- Função printf
  - *printf (formato, lista de constantes/variáveis/expr...);*

`%c`	*especifica um char*
`%d`	*especifica um int*
`%u`	*especifica um unsigned int*
`%f`	*especifica um double (ou float)*
`%e`	*especifica um double (ou float) no formato científico*
`%g`	*especifica um double (ou float) no formato mais apropriado (%f ou %e)*
`%s`	*especifica uma cadeia de caracteres*

O que você já sabe?

Entrada e saída

```
printf ("Inteiro = %d Real = %g", 33, 5.3);
```

com saída:

```
Inteiro = 33 Real = 5.3
```

- Caracteres de escape

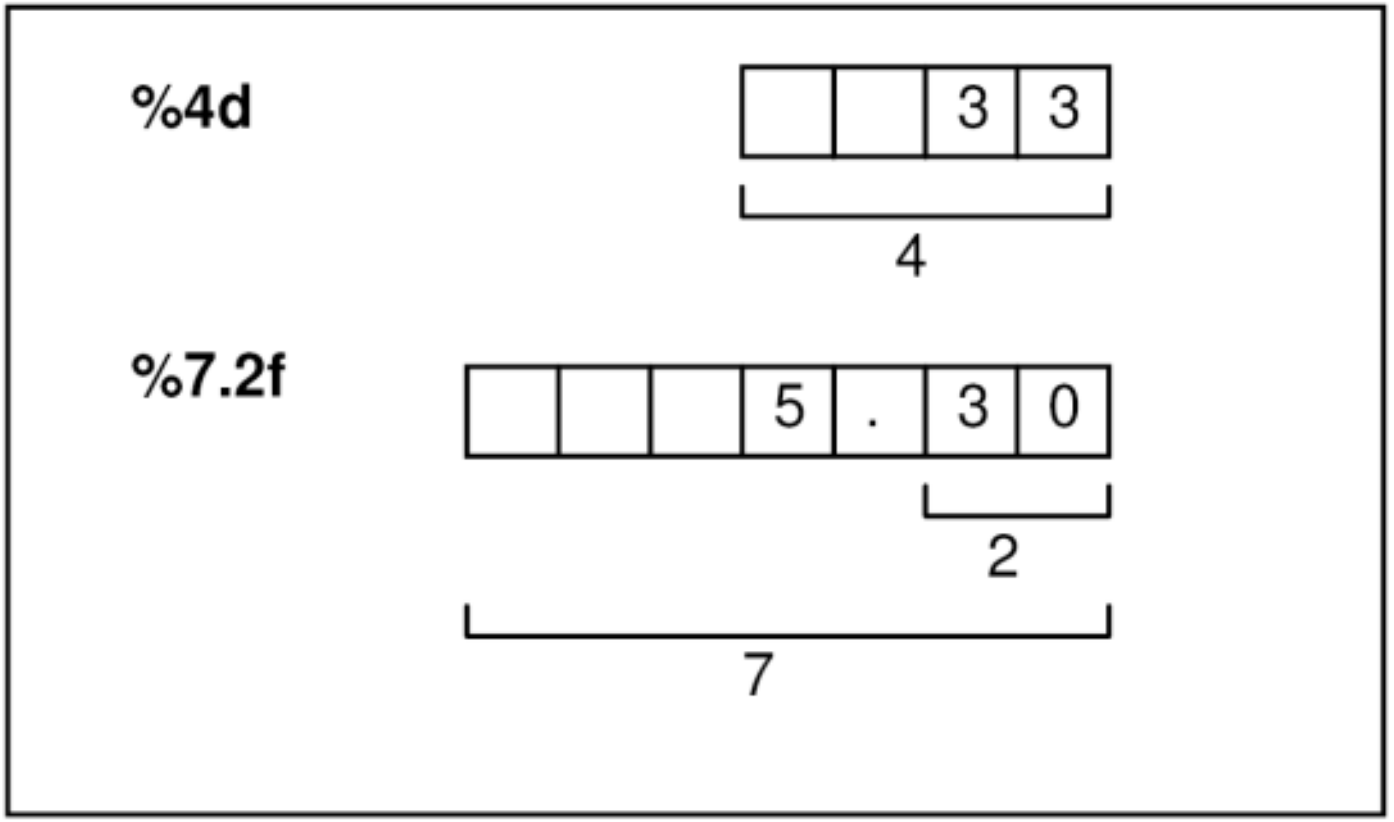
`\n`	*caractere de nova linha*
`\t`	*caractere de tabulação*
`\r`	*caractere de retrocesso*
`\"`	*caractere “*
`\\`	*caractere *

Design de Estruturas de Busca e Armazenamento

O que você já sabe?

