

INF1007: Programação 2

0 – Revisão



Tópicos Principais

- Variáveis e Constantes
- Operadores e Expressões
- Entrada e Saída
- Tomada de Decisão
- Construção com laços
- Definição de funções
- Pilha de Execução

Variáveis e Constantes

- Tipos básicos na linguagem C:

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

(*) depende da máquina, sendo 4 bytes para arquiteturas de 32 bits

Variáveis e Constantes

- Valor Constante:
 - armazenado na memória
 - possui um tipo, indicado pela sintaxe da constante

```
123          /* constante inteira do tipo "int" */  
12.45        /* constante real do tipo "double" */  
1245e-2      /* constante real do tipo "double" */  
12.45F      /* constante real do tipo "float" */
```

Variáveis e Constantes

- Variável:
 - espaço de memória para armazenar um dado
 - não é uma variável no sentido matemático
 - possui um tipo e um nome
 - nome: identifica o espaço de memória
 - tipo: determina a natureza do dado

Variáveis e Constantes

- Declaração de variável:
 - variáveis devem ser explicitamente declaradas
 - variáveis podem ser declaradas em conjunto

```
int a;      /* declara uma variável do tipo int */  
int b;      /* declara uma variável do tipo int */  
float c;    /* declara uma variável do tipo float */  
  
int d, e;   /* declara duas variáveis do tipo int */
```

Variáveis e Constantes

- Declaração de variável:
 - variáveis só armazenam valores do mesmo tipo com que foram declaradas

```
int a;          /* declara uma variável do tipo int */
```

```
a = 4.3;
```

a armazena o valor 4! porque?

Variáveis e Constantes

- Variável com valor indefinido:
 - uma variável pode receber um valor quando é definida (inicializada), ou através de um operador de atribuição

```
int a = 5, b = 10;    /* declara e inicializa duas variáveis do tipo int */  
float c = 5.3f; /* declara e inicializa uma variável do tipo float */
```



```
int a;  
int b;  
float c;  
a=5;  
b=10;  
c=5.3f;
```

C tem muitas maneiras de fazer a mesma coisa ☹

Variáveis e Constantes

- Variável com valor indefinido:
 - uma variável deve ter um valor definido quando é utilizada

```
int a, b, c;      /* declara 3 variáveis do tipo int */  
a = 2;  
c = a + b;       /* ERRO: o que tem em b? Lixo. */
```

Operadores e Expressões

- Operadores:
 - aritméticos: + , - , * , / , %
 - atribuição: = , += , -= , *= , /= , %=
 - incremento e decremento: ++ , --
 - relacionais e lógicos: < , <= , == , >= , > , !=
 - outros

Operadores e Expressões

- Operadores aritméticos (+ , - , * , / , %):
 - operações são feitas na precisão dos operandos
 - o operando com tipo de menor expressividade é convertido para o tipo do operando com tipo de maior expressividade
 - divisão entre inteiros trunca a parte fracionária

```
int a
double b, c;
a = 3.5;      /* a recebe o valor 3 */
b = a / 2.0;  /* b recebe o valor 1.5 */
c = 1/3 + b;  /* 1/3 retorna 0 pois a operação será sobre inteiros */
              /* c recebe o valor de b */
```

Operadores e Expressões

- Operadores aritméticos (cont.):
 - o operador módulo, “%”, aplica-se a inteiros
 - precedência dos operadores: $*$, $/$, $-$, $+$

```
x % 2          /* o resultado será 0, se x for par;  
                caso contrário, será 1 */
```

```
a + b * c / d  é equivalente a      (a + ((b * c) / d))
```

Operadores e Expressões

- Operadores de atribuição :
(= , += , -= , *= , /= , %=)
 - C trata uma atribuição como uma expressão
 - a ordem é da direita para a esquerda
 - C oferece uma notação compacta para atribuições em que a mesma variável aparece dos dois lados
var **op=** expr é equivalente a var = var **op** (expr)

i += 2;	é equivalente a	i = i + 2;
x *= y + 1;	é equivalente a	x = x * (y + 1);

Operadores e Expressões

- Operadores de incremento e decremento (++ , --) :
 - incrementa ou decrementa de uma unidade o valor de uma variável
 - os operadores não se aplicam a expressões
 - o incremento pode ser antes ou depois da variável ser utilizada
- n++ incrementa n de uma unidade, depois de ser usado
- ++n incrementa n de uma unidade, 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 */
```

Operadores e Expressões

- Operadores Relacionais

< <= == >= > !=

- o resultado será 0 (FALSE) ou 1 (TRUE)

- não há valores booleanos em C

```
int a, b;  
int c = 23;  
int d = c + 4;
```

```
a = c < 20;  
b = d > c;
```

a=0 e b=1

Operadores e Expressões

- Operadores lógicos

& & | | !

```
int a, b, f;  
int c = 23;  
int d = c + 4;
```

```
a = (c < 20) || (d > c);
```

(c < 20) e (d > c) são avaliadas

```
b = (c < 20) && (d > c);
```

apenas(c < 20) é avaliada

a=1 e b=0

- a avaliação é da esquerda para a direita
- a avaliação pára quando o resultado pode ser conhecido

Operadores e Expressões

- conversão de tipo:
 - conversão de tipo é automática na avaliação de uma expressão
 - conversão de tipo pode ser requisita explicitamente

```
float f=3; /* valor 3 é convertido automaticamente para “float” */  
          /* ou seja, passa a valer 3.0F, antes de ser atribuído a f */
```

```
int g, h;          /* 3.5 é convertido (e arredondado) para “int” */  
g = (int) 3.5;     /* antes de ser atribuído à variável g */  
h = (int) 3.5 % 2  /* e antes de aplicar o operador módulo “%” */
```

Entrada e Saída: printf

- Função “printf”:
 - possibilita a saída de valores segundo um determinado formato

