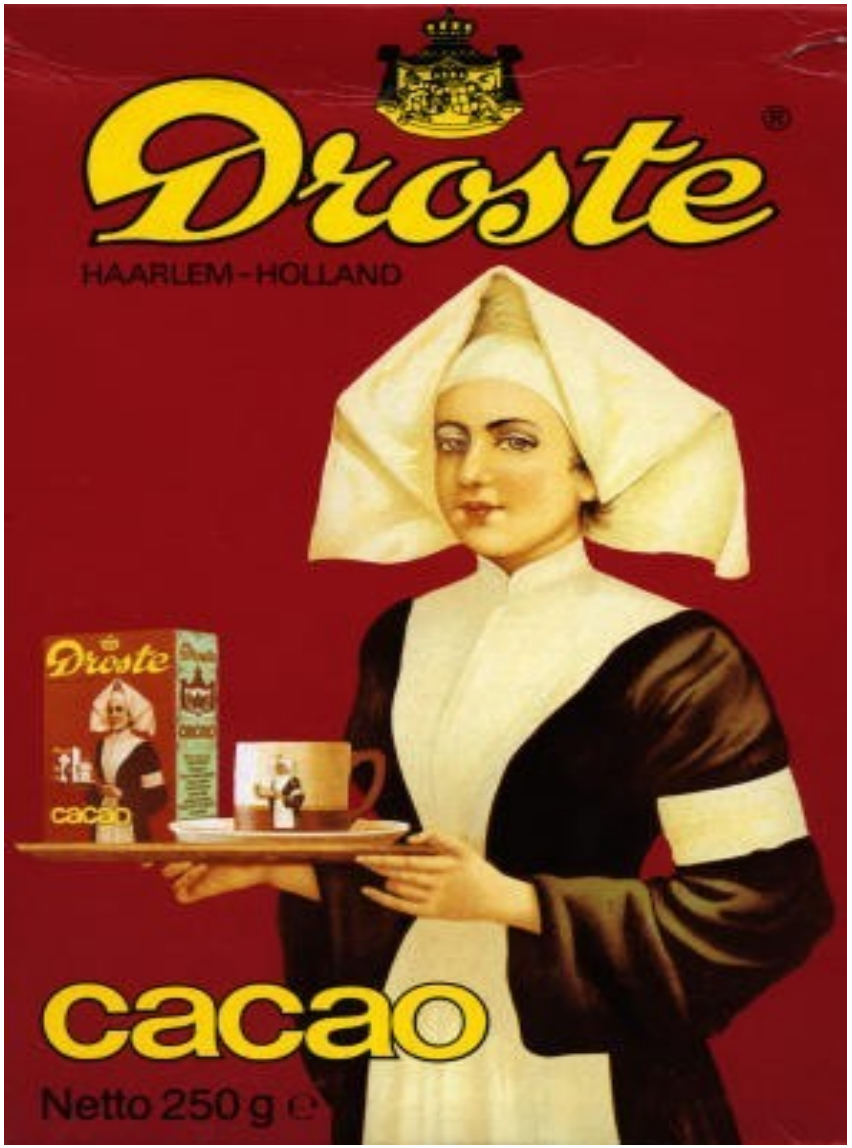
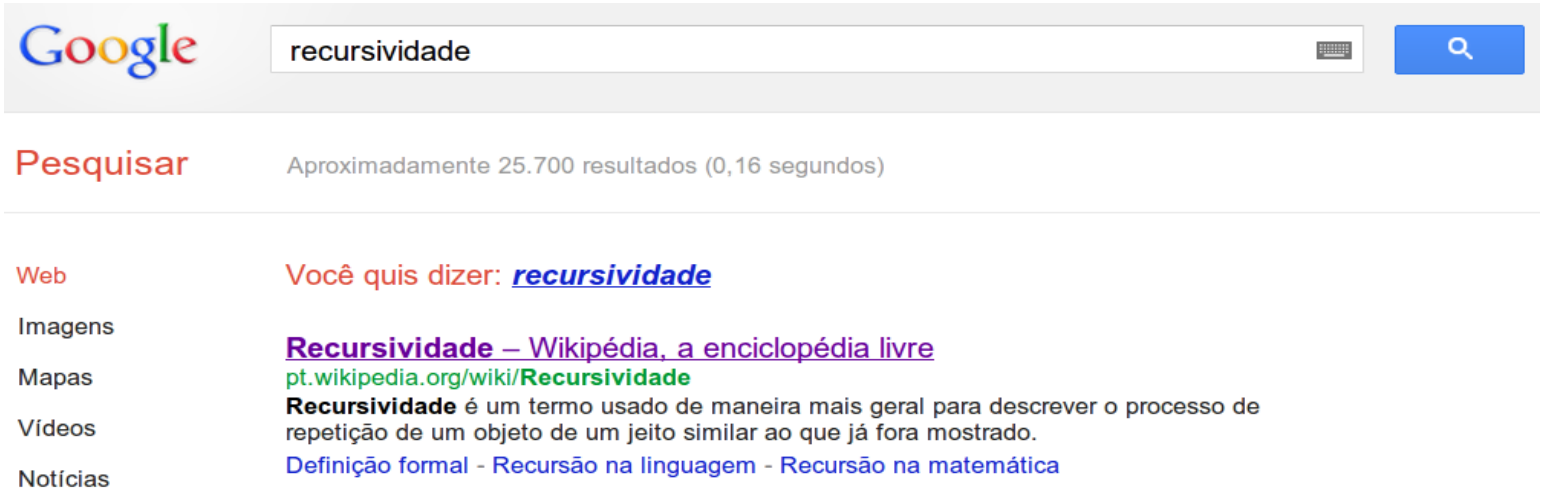
Universidade Federal de Viçosa
Departamento de Informática
Centro de Ciências Exatas e Tecnológicas