Entrada e saída

- Especificação do tamanho do campo



O que você já sabe?

Entrada e saída

- scanf (formato, lista de endereços das variáveis...)

```
int n;
scanf ("%d", &n);
```

`%c`	*especifica um char*
`%d`	*especifica um int*
`%u`	*especifica um unsigned int*
`%f, %e, %g`	*especificam um float*
`%lf, %le, %lg`	*especificam um double*
`%s`	*especifica uma cadeia de caracteres*

O que você já sabe?

Funções

- Comando de definição de uma função

```
Tipo_retorno nome_funcao (lista de parametros)
{
    Corpo da função
}
```

Design de Estruturas de Busca e Armazenamento

O que você já sabe?

Definição de Funções

```
/* 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

Sistema Fiep FIEP SESI SENAI IEL

Design de Estruturas de Busca e Armazenamento

O que você já sabe?

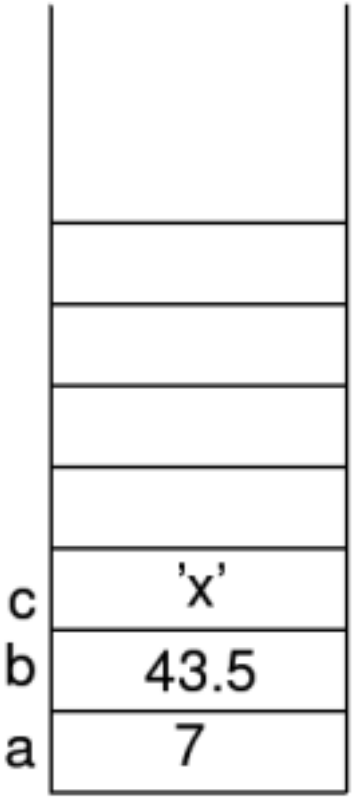
Pilha de Execução

- Variáveis locais têm escopo local
- Funções são independentes entre si
- Transferência de dados entre funções através de
  - Passagem de parâmetros
  - Valor de retorno
- Parâmetros em C são passados por valor
- Pilha de Execução: Coordena comunicação entre a função que chama e a função chamada
  - Permite passagem de parâmetros e valores de retorno

Design de Estruturas de Busca e Armazenamento

O que você já sabe?

Esquema representativo da memória



c	'x'	112	- variável c no endereço 112 com valor igual a 'x'
b	43.5	108	- variável b no endereço 108 com valor igual a 43.5
a	7	104	- variável a no endereço 104 com valor igual a 7

O que você já sabe?

Exemplo fat (5)

```
/* 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*

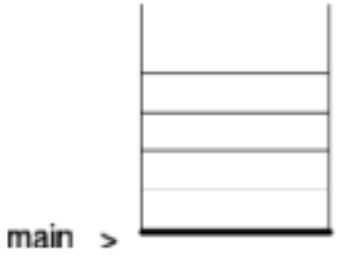
alteração no valor de *n* em *fat* não altera o valor de *n* em *main*

Design de Estruturas de Busca e Armazenamento

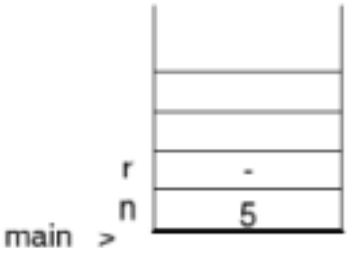
O que você já sabe?

Pilha de execução

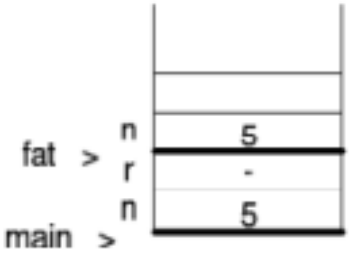
1 - Início do programa: pilha vazia



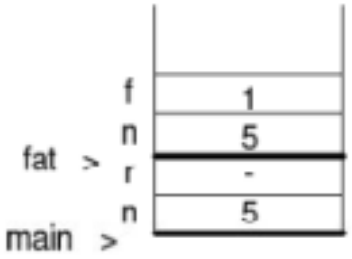
2 - Declaração das variáveis: n, r



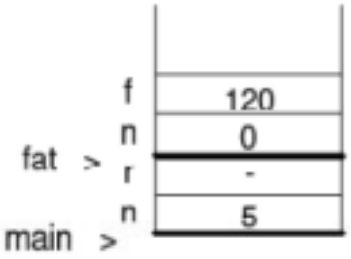
3 – Chamada da função : cópia do parâmetro



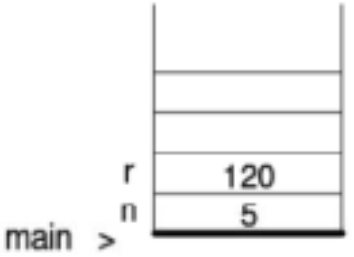
4 - Declaração da variável local: f



5 - Final do laço



6 - Retorno da função: desempilha



O que você já sabe?