```
printf (formato, lista de constantes/variáveis/expressões...);
```

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

tem como resultado a impressão da linha:

33 5.3

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

com saída:

Inteiro = 33 Real = 5.3

Entrada e Saída: formato do printf

- Especificação de formato:

`%c` *especifica um **char***

`%d` *especifica um **int***

`%u` *especifica um **unsigned int***

`%f` *especifica um **double** (ou **float**)*

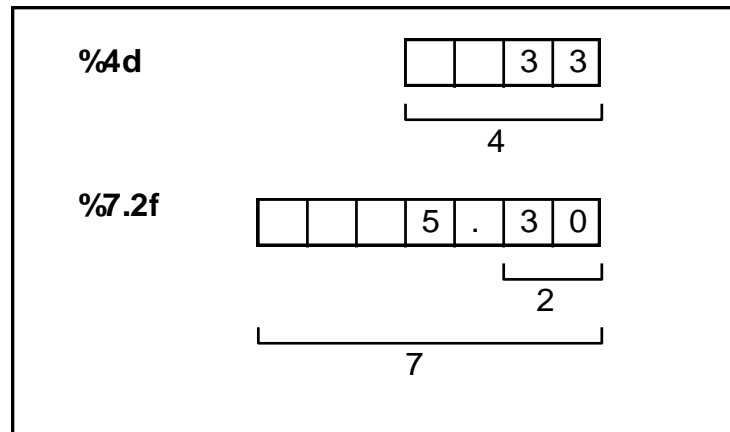
`%e` *especifica um **double** (ou **float**) no formato científico*

`%g` *especifica um **double** (ou **float**) no formato mais apropriado
(**%f** ou **%e**)*

`%s` *especifica uma cadeia de caracteres*

Entrada e Saída: ajuste de impressão

- Especificação de tamanho de campo:



Entrada e Saída: exemplo

- Impressão de texto:

```
printf("Curso de Estruturas de Dados\n");
```

exibe na tela a mensagem:

Curso de Estruturas de Dados

Entrada e Saída: scanf

- captura valores fornecidos via teclado

```
scanf (formato, lista de endereços das variáveis...);
```

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

valor inteiro digitado pelo usuário é armazenado na variável n

Entrada e Saída

- Especificação de formato:

<code>%c</code>	<i>especifica um char</i>
<code>%d</code>	<i>especifica um int</i>
<code>%u</code>	<i>especifica um unsigned int</i>
<code>%f, %e, %g</code>	<i>especificam um float</i>
<code>%lf, %le, %lg</code>	<i>especificam um double</i>
<code>%s</code>	<i>especifica uma cadeia de caracteres</i>

Entrada e Saída

- Função “scanf” (cont.):
 - caracteres diferentes dos especificadores no formato servem para cercar a entrada
 - espaço em branco dentro do formato faz com que sejam "pulados" eventuais brancos da entrada
 - %d, %f, %e e %g automaticamente pulam os brancos que precederem os valores numéricos a serem capturados

```
scanf ("%d:%d", &h, &m) ;
```

valores (inteiros) fornecidos devem ser separados pelo caractere dois pontos (:)

Tomada de Decisão

- Comando “if”:
 - comando básico para codificar tomada de decisão
 - se *expr* for verdadeira ($\neq 0$), executa o bloco de comandos 1
 - se *expr* for falsa ($= 0$), executa o bloco de comandos 2

```
if ( expr )  
{ bloco de comandos 1 }  
else  
{ bloco de comandos 2 }
```

ou

```
if ( expr )  
{ bloco de comandos }
```

Exemplo

```
/* nota */
#include <stdio.h>

int main (void)
{
    float nota ;
    printf("Digite sua nota: ");
    scanf("%f", &nota);
    if (nota >= 7 ){
        printf(" Boa nota, parabens! \n");
    }
    else {
        printf(" Voce precisa melhorar. \n");
    }
    return 0;
}
```

Exemplo

```
/* nota */
#include <stdio.h>

int main (void)
{
    float nota ;
    printf("Digite sua nota: ");
    scanf("%f", &nota);
    if (nota >= 7 )
        printf(" Boa nota, parabens! \n");

    else
        printf(" Voce precisa melhorar. \n");

    return 0;
}
```

Bloco de comandos

- {
- comando1;
- comando2;
- ...
- }

- ou

- comando;

Tomada de Decisão

- Exemplo:

- função para qualificar a temperatura:

se a temperatura for menor do que 20°C, então está frio

se a temperatura estiver entre 20°C e 30°C, então está agradável

se a temperatura for maior do que 30°C, então está quente

Tomada de Decisão

```
/* temperatura (versao 1 - incorreta) */
#include <stdio.h>

int main (void)
{
    int temp;
    printf("Digite a temperatura: ");
    scanf("%d", &temp);
    if (temp < 30)
        if (temp > 20)
            printf(" Temperatura agradável \n");
    else
        printf(" Temperatura quente \n");
    return 0;
}
```

Em C, um `else` está associado ao último `if` que não tiver seu próprio `else`.

Tomada de Decisão

```
/* temperatura (versao 1 - incorreta) */  
#include <stdio.h>  
  
int main (void)  
{  
    int temp;  
    printf("Digite a temperatura: ");  
    scanf("%d", &temp);  
    if (temp < 30)  
        if (temp > 20)  
            printf(" Temperatura agradável \n");  
        else  
            printf(" Temperatura quente \n");  
    return 0;  
}
```

Tomada de Decisão

```
/* temperatura (versao 2) */
#include <stdio.h>

int main (void)
{
    int temp;
    printf ( "Digite a temperatura: " );
    scanf ( "%d", &temp );
    if ( temp < 30 ) {
        if ( temp > 20 )
            printf ( " Temperatura agradável \n" );
    }
    else
        printf ( " Temperatura quente \n" );
    return 0;
}
```



```
/* temperatura (versao 3) */  
#include <stdio.h>  
  
int main (void)  
{  
    int temp;  
    printf("Digite a temperatura: ");  
    scanf("%d", &temp);  
  
    if (temp < 10)  
        printf("Temperatura muito fria \n");  
    else if (temp < 20)  
        printf(" Temperatura fria \n");  
    else if (temp < 30)  
        printf("Temperatura agradável \n");  
    else  
        printf("Temperatura quente \n");  
    return 0;  
}
```

```
/* temperatura (versao 3) */  
#include <stdio.h>  
  
int main (void)  
{  
    int temp;  
    printf("Digite a temperatura: ");  
    scanf("%d", &temp);  
  
    if (temp < 10)  
        printf("Temperatura muito fria \n");  
    else if (temp < 20)  
        printf(" Temperatura fria \n");  
    else if (temp < 30)  
        printf("Temperatura agradável \n");  
    else  
        printf("Temperatura quente \n");  
    return 0;  
}
```