INF 112
Programação 2
Aula “5”
Recursividade

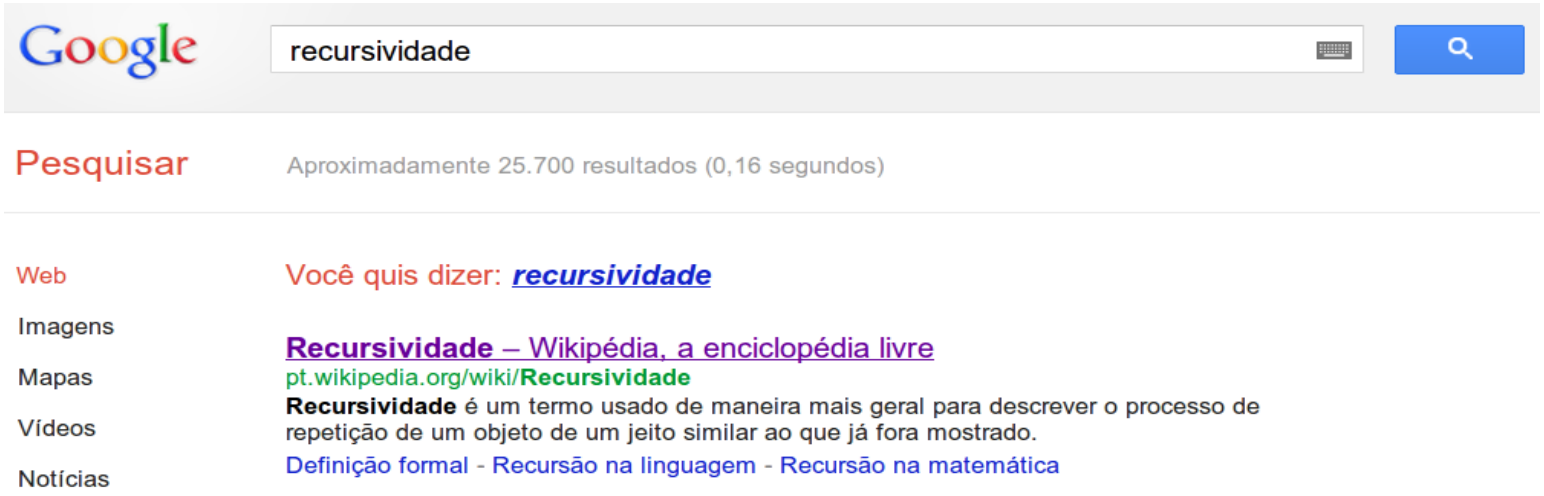
Recursividade

- Definição de recursividade: veja no próximo slide...