Ponteiro de variáveis

- Pode ser necessário comunicar mais de um valor de retorno para função que chama
- Por exemplo, uma função que deve calcular a soma e o produto de dois números

```
#include <stdio.h>
void somaprod (int a, int b, int c, int d)
{  c = a + b;
   d = a * b;
}

int main (void)
{
    int s, p;
    somaprod (3, 5, s, p);
    printf ("soma = %d produto =%d\n", s, p);
    return 0;
}
```

O que você já sabe?

Ponteiros

- Permitem manipulação direta de endereços de memória no C
- Variáveis do tipo ponteiro
  - Armazenam endereços de memória
  - É possível definir um ponteiro para cada tipo do C que seja capaz de armazenar endereços de memória em que existem valores do tipo correpondente
  - int a;
  - int* p; // p armazena endereço de memória em que há valor inteiro

O que você já sabe?

Operadores de ponteiros

- Operador & (“endereço de”)
  - Aplicado a variáveis, retorna o endereço da posição de memória reservada para variável

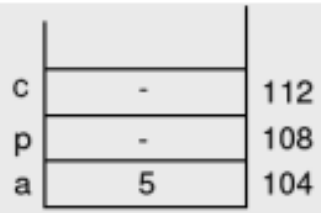
- Operador * (“conteúdo de”)
  - Aplicado a ponteiros, acessa o conteúdo de memória do endereço armazenado pela variável ponteiro

O que você já sabe?

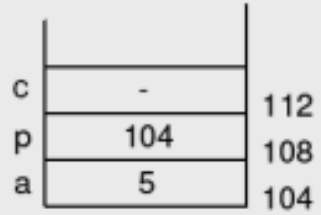
Exemplo

- int a; int* p; int c;

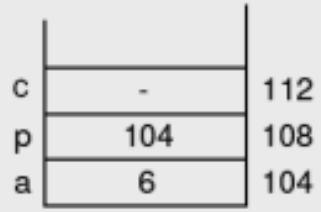
```
/ * a recebe o valor 5 */
a = 5;
```



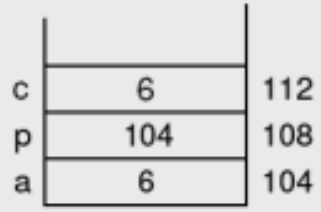

```
/* p recebe o endereço de a
   ou seja, p aponta para a */
p = &a;
```




```
/* posição de memória apontada por p
   recebe 6 */
*p = 6;
```




```
/* c recebe o valor armazenado
   na posição de memória apontada por p */
c = *p;
```



O que você já sabe?

Exemplos

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

Imprime o valor 2

Exemplos

```
int main (void)
{
    int a, b, *p;
    a = 2;
    *p = 3;
    b = a + (*p);
    printf (" %d ", b);
    return 0;
}
```

ERRO!

O que você já sabe?

Passando ponteiros para função

- Ponteiros permitem modificar o valor das variáveis indiretamente
- Possível solução para passagem por ref!