```
/* temperatura (versao 3) */
#include <stdio.h>

int main (void)
{
    int temp;
    printf("Digite a temperatura: ");
    scanf("%d", &temp);

    if (temp < 10)
        printf("Temperatura muito fria \n");
    else if (temp < 20)
        printf(" Temperatura fria \n");
    else if (temp < 30)
        printf("Temperatura agradável \n");
    else
        printf("Temperatura quente \n");

    return 0;
}
```

```
/* temperatura (versao 3) */  
#include <stdio.h>  
  
int main (void)  
{  
    int temp;  
    printf("Digite a temperatura: ");  
    scanf("%d", &temp);  
  
    if (temp < 10)  
        printf("Temperatura muito fria \n");  
    else if (temp < 20)  
        printf(" Temperatura fria \n");  
    else if (temp < 30)  
        printf("Temperatura agradável \n");  
    else  
        printf("Temperatura quente \n");  
  
    return 0;  
}
```

```
/* temperatura (versao 3) */
#include <stdio.h>

int main (void)
{
    int temp;
    printf("Digite a temperatura: ");
    scanf("%d", &temp);

    if (temp < 10)
        printf("Temperatura muito fria \n");
    else if (temp < 20)
        printf(" Temperatura fria \n");
    else if (temp < 30)
        printf("Temperatura agradável \n");
    else
        printf("Temperatura quente \n");

    return 0;
}
```

Tomada de Decisão

- Estrutura de bloco:
 - declaração de variáveis:
 - só podem ocorrer no início do corpo da função ou de um bloco
 - (esta restrição não existe no C99)
 - escopo de uma variável:
 - uma variável declarada dentro de um bloco é válida no bloco
 - após o término do bloco, a variável deixa de existir

```
if ( n > 0 )  
    { int i;    ... }  
...           /* a variável i não existe neste ponto do programa */
```

Tomada de Decisão

- Operador condicional:

- formato geral:

- se a condição for verdadeira, a expressão1 é avaliada;
caso contrário, a expressão2 é avaliada