Recursividade

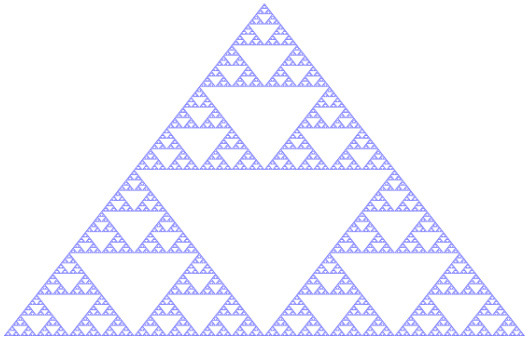
- Definição de recursividade: veja no slide anterior...



Recursividade

Conceito muito utilizado na computação, matemática, etc.. Também encontrado com frequência na natureza.

- Na computação, uma função é recursiva quando ela é definida em termos de si mesma.



Recursividade

- A recursividade é tão importante na computação que muitas siglas famosas são acrônimos recursivos.
- Ex:
  - GNU: “GNU is Not Unix”
  - Wine: “Wine is not an emulator”.

Recursividade

- Exemplo: fatorial.
- O fatorial de n pode ser calculado de forma iterativa (não recursiva) com base na seguinte definição:
  - n! = n x (n-1) x(n-2) x .... x (1) x (1)
- Como seria uma implementação de um código para cálculo de fatorial?

Recursividade

- Exemplo: fatorial.
- O fatorial de n pode ser calculado de forma iterativa (não recursiva) com base na seguinte definição:
  - n! = n x (n-1) x(n-2) x .... x (1) x (1)
- Como seria uma implementação de um código para cálculo de fatorial?

```
int fat(int n) {
	int resp = 1;
	for(int i= n;i >= 2;i--)
		resp *= i;
	return resp;
}
```

Recursividade

- Exemplo: fatorial.
- Normalmente o fatorial é definido na matemática de forma recursiva:
  - 0! = 1
  - n! = n x (n-1)!
- Esse tipo de relação é chamada de “relação de recorrência”: o fatorial é definido em termos de si próprio , mas há um caso base utilizado para “parar” a avaliação da função.
- Exemplo:
  - 4! = 4 x 3!
  - 3! = 3 x 2!
  - 2! = 2 x 1!
  - 1! = 1 x 0!
  - 0! = 1 (caso base).

Recursividade

- Exemplo: fatorial.
- Exemplo de implementação recursiva do fatorial:

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

- Exercício: rastreie a execução desse código supondo que se deseja calcular o fatorial do número 6.
- Note que o “if” é de extrema importância nesse código, já que ele serve para fazer as chamadas recursivas pararem no caso base.

Recursividade

- Vantagens x Desvantagens

```
long long fat(int n) {
        long long  resp = 1;
        for(int i= n;i >= 2;i--)
                resp *= i;
        return resp;
}
```

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

- Quais são as vantagens/desvantagens da versão recursiva?

* Note que utilizei *long long* para evitar overflow!

Recursividade

- Vantagens x Desvantagens

```
long long fat(int n) {
        long long  resp = 1;
        for(int i= n;i >= 2;i--)
                resp *= i;
        return resp;
}
```

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

- Quais são as vantagens/desvantagens da versão recursiva?
- Vantagens: código elegante; mais simples de entender; por ser mais simples, talvez ele pode ser otimizado pelo programador com maior facilidade utilizando algumas técnicas (que serão vistas mais para frente).
- Desvantagens: eficiência (as chamadas de função são empilhadas e realizadas várias vezes), pode consumir mais memória, ...

