



PROGRAMAÇÃO E ESTRUTURAS DE DADOS II

Prof. Rodrigo De Vit
SI UFSM-FW

Resumo CV



- Doutorado (em finalização) em Ciência da Computação pela PUCRS, Mestrado em Ciência da Computação pela UFRGS e Bacharelado em Informática pela Unijuí. Professor em regime de Dedicação Exclusiva junto à UFSM/FW. Tem experiência na área de Ciência da Computação, com ênfase em Redes de Computadores, atuando principalmente nos seguintes temas: Programação de Computadores (C, C++ e Java), Computação Móvel, Sistemas Distribuídos, Redes de Sensores Sem Fio (RSSF), Redes Ad Hoc Móveis (MANETs) e Delay or Disruption Tolerant Networks (DTNs).
- Atividades Coordenação: Unicruz, Unibalsas, Imed e UFSM.
- <http://lattes.cnpq.br/8345071196441362>

Plano de Ensino



- Plano de Ensino

Material das aulas



UFSM
Portal do Professor







OU ANTONIO RODRIGO DELEPIANE DE VIT (1790483)

Início : Minhas turmas
Minhas turmas

Disciplina
DCG2255 - TÓPICOS ESPECIAIS EM SEGURANÇA COMPUTACIONAL
Turma
60
Curso
2012 - Sistemas de Informação - CAMPUS UFSM-FW
Período
1. Semestre / 2019
Carga horária
60 h/aula
Créditos
4

Alterar senha | Sair

Alunos matriculados na turma

	Nome	Data de inclusão na turma	Curso	E-mail
1.	 CRISTIAN GOETTERT (201420829)	25/02/2019	Sistemas de Informação - CAMPUS UFSM-FW	cristiang149@hotmail.com
2.	 DIOGO CANCIAN BAGGIOTTO (2017510357)	25/02/2019	Sistemas de Informação - CAMPUS UFSM-FW	diogobaggiotto@gmail.com
3.	 DRESLEI GARBIN BARCELOS (201320890)	25/02/2019	Sistemas de Informação - CAMPUS UFSM-FW	dresleigb@gmail.com
4.	 EDSON NOETZOLD (2018520296)	25/02/2019	Sistemas de Informação - CAMPUS UFSM-FW	edsonversusnoetzold@gmail.com
5.	 EVANDRO LUIZ WOLFF (201320220)	25/02/2019	Sistemas de Informação - CAMPUS UFSM-FW	evandro.wolff@hotmail.com
6.	 JEAN FLAVIO DE SOUZA (201420602)	25/02/2019	Sistemas de Informação - CAMPUS UFSM-FW	jeansouz4@gmail.com

UFSM

Inicio ! Notificações Mensagens Ajuda

ANTONIO Sair

TÓPICOS ESPECIAIS EM SEGURANÇA COMPUTACIONAL

Panel / Presencial - UFSM / Sistemas de Informação / 2019/1. Semestre / C2012/T560/D.DCG2255/A/2019/P:101 / Geral / Forum de noticias / Local da aula de 13 de março de 2019

Buscar no fórum

Forum de noticias

Local da aula de 13 de março de 2019

Mostrar respostas aninhadas

Transfira esta discussão para ... Mover

Destacar

Local da aula de 13 de março de 2019
por ANTONIO RODRIGO DELEPIANE DE VIT - quarta, 13 Mar 2019, 13:56

Prezados(as) Alunos(as),

Informo que nossa aula de DCG2255 - Tópicos Especiais em Segurança Computacional se dará no Laboratório de Hardware do Bloco 6.

Saudações,
Prof. Rodrigo.

[Link direto](#) | [Editar](#) | [Excluir](#) | [Responder](#)

Administração

- Administração do fórum
 - Editar configurações
 - Papéis atribuídos localmente
 - Verificar permissões
 - Filtros
 - Logs
 - Backup
 - Restaurar
- Modo de assinatura
 - Mostrar assinantes
- Administração do curso
- Mudar papel para...

Avisos importantes

- Avisos para a turma:
 - Somente por email/Moodle.
 - Para entrar na lista da disciplina, basta ter email atualizado no Portal do Aluno.

Isto paga bem?