```
condição ? expressão1 : expressão2;
```

- exemplo:

- comando

```
maximo = a > b ? a : b ;
```

- comando “if” equivalente

```
if ( a > b )  
    maximo = a;  
else  
    maximo = b;
```

Construções com laços

- Exemplo:
 - fatorial de um número inteiro não negativo:

~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ ~~6~~ ~~7~~ ~~8~~ ~~9~~ ~~10~~ ~~11~~ ~~12~~ ~~13~~ ~~14~~ ~~15~~ ~~16~~ ~~17~~ ~~18~~ ~~19~~ ~~20~~ ~~21~~ ~~22~~ ~~23~~ ~~24~~ ~~25~~ ~~26~~ ~~27~~ ~~28~~ ~~29~~ ~~30~~ ~~31~~ ~~32~~ ~~33~~ ~~34~~ ~~35~~ ~~36~~ ~~37~~ ~~38~~ ~~39~~ ~~40~~ ~~41~~ ~~42~~ ~~43~~ ~~44~~ ~~45~~ ~~46~~ ~~47~~ ~~48~~ ~~49~~ ~~50~~ ~~51~~ ~~52~~ ~~53~~ ~~54~~ ~~55~~ ~~56~~ ~~57~~ ~~58~~ ~~59~~ ~~60~~ ~~61~~ ~~62~~ ~~63~~ ~~64~~ ~~65~~ ~~66~~ ~~67~~ ~~68~~ ~~69~~ ~~70~~ ~~71~~ ~~72~~ ~~73~~ ~~74~~ ~~75~~ ~~76~~ ~~77~~ ~~78~~ ~~79~~ ~~80~~ ~~81~~ ~~82~~ ~~83~~ ~~84~~ ~~85~~ ~~86~~ ~~87~~ ~~88~~ ~~89~~ ~~90~~ ~~91~~ ~~92~~ ~~93~~ ~~94~~ ~~95~~ ~~96~~ ~~97~~ ~~98~~ ~~99~~ ~~100~~ ~~101~~ ~~102~~ ~~103~~ ~~104~~ ~~105~~ ~~106~~ ~~107~~ ~~108~~ ~~109~~ ~~110~~ ~~111~~ ~~112~~ ~~113~~ ~~114~~ ~~115~~ ~~116~~ ~~117~~ ~~118~~ ~~119~~ ~~120~~ ~~121~~ ~~122~~ ~~123~~ ~~124~~ ~~125~~ ~~126~~ ~~127~~ ~~128~~ ~~129~~ ~~130~~ ~~131~~ ~~132~~ ~~133~~ ~~134~~ ~~135~~ ~~136~~ ~~137~~ ~~138~~ ~~139~~ ~~140~~ ~~141~~ ~~142~~ ~~143~~ ~~144~~ ~~145~~ ~~146~~ ~~147~~ ~~148~~ ~~149~~ ~~150~~ ~~151~~ ~~152~~ ~~153~~ ~~154~~ ~~155~~ ~~156~~ ~~157~~ ~~158~~ ~~159~~ ~~160~~ ~~161~~ ~~162~~ ~~163~~ ~~164~~ ~~165~~ ~~166~~ ~~167~~ ~~168~~ ~~169~~ ~~170~~ ~~171~~ ~~172~~ ~~173~~ ~~174~~ ~~175~~ ~~176~~ ~~177~~ ~~178~~ ~~179~~ ~~180~~ ~~181~~ ~~182~~ ~~183~~ ~~184~~ ~~185~~ ~~186~~ ~~187~~ ~~188~~ ~~189~~ ~~190~~ ~~191~~ ~~192~~ ~~193~~ ~~194~~ ~~195~~ ~~196~~ ~~197~~ ~~198~~ ~~199~~ ~~200~~ ~~201~~ ~~202~~ ~~203~~ ~~204~~ ~~205~~ ~~206~~ ~~207~~ ~~208~~ ~~209~~ ~~210~~ ~~211~~ ~~212~~ ~~213~~ ~~214~~ ~~215~~ ~~216~~ ~~217~~ ~~218~~ ~~219~~ ~~220~~ ~~221~~ ~~222~~ ~~223~~ ~~224~~ ~~225~~ ~~226~~ ~~227~~ ~~228~~ ~~229~~ ~~230~~ ~~231~~ ~~232~~ ~~233~~ ~~234~~ ~~235~~ ~~236~~ ~~237~~ ~~238~~ ~~239~~ ~~240~~ ~~241~~ ~~242~~ ~~243~~ ~~244~~ ~~245~~ ~~246~~ ~~247~~ ~~248~~ ~~249~~ ~~250~~ ~~251~~ ~~252~~ ~~253~~ ~~254~~ ~~255~~ ~~256~~ ~~257~~ ~~258~~ ~~259~~ ~~260~~ ~~261~~ ~~262~~ ~~263~~ ~~264~~ ~~265~~ ~~266~~ ~~267~~ ~~268~~ ~~269~~ ~~270~~ ~~271~~ ~~272~~ ~~273~~ ~~274~~ ~~275~~ ~~276~~ ~~277~~ ~~278~~ ~~279~~ ~~280~~ ~~281~~ ~~282~~ ~~283~~ ~~284~~ ~~285~~ ~~286~~ ~~287~~ ~~288~~ ~~289~~ ~~290~~ ~~291~~ ~~292~~ ~~293~~ ~~294~~ ~~295~~ ~~296~~ ~~297~~ ~~298~~ ~~299~~ ~~300~~ ~~301~~ ~~302~~ ~~303~~ ~~304~~ ~~305~~ ~~306~~ ~~307~~ ~~308~~ ~~309~~ ~~310~~ ~~311~~ ~~312~~ ~~313~~ ~~314~~ ~~315~~ ~~316~~ ~~317~~ ~~318~~ ~~319~~ ~~320~~ ~~321~~ ~~322~~ ~~323~~ ~~324~~ ~~325~~ ~~326~~ ~~327~~ ~~328~~ ~~329~~ ~~330~~ ~~331~~ ~~332~~ ~~333~~ ~~334~~ ~~335~~ ~~336~~ ~~337~~ ~~338~~ ~~339~~ ~~340~~ ~~341~~ ~~342~~ ~~343~~ ~~344~~ ~~345~~ ~~346~~ ~~347~~ ~~348~~ ~~349~~ ~~350~~ ~~351~~ ~~352~~ ~~353~~ ~~354~~ ~~355~~ ~~356~~ ~~357~~ ~~358~~ ~~359~~ ~~360~~ ~~361~~ ~~362~~ ~~363~~ ~~364~~ ~~365~~ ~~366~~ ~~367~~ ~~368~~ ~~369~~ ~~370~~ ~~371~~ ~~372~~ ~~373~~ ~~374~~ ~~375~~ ~~376~~ ~~377~~ ~~378~~ ~~379~~ ~~380~~ ~~381~~ ~~382~~ ~~383~~ ~~384~~ ~~385~~ ~~386~~ ~~387~~ ~~388~~ ~~389~~ ~~390~~ ~~391~~ ~~392~~ ~~393~~ ~~394~~ ~~395~~ ~~396~~ ~~397~~ ~~398~~ ~~399~~ ~~400~~ ~~401~~ ~~402~~ ~~403~~ ~~404~~ ~~405~~ ~~406~~ ~~407~~ ~~408~~ ~~409~~ ~~410~~ ~~411~~ ~~412~~ ~~413~~ ~~414~~ ~~415~~ ~~416~~ ~~417~~ ~~418~~ ~~419~~ ~~420~~ ~~421~~ ~~422~~ ~~423~~ ~~424~~ ~~425~~ ~~426~~ ~~427~~ ~~428~~ ~~429~~ ~~430~~ ~~431~~ ~~432~~ ~~433~~ ~~434~~ ~~435~~ ~~436~~ ~~437~~ ~~438~~ ~~439~~ ~~440~~ ~~441~~ ~~442~~ ~~443~~ ~~444~~ ~~445~~ ~~446~~ ~~447~~ ~~448~~ ~~449~~ ~~450~~ ~~451~~ ~~452~~ ~~453~~ ~~454~~ ~~455~~ ~~456~~ ~~457~~ ~~458~~ ~~459~~ ~~460~~ ~~461~~ ~~462~~ ~~463~~ ~~464~~ ~~465~~ ~~466~~ ~~467~~ ~~468~~ ~~469~~ ~~470~~ ~~471~~ ~~472~~ ~~473~~ ~~474~~ ~~475~~ ~~476~~ ~~477~~ ~~478~~ ~~479~~ ~~480~~ ~~481~~ ~~482~~ ~~483~~ ~~484~~ ~~485~~ ~~486~~ ~~487~~ ~~488~~ ~~489~~ ~~490~~ ~~491~~ ~~492~~ ~~493~~ ~~494~~ ~~495~~ ~~496~~ ~~497~~ ~~498~~ ~~499~~ ~~500~~ ~~501~~ ~~502~~ ~~503~~ ~~504~~ ~~505~~ ~~506~~ ~~507~~ ~~508~~ ~~509~~ ~~510~~ ~~511~~ ~~512~~ ~~513~~ ~~514~~ ~~515~~ ~~516~~ ~~517~~ ~~518~~ ~~519~~ ~~520~~ ~~521~~ ~~522~~ ~~523~~ ~~524~~ ~~525~~ ~~526~~ ~~527~~ ~~528~~ ~~529~~ ~~530~~ ~~531~~ ~~532~~ ~~533~~ ~~534~~ ~~535~~ ~~536~~ ~~537~~ ~~538~~ ~~539~~ ~~540~~ ~~541~~ ~~542~~ ~~543~~ ~~544~~ ~~545~~ ~~546~~ ~~547~~ ~~548~~ ~~549~~ ~~550~~ ~~551~~ ~~552~~ ~~553~~ ~~554~~ ~~555~~ ~~556~~ ~~557~~ ~~558~~ ~~559~~ ~~560~~ ~~561~~ ~~562~~ ~~563~~ ~~564~~ ~~565~~ ~~566~~ ~~567~~ ~~568~~ ~~569~~ ~~570~~ ~~571~~ ~~572~~ ~~573~~ ~~574~~ ~~575~~ ~~576~~ ~~577~~ ~~578~~ ~~579~~ ~~580~~ ~~581~~ ~~582~~ ~~583~~ ~~584~~ ~~585~~ ~~586~~ ~~587~~ ~~588~~ ~~589~~ ~~590~~ ~~591~~ ~~592~~ ~~593~~ ~~594~~ ~~595~~ ~~596~~ ~~597~~ ~~598~~ ~~599~~ ~~600~~ ~~601~~ ~~602~~ ~~603~~ ~~604~~ ~~605~~ ~~606~~ ~~607~~ ~~608~~ ~~609~~ ~~610~~ ~~611~~ ~~612~~ ~~613~~ ~~614~~ ~~615~~ ~~616~~ ~~617~~ ~~618~~ ~~619~~ ~~620~~ ~~621~~ ~~622~~ ~~623~~ ~~624~~ ~~625~~ ~~626~~ ~~627~~ ~~628~~ ~~629~~ ~~630~~ ~~631~~ ~~632~~ ~~633~~ ~~634~~ ~~635~~ ~~636~~ ~~637~~ ~~638~~ ~~639~~ ~~640~~ ~~641~~ ~~642~~ ~~643~~ ~~644~~ ~~645~~ ~~646~~ ~~647~~ ~~648~~ ~~649~~ ~~650~~ ~~651~~ ~~652~~ ~~653~~ ~~654~~ ~~655~~ ~~656~~ ~~657~~ ~~658~~ ~~659~~ ~~660~~ ~~661~~ ~~662~~ ~~663~~ ~~664~~ ~~665~~ ~~666~~ ~~667~~ ~~668~~ ~~669~~ ~~670~~ ~~671~~ ~~672~~ ~~673~~ ~~674~~ ~~675~~ ~~676~~ ~~677~~ ~~678~~ ~~679~~ ~~680~~ ~~681~~ ~~682~~ ~~683~~ ~~684~~ ~~685~~ ~~686~~ ~~687~~ ~~688~~ ~~689~~ ~~690~~ ~~691~~ ~~692~~ ~~693~~ ~~694~~ ~~695~~ ~~696~~ ~~697~~ ~~698~~ ~~699~~ ~~700~~ ~~701~~ ~~702~~ ~~703~~ ~~704~~ ~~705~~ ~~706~~ ~~707~~ ~~708~~ ~~709~~ ~~710~~ ~~711~~ ~~712~~ ~~713~~ ~~714~~ ~~715~~ ~~716~~ ~~717~~ ~~718~~ ~~719~~ ~~720~~ ~~721~~ ~~722~~ ~~723~~ ~~724~~ ~~725~~ ~~726~~ ~~727~~ ~~728~~ ~~729~~ ~~730~~ ~~731~~ ~~732~~ ~~733~~ ~~734~~ ~~735~~ ~~736~~ ~~737~~ ~~738~~ ~~739~~ ~~740~~ ~~741~~ ~~742~~ ~~743~~ ~~744~~ ~~745~~ ~~746~~ ~~747~~ ~~748~~ ~~749~~ ~~750~~ ~~751~~ ~~752~~ ~~753~~ ~~754~~ ~~755~~ ~~756~~ ~~757~~ ~~758~~ ~~759~~ ~~760~~ ~~761~~ ~~762~~ ~~763~~ ~~764~~ ~~765~~ ~~766~~ ~~767~~ ~~768~~ ~~769~~ ~~770~~ ~~771~~ ~~772~~ ~~773~~ ~~774~~ ~~775~~ ~~776~~ ~~777~~ ~~778~~ ~~779~~ ~~780~~ ~~781~~ ~~782~~ ~~783~~ ~~784~~ ~~785~~ ~~786~~ ~~787~~ ~~788~~ ~~789~~ ~~790~~ ~~791~~ ~~792~~ ~~793~~ ~~794~~ ~~795~~ ~~796~~ ~~797~~ ~~798~~ ~~799~~ ~~800~~ ~~801~~ ~~802~~ ~~803~~ ~~804~~ ~~805~~ ~~806~~ ~~807~~ ~~808~~ ~~809~~ ~~810~~ ~~811~~ ~~812~~ ~~813~~ ~~814~~ ~~815~~ ~~816~~ ~~817~~ ~~818~~ ~~819~~ ~~820~~ ~~821~~ ~~822~~ ~~823~~ ~~824~~ ~~825~~ ~~826~~ ~~827~~ ~~828~~ ~~829~~ ~~830~~ ~~831~~ ~~832~~ ~~833~~ ~~834~~ ~~835~~ ~~836~~ ~~837~~ ~~838~~ ~~839~~ ~~840~~ ~~841~~ ~~842~~ ~~843~~ ~~844~~ ~~845~~ ~~846~~ ~~847~~ ~~848~~ ~~849~~ ~~850~~ ~~851~~ ~~852~~ ~~853~~ ~~854~~ ~~855~~ ~~856~~ ~~857~~ ~~858~~ ~~859~~ ~~860~~ ~~861~~ ~~862~~ ~~863~~ ~~864~~ ~~865~~ ~~866~~ ~~867~~ ~~868~~ ~~869~~ ~~870~~ ~~871~~ ~~872~~ ~~873~~ ~~874~~ ~~875~~ ~~876~~ ~~877~~ ~~878~~ ~~879~~ ~~880~~ ~~881~~ ~~882~~ ~~883~~ ~~884~~ ~~885~~ ~~886~~ ~~887~~ ~~888~~ ~~889~~ ~~890~~ ~~891~~ ~~892~~ ~~893~~ ~~894~~ ~~895~~ ~~896~~ ~~897~~ ~~898~~ ~~899~~ ~~900~~ ~~901~~ ~~902~~ ~~903~~ ~~904~~ ~~905~~ ~~906~~ ~~907~~ ~~908~~ ~~909~~ ~~910~~ ~~911~~ ~~912~~ ~~913~~ ~~914~~ ~~915~~ ~~916~~ ~~917~~ ~~918~~ ~~919~~ ~~920~~ ~~921~~ ~~922~~ ~~923~~ ~~924~~ ~~925~~ ~~926~~ ~~927~~ ~~928~~ ~~929~~ ~~930~~ ~~931~~ ~~932~~ ~~933~~ ~~934~~ ~~935~~ ~~936~~ ~~937~~ ~~938~~ ~~939~~ ~~940~~ ~~941~~ ~~942~~ ~~943~~ ~~944~~ ~~945~~ ~~946~~ ~~947~~ ~~948~~ ~~949~~ ~~950~~ ~~951~~ ~~952~~ ~~953~~ ~~954~~ ~~955~~ ~~956~~ ~~957~~ ~~958~~ ~~959~~ ~~960~~ ~~961~~ ~~962~~ ~~963~~ ~~964~~ ~~965~~ ~~966~~ ~~967~~ ~~968~~ ~~969~~ ~~970~~ ~~971~~ ~~972~~ ~~973~~ ~~974~~ ~~975~~ ~~976~~ ~~977~~ ~~978~~ ~~979~~ ~~980~~ ~~981~~ ~~982~~ ~~983~~ ~~984~~ ~~985~~ ~~986~~ ~~987~~ ~~988~~ ~~989~~ ~~990~~ ~~991~~ ~~992~~ ~~993~~ ~~994~~ ~~995~~ ~~996~~ ~~997~~ ~~998~~ ~~999~~ ~~1000~~ ~~1001~~ ~~1002~~ ~~1003~~ ~~1004~~ ~~1005~~ ~~1006~~ ~~1007~~ ~~1008~~ ~~1009~~ ~~1010~~ ~~1011~~ ~~1012~~ ~~1013~~ ~~1014~~ ~~1015~~ ~~1016~~ ~~1017~~ ~~1018~~ ~~1019~~ ~~1020~~ ~~1021~~ ~~1022~~ ~~1023~~ ~~1024~~ ~~1025~~ ~~1026~~ ~~1027~~ ~~1028~~ ~~1029~~ ~~1030~~ ~~1031~~ ~~1032~~ ~~1033~~ ~~1034~~ ~~1035~~ ~~1036~~ ~~1037~~ ~~1038~~ ~~1039~~ ~~1040~~ ~~1041~~ ~~1042~~ ~~1043~~ ~~1044~~ ~~1045~~ ~~1046~~ ~~1047~~ ~~1048~~ ~~1049~~ ~~1050~~ ~~1051~~ ~~1052~~ ~~1053~~ ~~1054~~ ~~1055~~ ~~1056~~ ~~1057~~ ~~1058~~ ~~1059~~ ~~1060~~ ~~1061~~ ~~1062~~ ~~1063~~ ~~1064~~ ~~1065~~ ~~1066~~ ~~1067~~ ~~1068~~ ~~1069~~ ~~1070~~ ~~1071~~ ~~1072~~ ~~1073~~ ~~1074~~ ~~1075~~ ~~1076~~ ~~1077~~ ~~1078~~ ~~1079~~ ~~1080~~ ~~1081~~ ~~1082~~ ~~1083~~ ~~1084~~ ~~1085~~ ~~1086~~ ~~1087~~ ~~1088~~ ~~1089~~ ~~1090~~ ~~1091~~ ~~1092~~ ~~1093~~ ~~1094~~ ~~1095~~ ~~1096~~ ~~1097~~ ~~1098~~ ~~1099~~ ~~1100~~ ~~1101~~ ~~1102~~ ~~1103~~ ~~1104~~ ~~1105~~ ~~1106~~ ~~1107~~ ~~1108~~ ~~1109~~ ~~1110~~ ~~1111~~ ~~1112~~ ~~1113~~ ~~1114~~ ~~1115~~ ~~1116~~ ~~1117~~ ~~1118~~ ~~1119~~ ~~1120~~ ~~1121~~ ~~1122~~ ~~1123~~ ~~1124~~ ~~1125~~ ~~1126~~ ~~1127~~ ~~1128~~ ~~1129~~ ~~1130~~ ~~1131~~ ~~1132~~ ~~1133~~ ~~1134~~ ~~1135~~ ~~1136~~ ~~1137~~ ~~1138~~ ~~1139~~ ~~1140~~ ~~1141~~ ~~1142~~ ~~1143~~ ~~1144~~ ~~1145~~ ~~1146~~ ~~1147~~ ~~1148~~ ~~1149~~ ~~1150~~ ~~1151~~ ~~1152~~ ~~1153~~ ~~1154~~ ~~1155~~ ~~1156~~ ~~1157~~ ~~1158~~ ~~1159~~ ~~1160~~ ~~1161~~ ~~1162~~ ~~1163~~ ~~1164~~ ~~1165~~ ~~1166~~ ~~1167~~ ~~1168~~ ~~1169~~ ~~1170~~ ~~1171~~ ~~1172~~ ~~1173~~ ~~1174~~ ~~1175~~ ~~1176~~ ~~1177~~ ~~1178~~ ~~1179~~ ~~1180~~ ~~1181~~ ~~1182~~ ~~1183~~ ~~1184~~ ~~1185~~ ~~1186~~ ~~1187~~ ~~1188~~ ~~1189~~ ~~1190~~ ~~1191~~ ~~1192~~ ~~1193~~ ~~1194~~ ~~1195~~ ~~1196~~ ~~1197~~ ~~1198~~ ~~1199~~ ~~1200~~ ~~1201~~ ~~1202~~ ~~1203~~ ~~1204~~ ~~1205~~ ~~1206~~ ~~1207~~ ~~1208~~ ~~1209~~ ~~1210~~ ~~1211~~ ~~1212~~ ~~1213~~ ~~1214~~ ~~1215~~ ~~1216~~ ~~1217~~ ~~1218~~ ~~1219~~ ~~1220~~ ~~1221~~ ~~1222~~ ~~1223~~ ~~1224~~ ~~1225~~ ~~1226~~ ~~1227~~ ~~1228~~ ~~1229~~ ~~1230~~ ~~1231~~ ~~1232~~ ~~1233~~ ~~1234~~ ~~1235~~ ~~1236~~ ~~1237~~ ~~1238~~ ~~1239~~ ~~1240~~ ~~1241~~ ~~1242~~ ~~1243~~ ~~1244~~ ~~1245~~ ~~1246~~ ~~1247~~ ~~1248~~ ~~1249~~ ~~1250~~ ~~1251~~ ~~1252~~ ~~1253~~ ~~1254~~ ~~1255~~ ~~1256~~ ~~1257~~ ~~1258~~ ~~1259~~ ~~1260~~ ~~1261~~ ~~1262~~ ~~1263~~ ~~1264~~ ~~1265~~ ~~1266~~ ~~1267~~ ~~1268~~ ~~1269~~ ~~1270~~ ~~1271~~ ~~1272~~ ~~1273~~ ~~1274~~ ~~1275~~ ~~1276~~ ~~1277~~ ~~1278~~ ~~1279~~ ~~1280~~ ~~1281~~ ~~1282~~ ~~1283~~ ~~1284~~ ~~1285~~ ~~1286~~ ~~1287~~ ~~1288~~ ~~1289~~ ~~1290~~ ~~1291~~ ~~1292~~ ~~1293~~ ~~1294~~ ~~1295~~ ~~1296~~ ~~1297~~ ~~1298~~ ~~1299~~ ~~1300~~ ~~1301~~ ~~1302~~ ~~1303~~ ~~1304~~ ~~1305~~ ~~1306~~ ~~1307~~ ~~1308~~ ~~1309~~ ~~1310~~ ~~1311~~ ~~1312~~ ~~1313~~ ~~1314~~ ~~1315~~ ~~1316~~ ~~1317~~ ~~1318~~ ~~1319~~ ~~1320~~ ~~1321~~ ~~1322~~ ~~1323~~ ~~1324~~ ~~1325~~ ~~1326~~ ~~1327~~ ~~1328~~ ~~1329~~ ~~1330~~ ~~1331~~ ~~1332~~ ~~1333~~ ~~1334~~ ~~1335~~ ~~1336~~

Construções com laços

- Exemplo:
 - definição recursiva da função *fatorial*: $N \rightarrow N$
 $fatorial(0) = 1$
 $fatorial(n) = n \times fatorial(n-1)$
 - cálculo não recursivo de *fatorial*(*n*)
 - comece com:
 $k = 1$
 $fatorial = 1$
 - faça enquanto $k \leq n$
 $fatorial = fatorial * k$
incremente k

Construções com laços

- Comando “while”:
 - enquanto *expr* for verdadeira, o bloco de comandos é executado
 - quando *expr* for falsa, o comando termina