Recursividade

- No caso do código do fatorial, como o cálculo dele é muito simples, a diferença de performance entre a versão iterativa e a recursiva não é muito grande.
- Por exemplo, para se calcular o fatorial do número 10000 (que gera até um overflow no long long) 100000 vezes, a versão recursiva gasta aproximadamente 6 segundos enquanto a iterativa gasta 4 segundos *.
- * Em um computador Core 2 Duo E7500 e compilado com o g++ com otimização O3.

Recursividade

- Exercícios:
  - Construa uma função recursiva que conta quantos dígitos um número positivo tem.
  - Desenvolva uma função recursiva que recebe um arranjo de números inteiros (e o tamanho do arranjo) e retorna o produtório dos mesmos.
  - Construa uma função que recebe como parâmetro dois números (a,b) e, então, retorna o somatório dos números inteiros entre a e b (inclusive).

Recursividade

- Outro exemplo: conversão para binário
- Como poderíamos converter imprimir um número positivo em binário?
- Ex: 22 → 10110
- Uma solução utilizando divisões:
- Os dígitos binários podem ser extraídos tirando-se o resto da divisão do número por 2 e, então, dividindo-o por 2...
- Ao dividir um número binário por 2, eliminamos o bit menos significativo do número (ou seja, o “deslocamos” para a direita).
- Há um problema no código abaixo... qual é?
- Rastreie o código para n = 25

```
void printBinIterativo(int n) {
	int digitos = 0;
	while(n != 0) {
		cout << n%2;
		n/=2;
	}
}
```

Recursividade

- Solução que imprime o número do bit mais significativo para o menos significativo:

```
void printBinIterativo(int n) {
	int temp[32];
	int numDigitos = 0;
	while(n != 0) {
		temp[numDigitos] = n%2;
		n/=2;
		numDigitos++;
	}
	for(int i=numDigitos-1;i>=0;i--)
		cout << temp[i];
}
```

- Pergunta: por que temp é alocado com 32 posições? se fosse alocado com 31 posições, o código ainda estaria correto?

Recursividade

- Como seria um código recursivo para imprimir um número inteiro em binário?

Recursividade

- Como seria um código recursivo para imprimir um número inteiro em binário?

```
void printBinRecursivo(int n) {
	if (n==0)
		return;
	printBinRecursivo(n/2);
	cout << n%2;
}
```

- Exercício 1: Rastreie o código para n = 25
- Exercício 2: Como poderíamos imprimir o número do bit menos significativo para o mais significativo?
- Exercício 3: Faça um código para calcular quantos bits são necessários para representar determinado um número positivo.
- Exercício 4: faça com que a função do exercício 3 tenha apenas uma linha de código!

Recursividade

- Exercício: implemente um código recursivo que, dados um arranjo de caracteres contendo um número binário (com até 31 dígitos) e o número de dígitos desse número, retorne um número inteiro positivo contendo o valor decimal do número.

Recursividade

- Exercício: implemente um código recursivo que, dados um arranjo de caracteres contendo um número binário (com até 31 dígitos) e o número de dígitos desse número, retorne um número inteiro positivo contendo o valor decimal do número.
- Solução:

```
int bin2dec(char *numero, int tam, int begin = 0) {
	if (begin ==tam) return 0;
	return (numero[tam-begin-1] == '1') + 2*bin2dec(numero,tam,begin+1);
}


int main() {
	char numero[32];
	cout << "Digite os digitos do numero binario (ate 31 caracteres): ";
	cin >> numero;
	cout << "Em decimal: " << bin2dec(numero,strlen(numero)) << endl;
	return 0;
}
```

Recursividade

- Outro exemplo de recursividade...
- Impressão de todos os subconjuntos de n itens.
- Como seria um código para imprimir todos os subconjuntos?

Recursividade

- Impressão de todos os subconjuntos de n itens.
- Exercício: rastreie o código abaixo supondo que n = 3.
- O que aconteceria se as linhas 7 e 9 fossem trocadas?

```
void printCombinacoes(bool estados[],int n, int begin) {
	if (begin == n) {
		for(int i=0;i<n;i++)
			cout << estados[i];
		cout << endl;
	} else {
		estados[begin] = false;
		printCombinacoes(estados,n,begin+1);
		estados[begin] = true;
		printCombinacoes(estados,n,begin+1);
	}
}
```