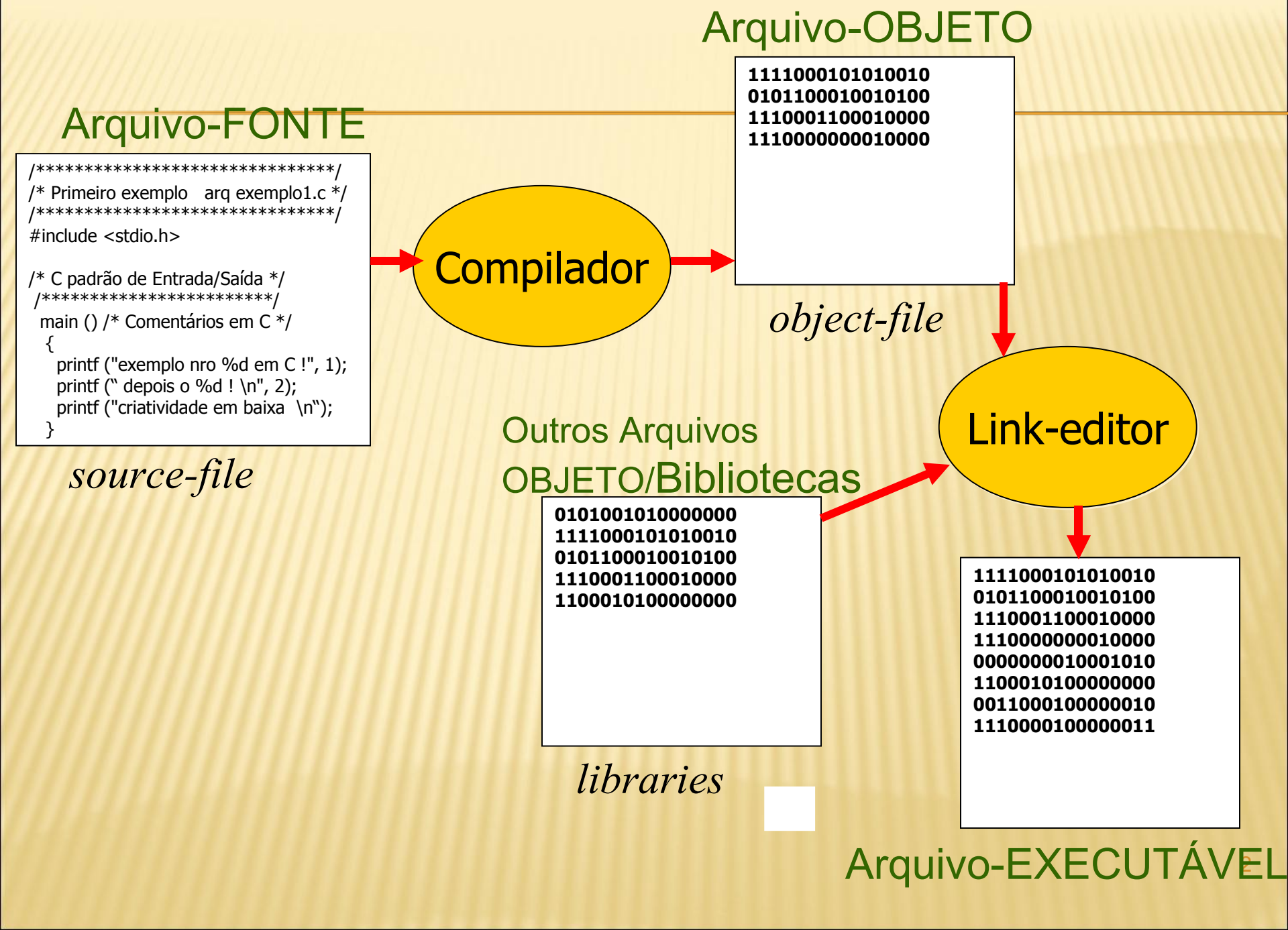
Universidade de São Paulo – São Carlos
Instituto de Ciências Matemáticas e de Computação