- **Desenvolvedor Full-Stack**

- O que faz: desenvolve softwares para gestão e alteração da base de dados, além de softwares para interface com o usuário final.
- Perfil da vaga: comprometimento com resultados e prazos, além da capacidade de trabalhar em equipe.
- Salário: entre R\$ 5 mil e R\$ 14 mil.
- Por que está em alta: a visão mais completa do sistema e a flexibilidade de trabalhar com diferentes etapas de desenvolvimento – desde o sistema do servidor até o software da interface final com o usuário – fazem com que este profissional seja altamente disputado pelas empresas, elevando assim, os salários ofertados.
- Mobile: entre R\$ 6 mil e R\$ 13 mil.
- Fonte: <https://exame.abril.com.br/carreira/17-cargos-na-area-de-tecnologia-que-vao-bombar-em-2019/>

Revisão

- UNIDADE 4 - DADOS, EXPRESSÕES E ALGORITMOS SEQUENCIAIS
- 4.1 - Tipos de dados.
- 4.2 - Constantes e variáveis.
- 4.3 - Expressões.
- 4.4 - Atribuição.
- 4.5 - Entrada e saída.

Revisão



- UNIDADE 5 - ALGORITMOS ESTRUTURADOS
- 5.1 - Execução condicional.
- 5.2 - Estruturas de repetição.
- 5.3 - Contadores e acumuladores.

Revisão



- UNIDADE 6 - DADOS ESTRUTURADOS

- 6.1 - Variáveis compostas homogêneas.
 - 6.1.1 - Unidimensionais.
 - 6.1.2 - Multidimensionais.
- 6.2 - Variáveis compostas heterogêneas.
- 6.3 - Ponteiros e estruturas dinâmicas.

Revisão



- UNIDADE 7 – MODULARIZAÇÃO
- 7.1 - Subprogramas.
- 7.2 - Argumentos.

Revisão

UNIDADE 4 - DADOS, EXPRESSÕES E ALGORITMOS SEQUENCIAIS



Tipo	Tamanho	Representatividade
char	1 byte	-128 a 127
unsigned char	1 byte	0 a 255
short int	2 bytes	-32 768 a 32 767
unsigned short int	2 bytes	0 a 65 535
long int	4 bytes	-2 147 483 648 a 2 147 483 647
unsigned long int	4 bytes	4 294 967 295

Tipo	Tamanho	Representatividade
float	4 bytes	$\pm 10^{-38}$ a 10^{38}
double	8 bytes	$\pm 10^{-308}$ a 10^{308}

Revisão

UNIDADE 4 - DADOS, EXPRESSÕES E ALGORITMOS SEQUENCIAIS

```
int a;           /* declara uma variável do tipo int */  
int b;           /* declara outra variável do tipo int */  
float c;         /* declara uma variável do tipo float */
```

```
a = 5;           /* armazena o valor 5 em a */  
b = 10;          /* armazena o valor 10 em b */  
c = 5.3;         /* armazena o valor 5.3 em c */
```

```
int a, b;        /* declara duas variáveis do tipo int */
```

```
int a = 5, b = 10; /* declara e inicializa as variáveis */  
float c = 5.3;
```

Revisão

UNIDADE 4 - DADOS, EXPRESSÕES E ALGORITMOS SEQUENCIAIS



Operadores Aritméticos

Os operadores aritméticos binários são: +, -, *, / e o operador módulo %. Há ainda o operador unário -. A operação é feita na precisão dos operandos. Assim, a expressão $5/2$ resulta no valor 2, pois a operação de divisão é feita em precisão inteira, já que os dois operandos (5 e 2) são constantes inteiras. A divisão de inteiros trunca a parte fracionária, pois o valor resultante é sempre do mesmo tipo da expressão. Consequentemente, a expressão $5.0/2.0$ resulta no valor real 2.5 pois a operação é feita na precisão real (double, no caso).

O operador módulo, %, não se aplica a valores reais, seus operandos devem ser do tipo inteiro. Este operador produz o resto da divisão do primeiro pelo segundo operando. Como exemplo de aplicação deste operador, podemos citar o caso em que desejamos saber se o valor armazenado numa determinada variável inteira x é par ou ímpar. Para tanto, basta analisar o resultado da aplicação do operador %, aplicado à variável e ao valor dois.

$x \% 2$	se resultado for zero	\Rightarrow número é par
$x \% 2$	se resultado for um	\Rightarrow número é ímpar

Revisão

UNIDADE 4 - DADOS, EXPRESSÕES E ALGORITMOS SEQUENCIAIS



Os operadores $*$, $/$ e $\%$ têm precedência maior que os operadores $+$ e $-$. O operador unário tem precedência maior que $*$, $/$ e $\%$. Operadores com mesma precedência são avaliados da esquerda para a direita. Assim, na expressão:

$$a + b * c / d$$

executa-se primeiro a multiplicação, seguida da divisão, seguida da soma. Podemos utilizar parênteses para alterar a ordem de avaliação de uma expressão. Assim, se quisermos avaliar a soma primeiro, podemos escrever:

$$(a + b) * c / d$$

Uma tabela de precedência dos operadores da linguagem C é apresentada no final desta seção.