```
while ( expr )  
{  
    bloco de comandos  
}
```

```

/* Fatorial */
#include <stdio.h>
int main (void)
{
    int i;
    int n;
    long int f = 1;
    printf("Digite um numero inteiro nao negativo:");
    scanf("%d", &n);

    /* calcula fatorial */
    i = 1;
    while (i <= n)
    {
        f = f * i;          /* equivalente a "f *= i"   */
        i = i + 1;          /*   equivalente a "i++"       */
    }
    printf(" Fatorial = %d \n", f);
    return 0;
}

```

Construções com laços

- Comando “for”:
 - forma compacta para exprimir laços

```
for (expressão_inicial; expressão_booleana; expressão_de_incremento)  
{  
    bloco de comandos  
}
```

- equivalente a:

```
expressão_inicial;  
while ( expressão_booleana )  
{  
    bloco de comandos  
    ...  
    expressão_de_incremento  
}
```

```
/* Fatorial (versao 2) */
#include <stdio.h>

int main (void)
{
    int i;
    int n;
    int f = 1;

    printf("Digite um numero inteiro nao negativo:");
    scanf("%d", &n);

    /* calcula fatorial */
    for (i = 1; i <= n; i=i+1) {
        f = f * i;
    }
    printf(" Fatorial = %d \n", f);
    return 0;
}
```

```

/* Fatorial (versao 2) */
#include <stdio.h>

int main (void)
{
    int i;
    int n;
    int f = 1;

    printf("Digite um numero inteiro nao negativo:");
    scanf("%d", &n);

    /* calcula fatorial */
    for (i = 1; i <= n; i+1) {      /* o que acontece com este programa? */
        f = f * i;
    }
    printf(" Fatorial = %d \n", f);
    return 0;
}

```

Construções com laços

- Comando “do-while”:
 - teste de encerramento é avaliado no final

```
do
{
    bloco de comandos
} while (expr);
```

```
/* Fatorial (versao 3) */
#include <stdio.h>
int main (void)
{
    int i;
    int n;
    int f = 1;
    /* requisita valor até um número não negativo ser informado */
do
{
    printf("Digite um valor inteiro nao negativo:");
    scanf ("%d", &n);
} while (n<0);
/* calcula fatorial */
for (i = 1; i <= n; i++)
    f *= i;
printf(" Fatorial = %d\n", f);
return 0;
}
```



```

/* Fatorial (versao 4) */
#include <stdio.h>
int main (void)
{
    int i;
    int n;
    int f = 1;
    /* O que faz este programa? */
    do {
        printf("Digite um valor inteiro nao negativo:");
        scanf ("%d", &n);
        /* calcula fatorial */
        for (i = 1; i <= n; i++)
            f *= i;
        printf(" Fatorial = %d\n", f);
    } while (n>=0);
    return 0;
}

```

Construções com laços

- Comando “switch”:
 - seleciona uma entre vários casos
 (“op_k” deve ser um inteiro ou caractere)

```
switch ( expr )  
{  
    case op1: bloco de comandos 1; break;  
    case op2: bloco de comandos 2; break;  
    ...  
    default: bloco de comandos default; break;  
}
```

```

/* calculadora de quatro operações */
#include <stdio.h>
int main (void)
{
    float num1, num2;
    char op;
    printf("Digite: numero op numero\n");
    scanf ("%f %c %f", &num1, &op, &num2);
    switch (op)
    {
        case '+':  printf(" = %f\n", num1+num2); break;
        case '-':  printf(" = %f\n", num1-num2); break;
        case '*':  printf(" = %f\n", num1*num2); break;
        case '/':  printf(" = %f\n", num1/num2); break;
        default:   printf("Operador invalido!\n"); break;
    }
    return 0;
}

```

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;
```

```
}
```

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

chamada da função

**“main” retorna um
inteiro:**

0 : execução OK

≠ 0 : execução →OK

```
/* 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 = %f", f);
```

```
}
```