VARIÁVEIS EM C E COMANDOS DE LEITURA ESCRITA

Profa Rosana Braga

1º semestre de 2010


Arquivo-OBJETO
1111000101010010
0101100010010100
1110001100010000
1110000000010000
Arquivo-FONTE
/*******************************/
/* Primeiro exemplo arq exemplo1.c */
/*******************************/
#include <stdio.h>
/* C padrão de Entrada/Saída */
/*************************/
main () /* Comentários em C */
{
printf ("exemplo nro %d em C !", 1);
printf (" depois o %d ! \n", 2);
printf ("criatividade em baixa \n");
}
Compilador
object-file
source-file
Link-editor
Outros Arquivos
OBJETO/Bibliotecas
0101001010000000
1111000101010010
0101100010010100
1110001100010000
1100010100000000
libraries
1111000101010010
0101100010010100
1110001100010000
1110000000010000
0000000010001010
1100010100000000
0011000100000010
1110000100000011
Arquivo-EXECUTÁVEL

ESTRUTURA DE UM PROGRAMA C

Programa C

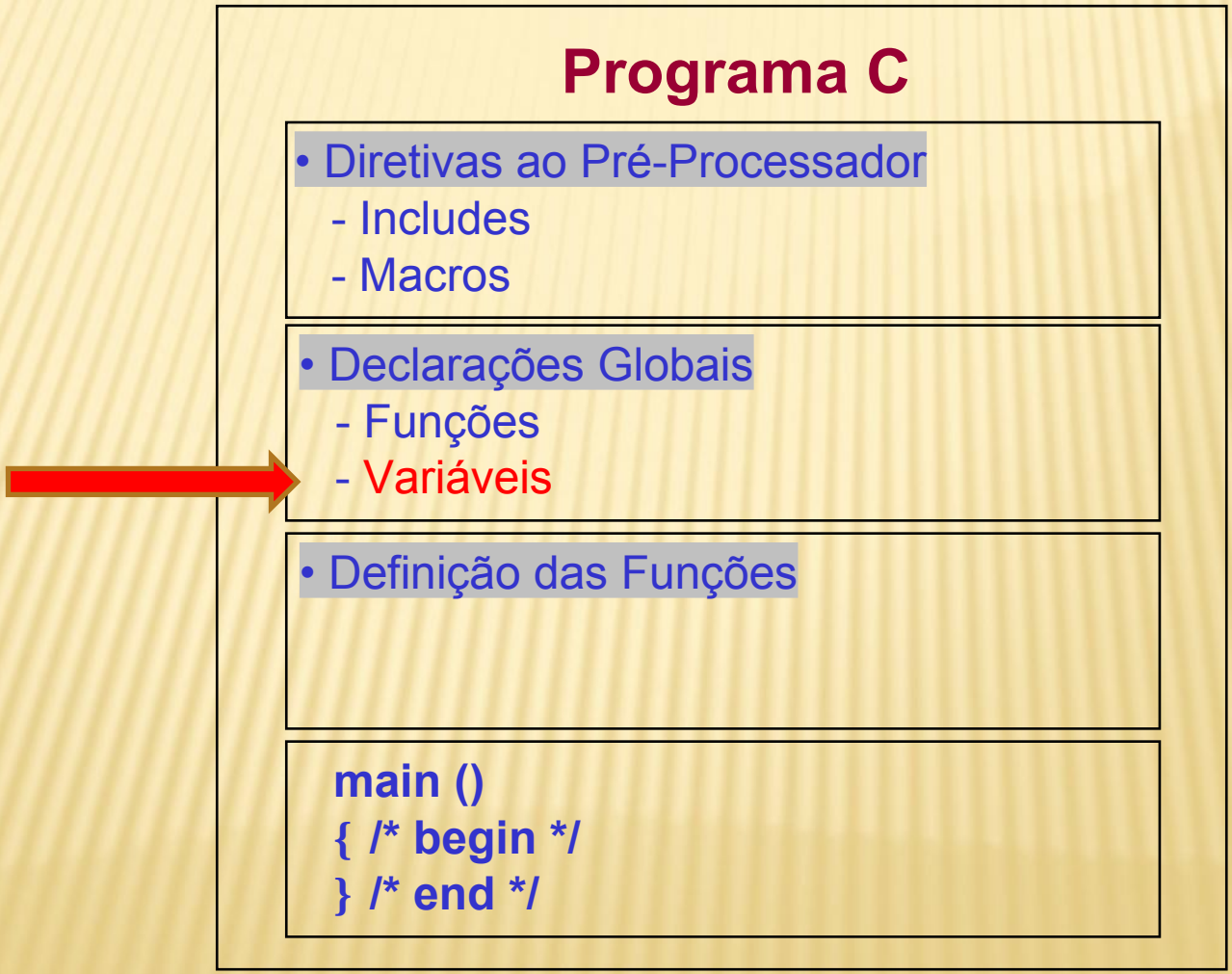
- Diretivas ao Pré-Processador
  - Includes
  - Macros