Revisão

UNIDADE 4 - DADOS, EXPRESSÕES E ALGORITMOS SEQUENCIAIS

- **Operadores de atribuição**

- Na linguagem C, uma atribuição é uma expressão cujo valor resultante corresponde ao valor atribuído. Assim, da mesma forma que a expressão:

$5 + 3$ resulta no valor 8, a atribuição:

$a = 5$ armazena o valor 5 na variável a

- $y = x = 5$; neste caso, **a ordem de avaliação é da direita para a esquerda**. Assim, o computador avalia $x = 5$, armazenando 5 em x , e em seguida armazena em y o valor produzido por $x = 5$, que é 5. Portanto, ambos, x e y , recebem o valor 5.
- $i = i + 2$;
- em que a variável à esquerda do sinal de atribuição também aparece à direita, podem ser escritas de forma mais compacta:
- $i += 2$; vale também para $-=$, $*=$, $/=$, $\%=$.

Revisão

UNIDADE 4 - DADOS, EXPRESSÕES E ALGORITMOS SEQUENCIAIS

- **Operadores de incremento e decremento**
- Se n é uma variável que armazena um valor, o comando:
- $n++$; incrementa de uma unidade o valor de n (análogo para o decremento em $n--$). O aspecto não usual é que $++$ e $--$ podem ser usados tanto como operadores pré-fixados (antes da variável, como em $++n$) ou pós-fixados (após a variável, como em $n++$). Em ambos os casos, a variável n é incrementada. Porém, a expressão $++n$ incrementa n *antes* de usar seu valor, enquanto $n++$ incrementa n *após* seu valor ser usado. Isto significa que, num contexto onde o valor de n é usado, $++n$ e $n++$ são diferentes. Se n armazena o valor 5, então:
 - $x = n++$;
 - atribui 5 a x , mas:
 - $x = ++n$;
 - atribuiria 6 a x . Em ambos os casos, n passa a valer 6, pois seu valor foi incrementado em uma unidade.

```
a = a + 1;  
a += 1;  
a++;  
++a;
```


Revisão

UNIDADE 4 - DADOS, EXPRESSÕES E ALGORITMOS SEQUENCIAIS

Operadores relacionais e lógicos

Os operadores relacionais em C são:

<	<i>menor que</i>
>	<i>maior que</i>
<=	<i>menor ou igual que</i>
>=	<i>maior ou igual que</i>
==	<i>igual a</i>
!=	<i>diferente de</i>

Operadores lógicos:

& &	<i>operador binário E (AND)</i>
	<i>operador binário OU (OR)</i>
!	<i>operador unário de NEGAÇÃO (NOT)</i>

Revisão

UNIDADE 4 - DADOS, EXPRESSÕES E ALGORITMOS SEQUENCIAIS



- **Conversão de tipo**
- Na expressão `3/1.5`, o valor da constante `3` (tipo `int`) é promovido (convertido) para `double` antes de a expressão ser avaliada, pois o segundo operando é do tipo `double` (`1.5`) e a operação é feita na precisão do tipo mais representativo.
- `int a = 3.5;` o valor `3.5` é convertido para inteiro (isto é, passa a valer `3`) antes de a atribuição ser efetuada. Como resultado, como era de se esperar, o valor atribuído à variável é `3` (inteiro). Alguns compiladores exibem advertências.
- O programador pode explicitamente requisitar uma conversão de tipo através do uso do operador de molde de tipo (operador *cast*). Por exemplo, são válidos (e isentos de qualquer advertência por parte dos compiladores) os comandos abaixo.

```
int a, b;  
a = (int) 3.5;  
b = (int) 3.5 % 2;
```

Revisão

UNIDADE 4 - DADOS, EXPRESSÕES E ALGORITMOS SEQUENCIAIS

Precedência e ordem de avaliação dos operadores

A tabela abaixo mostra a precedência, em ordem decrescente, dos principais operadores da linguagem C.