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

int main (void)
{
        int s, p;
        somaprod (3, 5, &s, &p);
        printf ("soma = %d produto =%d\n", s, p);
        return 0;
}
```

O que você já sabe?

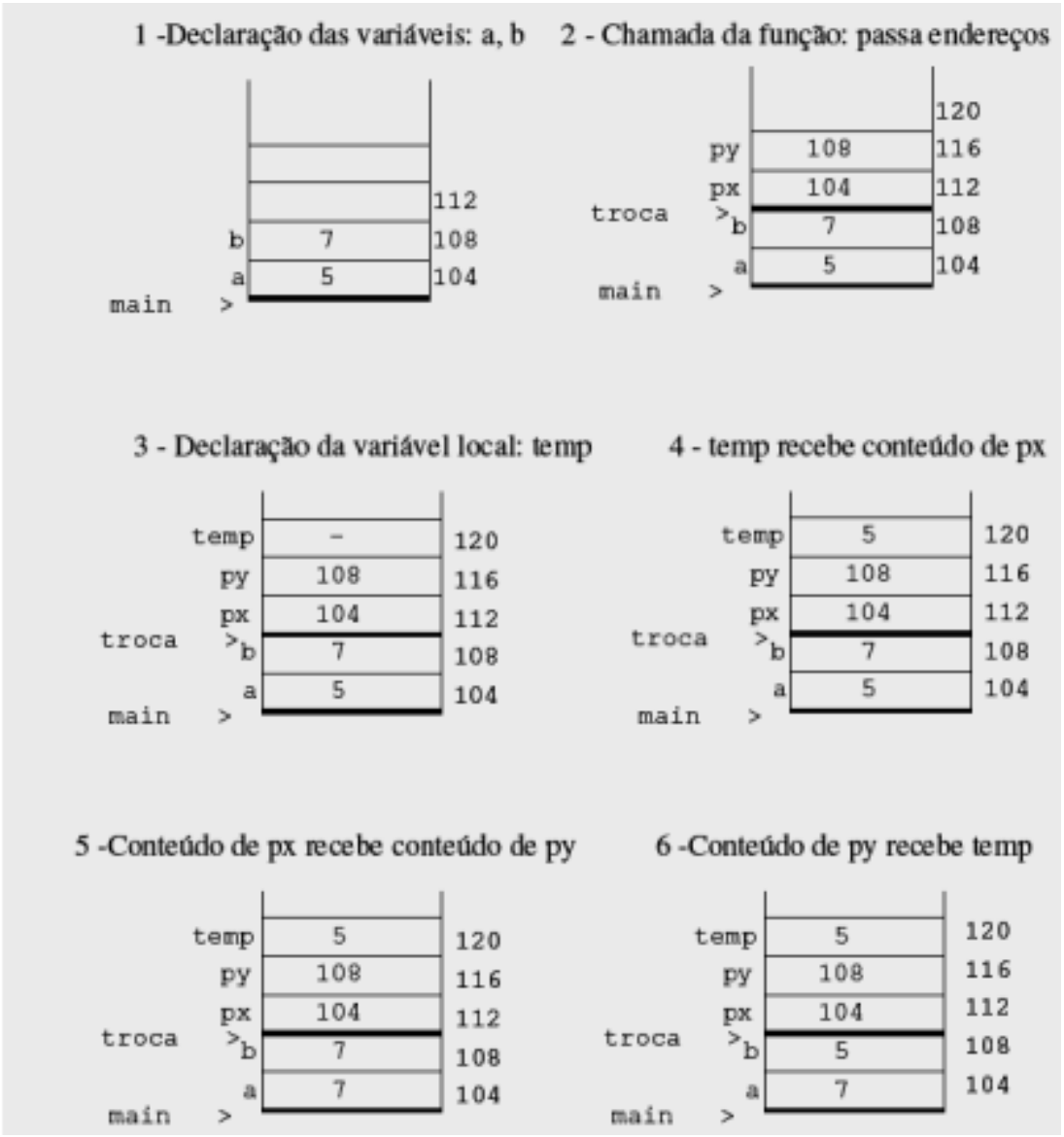
Exemplo

```
/* função troca */
#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;
}
```

Design de Estruturas de Busca e Armazenamento

O que você já sabe?

Exemplo



O que você já sabe?

Variáveis Globais

- Declaradas fora do escopo das funções
- São visíveis a todas as funções
- Existem enquanto o programa existir (não estão na pilha de execução)
- Utilização:
  - Devem ser usadas com critério
  - Podem criar muita dependência entre as funções
  - Dificulta o entendimento e o reuso de código

O que você já sabe?

Exemplo de 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;
}
```

Design de Estruturas de Busca e Armazenamento

O que você já sabe?

Variáveis Estáticas

- Declaradas no escopo de funções
- Existem enquanto o programa existir (não estão na pilha de execução)
- Somente são visíveis dentro das funções nas quais são declaradasa
- Utilização
  - Quando for necessário recuperar o valor de uma variável na execução passada da função

O que você já sabe?

Exemplo de variável estática

- Função que imprime números reais
  - Imprime um número por vez (máximo de 5 números por linha)

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

O que você já sabe?

Sobre variáveis estáticas e globais...

- Variáveis estáticas e globais são inicializadas com zero, quando não forem explicitamente inicializadas
- Variáveis globais estáticas
  - São visíveis para todas funções subsequentes
  - Não podem ser acessadas por funções de outros arquivos
- Funções estáticas
  - Não podem ser acessadas por funções de outros arquivos

O que você já sabe?

Pré-processador e Macros

- Código C antes de ser compilado é passado pelo pré-processador
- O Pré-processador
  - Reconhece diretivas
  - Altera o código e envia para o compilador
- Diretiva #include
  - O pré-processador substitui pelo corpo do arquivo especificado

O que você já sabe?

Pré-processador e Macros

- # include "nome_do_arquivo"
  - Procura o arquivo do diretório local
  - Caso não encontre, procura nos diretórios de include especificados para compilação
- # include <nome_do_arquivo>
  - Não procura no diretório local


Sistema Fiep
FIEP
SESI
SENAI
IEL
OBRIGADO
eltondalbem@gmail.com