- Declarações Globais
  - Funções
  - Variáveis

- Definição das Funções

```
main ()
{ /* begin */
} /* end */
```

ESTRUTURA DE UM PROGRAMA C



VARIÁVEIS

- Variáveis em um programa C estão associadas a posições de memória que armazenam informações
- Nomes de variáveis podem ter vários caracteres
- Em C, apenas os 31 primeiros caracteres são considerados
- O primeiro caracter tem que ser uma letra ou *underscore* "_"
- O restante do nome pode conter letras, dígitos e sublinhados

VARIÁVEIS

- Exemplos de nomes de variáveis:

Corretos	Incorretos
Contador	1contador
Teste23	oi!gente
Alto_Paraiso	Alto..Paraíso
__sizeint	_size-int

- Palavras-chave de C não podem ser utilizadas como nome de variáveis: ***int***, **main**, ***for***, ***while***, **if**, etc...

- C é *case-sensitive*:

  contador ≠ Contador ≠ CONTADOR

TIPOS DE DADOS BÁSICOS EM C

- char: um byte que armazena o código de um caracter do conjunto de caracteres local
- int: um inteiro cujo tamanho depende do processador, tipicamente 16 ou 32 bits
- float: um número real com precisão simples
- double: um número real com precisão dupla

MODIFICADORES DE TIPOS

- modificadores alteram algumas características dos tipos básicos para adequá-los a necessidades específicas

- Modificadores:
  - **signed**: indica número com sinal (inteiros e caracteres)
  - **unsigned**: número apenas positivo (inteiros e caracteres)
  - **long**: aumenta abrangência (inteiros e reais)
  - **short**: reduz a abrangência (inteiros)

ABRANGÊNCIA DE DADOS: 16 BITS

Tipo	Tamanho(bytes)	Abrangência	
char	1	-128 a	127
unsigned char	1	0 a	255
int	2	-32768 a	32767
unsigned int	2	0 a	65535
short int	2	-32768 a	32767
long int	4	-2.147.483.648 a	2.147.483.647
unsigned long	4	0 a	4.294.967.295
float	4	$-3,4 \cdot 10^{38}$ a	$3,4 \cdot 10^{38}$
double	8	$-1,7 \cdot 10^{308}$ a	$1,7 \cdot 10^{-308}$
long double	10	$-3,4 \cdot 10^{4932}$ a	$3,4 \cdot 10^{4932}$

ABRANGÊNCIA DE DADOS: 32 BITS

Tipo	Tamanho(bytes)	Abrangência
char	1	-128 a 127
unsigned char	1	0 a 255
int	4	-2.147.483.648 a 2.147.483.647
unsigned int	4	0 a 4.294.967.295
short int	2	-32768 a 32767
long int	4	-2.147.483.648 a 2.147.483.647
unsigned long	4	0 a 4.294.967.295
float	4	$3,4 \cdot 10^{-38}$ a $3,4 \cdot 10^{38}$
double	8	$1,7 \cdot 10^{-308}$ a $1,7 \cdot 10^{-308}$
long double	10	$3,4 \cdot 10^{-4932}$ a $3,4 \cdot 10^{4932}$

CONSTANTES

- Constantes são valores fixos que não podem ser modificados pelo programa:

Tipo		Exemplos
char	->	‘a’ ‘\n’ ‘9’
int	->	123 1 1000 -23
long int	->	35000L -45L
short int	->	10 -12 90
unsigned int	->	1000U 234U 4365U
float	->	123.45F 3.1415e-10F
double	->	123.45 -0.91254