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 */
{
}

```

/* 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

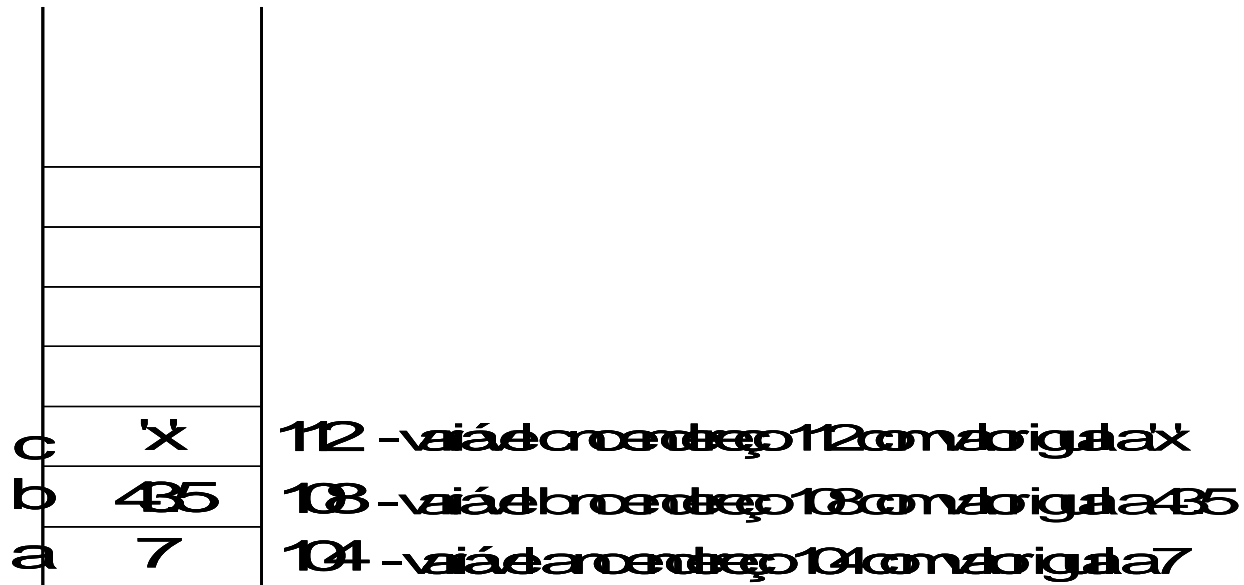
- Comunicação entre funções:
 - funções são independentes entre si
 - transferência de dados entre funções:
 - através dos parâmetros e do valor de retorno da função chamada
 - passagem de parâmetros é feita **por valor**
 - variáveis locais a uma função:
 - definidas dentro do corpo da função (incluindo 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;
}
```

1 - Início do programa: pilha vazia

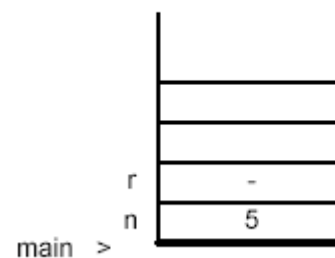


Exemplo: Declaração de n e r na main()

```
/* 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;
}
```

2 - Declaração das variáveis: n, r

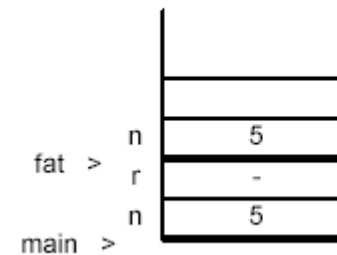


Exemplo: Declaração de n na fat(int n)

```
/* 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;
}
```

3 - Chamada da função: cópia do parâmetro



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;
}
```

4 - Declaração da variável local: f

f		1.0
n		5
fat >	r	-
main >	n	5

Exercícios

- Faça um programa que recebe como entrada três graus: G1, G2 e G3 e calcula a média, se o aluno estiver aprovado, ou informa a necessidade de uma prova final, se o aluno não tiver satisfeito o seguinte critério:
 - Todas as notas maiores ou iguais a 3 E
 - Média aritmética maior ou igual a 5
- Coloque o cálculo da média em uma função separada

Exercícios

```
#include <stdio.h>

float calculaMedia(float g1, float g2, float g3);

int main(void) {
    float g1, g2, g3, media;

    printf("Digite os graus G1, G2 e G3: ");
    scanf("%f %f %f", &g1, &g2, &g3);
    media = calculaMedia(g1, g2, g3);
    if (media >= 5.0 && g1 >= 3.0 && g2 >= 3.0 && g3 >= 3.0) {
        printf("SF = APROVADO, MF = %f\n", media);
    }
    else {
        printf("ALUNO EM PROVA FINAL.\n");
    }
}

float calculaMedia(float g1, float g2, float g3) {
    float media;

    media = (g1 + g2 + g3) / 3;
    return media;
}
```


Exercícios

- Implemente uma função que retorne uma aproximação do valor de PI, de acordo com a Fórmula de Leibniz:

$$\pi = 4 \left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \dots \right)$$

- Ou seja:

$$\pi = 4 \sum_{i=0}^{n-1} \frac{(-1)^i}{2i+1}$$

- A função deve obedecer ao protótipo:
 - float pi(int n);

Exercícios

```
#include <stdio.h>

float pi(int n);

int main(void) {
    int n;
    float p;

    printf("Digite o numero de termos: ");
    scanf("%d", &n);
    if (n < 1) {
        printf("Erro! O numero de termos deve ser maior que zero.\n");
    }
    else {
        p = pi(n);
        printf("PI = %f\n", p);
    }
    return 0;
}

float pi(int n) {
    float soma;
    int i;

    soma = 1;
    for (i = 1; i < n; i++) {
        if (i % 2) {
            soma = soma - (1.0 / ((2 * i) + 1));
        }
        else {
            soma = soma + (1.0 / ((2 * i) + 1));
        }
    }
    return 4*soma;
}
```

Exercícios

```
#include <stdio.h>
#include <math.h>

float pi(int n);

int main(void) {
    int n;
    float p;

    printf("Digite o numero de termos: ");
    scanf("%d", &n);
    if (n < 1) {
        printf("Erro! O numero de termos deve ser maior que zero.\n");
    }
    else {
        p = pi(n);
        printf("PI = %f\n", p);
    }
    return 0;
}

float pi(int n) {
    float soma;
    int i;

    soma = 1;
    for (i = 1; i < n; i++) {
        soma = soma + (pow(-1,i) / ((2 * i) + 1));
    }
    return 4*soma;
}
```

Referências

- Waldemar Celes, Renato Cerqueira, José Lucas Rangel, *Introdução a Estruturas de Dados*, Editora Campus (2004)
- Capítulo 1 – Ciclo de Desenvolvimento
- Capítulo 2 – Expressões e E/S
- Capítulo 3 – Controle de Fluxo
- Capítulo 4 – Funções