Operador	Associatividade
() [] -> .	esquerda para direita
! ~ ++ -- - (tipo) * & sizeof(tipo)	direita para esquerda
* / %	esquerda para direita
+ -	esquerda para direita
<< >>	esquerda para direita
< <= > >=	esquerda para direita
== !=	esquerda para direita
&	esquerda para direita
^	esquerda para direita
	esquerda para direita
&&	esquerda para direita
	esquerda para direita
?:	direita para esquerda
= += -= etc.	direita para esquerda
,	esquerda para direita

Revisão

UNIDADE 4 - DADOS, EXPRESSÕES E ALGORITMOS SEQUENCIAIS

◦ Função printf()

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

<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</code>	<i>especifica um double (ou float)</i>
<code>%e</code>	<i>especifica um double (ou float) no formato científico</i>
<code>%g</code>	<i>especifica um double (ou float) no formato mais apropriado (%f ou %e)</i>
<code>%s</code>	<i>especifica uma cadeia de caracteres</i>

Alguns exemplos:

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

tem como resultado a impressão da linha:

```
33 5.3
```

<code>\n</code>	<i>caractere de nova linha</i>
<code>\t</code>	<i>caractere de tabulação</i>
<code>\r</code>	<i>caractere de retrocesso</i>
<code>\"</code>	<i>o caractere "</i>
<code>\\</code>	<i>o caractere \</i>

Revisão

UNIDADE 4 - DADOS, EXPRESSÕES E ALGORITMOS SEQUENCIAIS

- **Função scanf()**

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

<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>

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

Revisão

UNIDADE 5 - ALGORITMOS ESTRUTURADOS



Um contador
`cont = cont + 1;`

Um acumulador
`acum = acum + variável;`

Revisão

UNIDADE 5 - ALGORITMOS ESTRUTURADOS



3.1. **Decisões com *if***

if é o comando de decisão básico em C. Sua forma pode ser:

```
if (expr) {  
    bloco de comandos 1  
    ...  
}
```

ou

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

```
if ( expr )  
    comando1;  
else  
    comando2;
```

A indentação (recuo de linha) dos comandos é fundamental para uma maior clareza do código. O estilo de indentação varia a gosto do programador. Além da forma ilustrada acima, outro estilo bastante utilizado por programadores C é:

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

Revisão

UNIDADE 5 - ALGORITMOS ESTRUTURADOS

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

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



Revisão

UNIDADE 5 - ALGORITMOS ESTRUTURADOS



```
/* Fatorial (versao 2) */

#include <stdio.h>

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

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

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

```
for (expr_inicial; expr_booleana; expr_de_incremento)
{
    bloco de comandos
    ...
}
```



Revisão

UNIDADE 5 - ALGORITMOS ESTRUTURADOS



```
/* Fatorial (versao 3) */  
  
#include <stdio.h>  
  
int main (void)  
{  
    int i;  
    int n;  
    int f = 1;  
  
    /* requisita valor do usuário */  
    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;  
}
```

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

Revisão

UNIDADE 5 - ALGORITMOS ESTRUTURADOS

```
/* 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;  
}
```

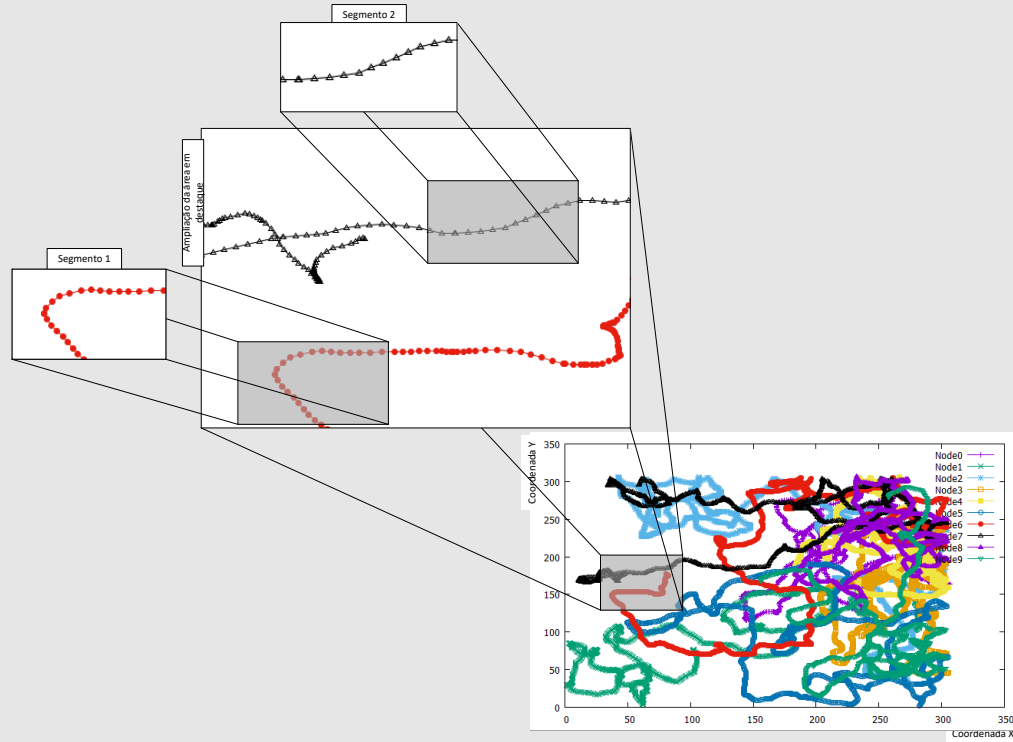
Seleção

Além da construção else-if, C provê um comando (switch) para selecionar um dentre um conjunto de possíveis casos. Sua forma geral é:

```
switch ( expr )  
{  
    case op1:  
        ...           /* comandos executados se expr == op1 */  
        break;  
    case op2:  
        ...           /* comandos executados se expr == op2 */  
        break;  
    case op3:  
        ...           /* comandos executados se expr == op3 */  
        break;  
    default:  
        ...           /* executados se expr for diferente de todos */  
        break;  
}
```

Revisão

UNIDADE 6 - DADOS ESTRUTURADOS



◦ Tipos estruturados

- Para ilustrar, vamos considerar o desenvolvimento de programas que manipulam pontos no plano cartesiano. Cada ponto pode ser representado por suas coordenadas x e y , ambas dadas por valores reais. Sem um mecanismo para agrupar as duas componentes, teríamos que representar cada ponto por duas variáveis independentes.

(1)	(2)	(3)	(4)
<pre>float x; float y;</pre>	<pre>struct ponto { float x; float y; };</pre>	<pre>struct ponto p;</pre>	<pre>ponto.x = 10.0; ponto.y = 5.0;</pre>

Revisão

UNIDADE 6 - DADOS ESTRUTURADOS



Exemplo: Capturar e imprimir as coordenadas de um ponto.

Para exemplificar o uso de estruturas em programas, vamos considerar um em que capturamos e imprimimos as coordenadas de um ponto qualquer.

```
/* Captura e imprime as coordenadas de um ponto qualquer */

#include <stdio.h>

struct ponto {
    float x;
    float y;
};

int main (void)
{
    struct ponto p;

    printf("Digite as coordenadas do ponto(x y): ");
    scanf("%f %f", &p.x, &p.y);
    printf("O ponto fornecido foi: (%.2f,%.2f)\n", p.x, p.y);
    return 0;
}
```

Revisão

UNIDADE 6 - DADOS ESTRUTURADOS

- **Vetores de ponteiros para estruturas**
- Podemos organizar os dados dos alunos em um vetor. Para cada aluno, vamos supor que sejam necessárias as seguintes informações:

(1)	(2)	(3)	(4)
<ul style="list-style-type: none">• <i>nome</i>: cadeia com até 80 caracteres• <i>matricula</i>: número inteiro• <i>endereço</i>: cadeia com até 120 caracteres• <i>telefone</i>: cadeia com até 20 caracteres	<pre>struct aluno { char nome[81]; int mat; char end[121]; char tel[21]; }; typedef struct aluno Aluno;</pre>	<pre>#define MAX 100 Aluno tab[MAX];</pre>	<pre>... tab[i].mat = 9912222; ...</pre>

Revisão

UNIDADE 7 – MODULARIZAÇÃO



```
/* 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);
}
```

Revisão

UNIDADE 7 – MODULARIZAÇÃO

As implementações recursivas devem ser pensadas considerando-se a definição recursiva do problema que desejamos resolver. Por exemplo, o valor do fatorial de um número pode ser definido de forma recursiva:

$$n! = \begin{cases} 1, & \text{se } n = 0 \\ n \times (n-1)!, & \text{se } n > 0 \end{cases}$$

Considerando a definição acima, fica muito simples pensar na implementação recursiva de uma função que calcula e retorna o fatorial de um número.

```
/* Função recursiva para calculo do fatorial */  
  
int fat (int n)  
{  
    if (n==0)  
        return 1;  
    else  
        return n*fat(n-1);  
}
```