CONSTANTES DO TIPO CHAR

- Barra invertida
- \a bip
- \b backspace
- \n newline
- \t tab horizontal
- \’ apóstrofe
- \” aspa
- \\ backslash
- \f form feed

EXEMPLO

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#define FALSE 0 /* define  F  igual 0 */
#define TRUE 1 /* define  T igual 1 */

int i = 35;

void main( ) {
  int resposta;
  printf( "Quer ver a mensagem?\n" );
  scanf( "%d", &resposta );
  if( resposta == TRUE )
      printf( "Isso foi um Teste ! O valor de i eh %d \n ", i);
  else
      printf( "Goodbye for now.\n" );
  system("pause");
}
```

EXEMPLO - EXPLICAÇÃO

```
#include   <stdio.h>
#include   <stdlib.h>
#include   <conio.h>
#define    FALSE 0      /* define macro F igual 0 */
#define    TRUE 1       /* define macro T igual 1 */
```

Diretivas ao Pré-processador

- **#include** <filename>
- indica ao pré-processador para incluir o arquivo filename antes da compilação
- os arquivos terminados em ".h" são chamados *headers* (ou cabeçalhos). Armazenam informações globais como declaração de tipos, funções e definição de macros

EXEMPLO - EXPLICAÇÃO

- **#define** FALSE 0
- define uma macro, que permite substituir um *string* por outro valor no corpo do programa *antes* da compilação se realizar
- no exemplo acima o pré-processador substitui as ocorrências de FALSE por 0 e TRUE por 1
- Ex:
  - **if** (resposta == TRUE) ==> **if** (resposta == 1)

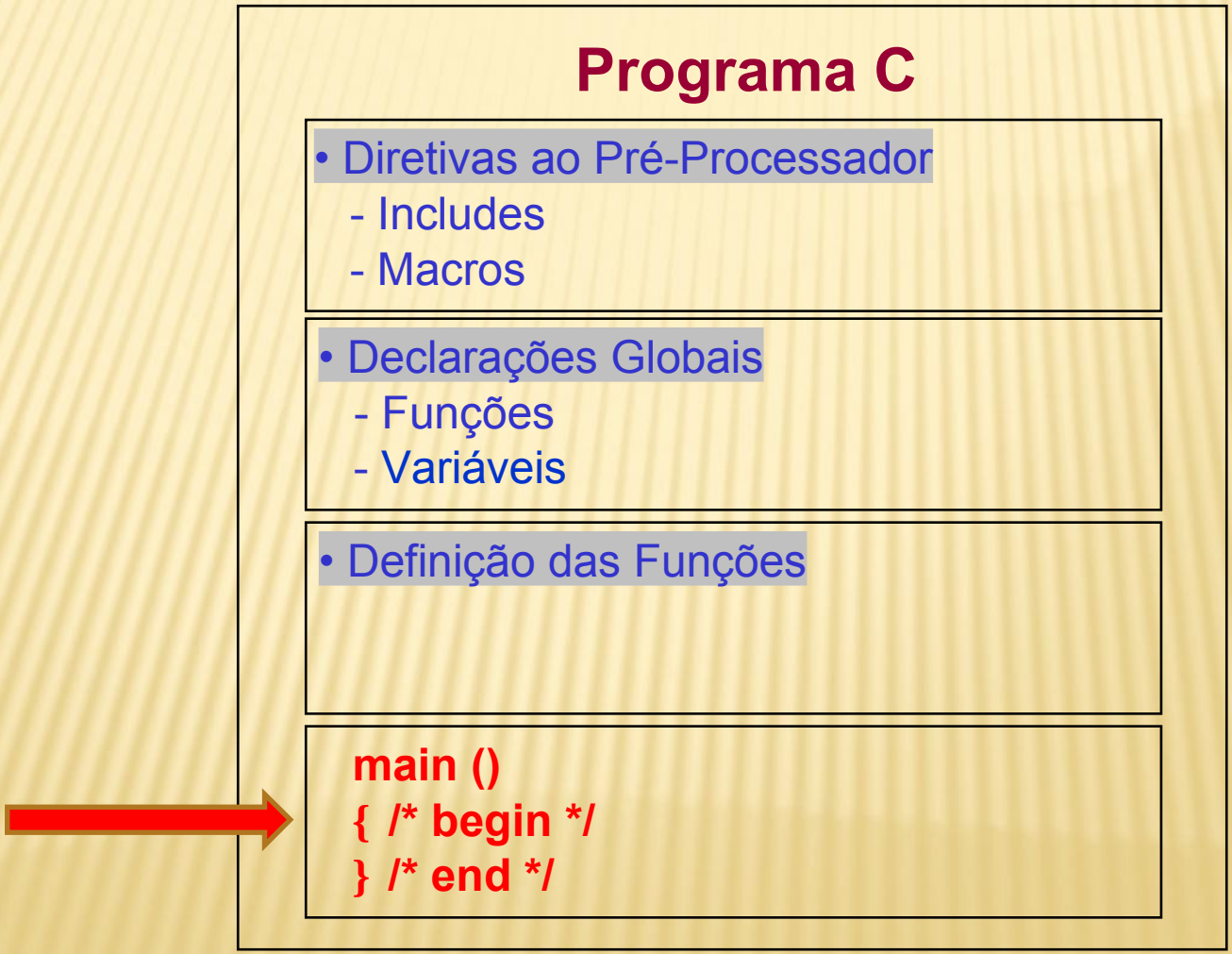
EXEMPLO - EXPLICAÇÃO

```
int i = 35;
```

Declarações Globais

- indica ao compilador que, *dentro do arquivo* onde aparecem estas declarações:
  - a variável *i* é inteira, iniciada com o valor 35

ESTRUTURA DE UM PROGRAMA C



EXEMPLO - EXPLICAÇÃO

```
void main( ) {
  int resposta;
  printf( "Quer ver a mensagem?\n" );
  scanf( "%d", &resposta );
  if( resposta == TRUE ) printf( "Isso foi
um Teste ! O valor de i eh %d ", i);
  else printf( "Goodbye for now." );
}
```

Função Principal

- *todo* programa C tem que ter uma função chamada **main**( ). É aqui que inicia a execução do programa
- em um programa pequeno, todo o algoritmo pode ser escrito em *main()*
- programas estruturados consistem em uma hierarquia de funções dentre as quais *main()* é aquela de mais alto nível

EXEMPLO - EXPLICAÇÃO

```
void main( ) {
  int resposta;
  printf( "Quer ver a mensagem?\n" );
  scanf( "%d", &resposta );
  if( resposta == TRUE ) printf( "Isso foi
um Teste ! O valor de i eh %d ", i);
  else printf( "Goodbye for now." );
}
```

Função Principal

- *A declaração da variável resposta é uma declaração local, ao contrário da declaração da variável i, que era global.*
- *Isso significa que resposta só é visível dentro de main().*

EXEMPLO - EXPLICAÇÃO

- O comando *if*

```
if( resposta == TRUE )
    printf( "Isso foi um Teste ! O valor de i eh %d ", i);
else
    printf( "Goodbye for now." );
```

- Formato:
  - *if (<teste>)*
  - *<comando1>*
  - *else*
  - *<comando2>*

se <teste> avalia para verdadeiro, <comando1> é executado, senão <comando2> é executado

EXEMPLO - EXPLICAÇÃO

- Entrada e Saída elementar:
  - C utiliza o conceito de *streams* (canais) tanto para realizar E/S como para acessar arquivos
  - Canais pré-definidos:
    - *stdin*: associado ao teclado para entrada de dados
    - *stdout*: associado a tela para exibição de dados
    - *stderr*: mensagens de erro, enviadas a tela por *default*

EXEMPLO - EXPLICAÇÃO

- Entrada formatada SCANF():
  - *scanf()* lê um string de entrada, converte os dados de acordo com a especificação de formato e armazena-os nas variáveis indicadas
  - Formato:

    scanf("<formato e texto>", endereço_variáveis);
    - para se armazenar os valores lidos, são passados os endereços das variáveis, de forma que scanf saiba onde colocar os dados

EXEMPLO - EXPLICAÇÃO

- Entrada formatada SCANF(). Exemplo:

  leitura de um inteiro do teclado:

```
#include <stdio.h>
void main()
{
    int i;
    scanf("%d", &i);
}
```

- o operador "&" localiza a variável *i* para *scanf*
- "%d" indica que o dado a ser lido é um inteiro

EXEMPLO - EXPLICAÇÃO

- Saída formatada PRINTF():
  - *printf()* escreve um conjunto de dados na saída, convertendo os dados de acordo com a especificação de formato. Note que, ao contrário de scanf, os valores das variáveis são fornecidos
  - Formato:

    printf(“<formato e texto>”, variáveis);

EXEMPLO - EXPLICAÇÃO

- Saída formatada PRINTF(). Ex:

```
int i = 10;
float r = 3.1514;
char s[] = “Blablabla”;  /* cadeia de caracteres */
printf(“Inteiro: %d, Real: %f, String: %s”,i,r,s);
```

produz:

Inteiro: 10, Real: 3.151400, String: Blablabla

DOCUMENTAÇÃO

- O código fonte pode ser documentado com comentários explicativos
- // é usado para um comentário que ocupa a linha toda
- /* comentario */ quando o comentário é somente em parte da linha
- Usar também identação nos comandos condicionais e de repetição
- Algoritmos em alto nível devem acompanhar o código

DOCUMENTAÇÃO: EXEMPLO

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#define FALSE 0 /* define  F  igual 0 */
#define TRUE 1 /* define  T igual 1 */
int i = 35;  /* variavel global do sistema */
// aqui começa o programa principal
void main( ) {
    int resposta;
    printf( “Quer ver a mensagem?\n" );
    scanf( "%d", &resposta );  // a resposta pode ser 0 ou 1
    if( resposta == TRUE )
            printf( “Isso foi um Teste ! O valor de i eh %d \n “, i);
     else /* tratamento de caso raro comeca aqui */
            printf( "Goodbye for now.\n" );
  system(“pause”);
}
```

```
//MESMO CÓDIGO, SEM
      DOCUMENTACAO E IDENTACAO
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#define FALSE 0
#define TRUE 1
int i = 35;
void main( ) { int resposta; printf( “Quer
ver a mensagem?\n" );
scanf( "%d", &resposta );
if( resposta == TRUE ) printf( “Isso foi
um Teste ! O valor de i eh %d \n “, i);
else printf( "Goodbye for now.\n" );
 system(“pause”); }
```

DOCUMENTAÇÃO: ALGORITMO

```
Inicio
   Variáveis: i, resposta: inteiro
   i = 35
   Escreva “Quer ver a resposta?”
   Leia resposta
   Se resposta == 0  então
        Escreva “Isso foi um teste. O valor de i é”,i
    Senão
        Escreva “Goodbye for now”
Fim
```

EXERCÍCIOS

1. Escreva um programa em C que calcula o preço total de um produto, tendo como entrada o preço unitário e a quantidade vendida.

```
Algoritmo calculo
Inicio.
    Var real: preco, qtde, total;
    escreva("entre com o preco do produto:");
    leia(preco);
    escreva("entre com a quantidade vendida:");
    leia(qtde);
    total = preco * qtde;
    escreva("o preco total do produto e:", total);
Fim.
```

2. Escreva um programa em C para calcular o consumo médio de um automóvel (medido em km/l), dada a distância total percorrida e o volume de combustível consumido para percorre-la (em litros). Antes de calcular teste se o valor informado para litros não é zero, para evitar erro!

EXERCÍCIOS

3. Escreva um programa em C para ler 2 números inteiros a e b e imprimir se a é maior que b, a é igual a b ou a é menor que b.
4. Escreva um programa em C para ler a nota do aluno e sua porcentagem de presença. Imprima se ele foi aprovado, reprovado por nota, reprovado por freqüência, reprovado em ambas ou ficou em recuperação, considerando que:

Aprovado, se nota maior ou igual a 5 e freqüência >= 70%

Reprovado por nota, se nota < 3 e freqüência >= 70%

Reprovado por freqüência, se nota >= 5 e freqüência < 70%

Reprovado em ambas, se nota < 5 e freqüência < 70%

Recuperação, se nota entre 3 e 5 e freqüência >= 70%