Recursão

Prof. Karl Apaza Agüero

Funções

- Recursos para modularizar o código
- Uma função recebe uma lista de argumentos (parâmetros), executa uma tarefa específica e pode retornar ou não um resultado
- Exemplos de funções em C são as funções de entrada/saída

```
// Corpo da função
int funcao1(/*parâmetros*/)
{
   // ...
   // bloco de instruções
   // ...
   return /*valor*/;
}

// Programa principal
int main()
{
   //...
   funcao1(/*parâmetros*/);
   //...
   return 0;
}
```

Funções

- Exemplo 1

```
#include <stdio.h>

// Definição da função soma
int soma(int a, int b) {
    return a+b;
}

// Função main
int main(){
    int a=5, b=9, c=10;
    int soma1=soma(a,b);
    printf("%d\n", soma1);
    int soma2=soma(soma1,c);
    printf("%d\n", soma2);
    return 0;
}
```

Saída:

```
14
24
```

Funções

- Exemplo 2

```
#include <stdio.h>
int i;
void soma1(){
  i++;
  printf("soma1 : i = %d\n", i);
}
void sub1(){
  int i = 10;
  i--;
  printf("sub1 : i = %d\n", i);
}
int main(){
  i = 0;
  soma1() ;
  sub1() ;
  printf("main : i = %d\n", i);
  return 0;
}
```

Saída:

```
soma1 : i = 1
sub1 : i = 9
main : i = 1
```

O que é uma recursão?

- Resolve um problema complexo a partir de uma versão menor do problema.
- Chamamos de função recursiva quando uma função chama a si mesma.
- Exemplo: Fatorial de um número N

$$N! = N*(N-1)*(N-2)*...*3*2*1$$

$$N! = N*(N-1)!$$
$$(N-1)! = (N-1)*(N-2)!$$

O que é uma recursão?

- Fatorial de N

$$3! = 3 * 2!$$
$$2! = 2 * 1!$$
$$1! = 1 * 0!$$
$$0! = 0 * -1!$$

Quando parar?

Critério de parada

$$1! = 1$$
$$0! = 1$$

Como implementar ?

- Comece pelo critério de parada para n<=1

1! = 1
0! = 1

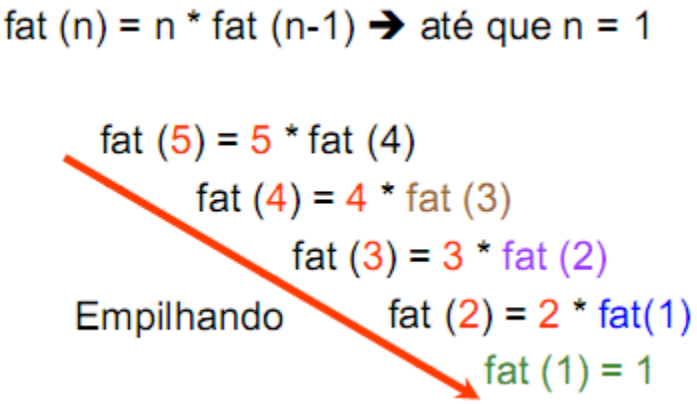
```
int fatorial(int n) {
    if(n <= 1){
        return 1;
    }
}
```

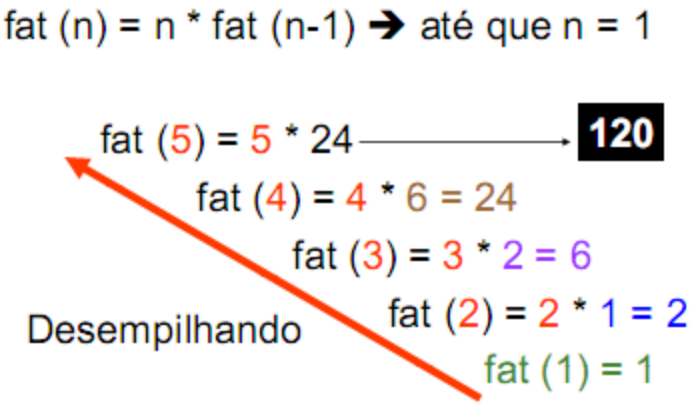
Como implementar ?

- Implemente a recursão para n>1

n!=n * (n-1)!

```
int fatorial(int n) {
   if(n <= 1){
      return 1;
   }
   return n*fatorial(n-1);
}
```





Problema

- Descrição
  - Calcular o fatorial de vários números de entrada usando recursão.
- Entrada
  - Vários inteiros representando, cada um, um caso de teste. A entrada termina com um número negativo (n<0).
- Saída
  - Para cada caso de teste, uma linha com o resultado do fatorial.

Solução

```
#include <stdio.h>

int fatorial(int n) {
   if(n <= 1)
        return 1;
   return n * fatorial(n-1);
}
```

```
int main() {
   int n;
   scanf("%d", &n);
   while(n > -1) {
     printf("%d\n", fatorial(n));
     scanf("%d", &n);
   }

   return 0;
}
```

Problema

- Descrição
  - Calcular x^n usando recursão
- Entrada
  - Dois números inteiros x e n (n>0)
- Saída
  - O resultado de x^n

Solução

```
#include <iostream>
using namespace std;

int elevar(int x, int n) {
  if (n <= 1){
    return x;
  }
  return x * elevar(x,n-1) ;
}
```

```
int main(){
  int x, n;
  cin>>x>>n;
  cout<<elevar(x,n)<<endl;
  return 0;
}
```

Problema

- Descrição
  - Implementar uma busca binária recursiva
- Entrada
  - Um inteiro representando o número procurado, um inteiro N representando o tamanho do vetor e N elementos inteiros do vetor em ordem crescente
- Saída
  - “SIM” se o número está no vetor, “NAO” caso contrário.

Solução

```
#include <iostream>
using namespace std;

int busca_binaria(int chave, int v[], int li, int ls){
  if(li>ls) return 0;

  int m=(li+ls)/2;
  if(chave==v[m]) return 1;

  if(chave<v[m]) ls=m-1;
  else           li=m+1;
  return busca_binaria(chave, v, li, ls);
}
```

```
int main(){

  int chave, n;
  cin>>chave>>n;
  int v[n];
  for(int i=0; i<n; i++)
    cin>>v[i];

  if(busca_binaria(chave, v, 0, n-1)==1)
    cout<<"SIM"<<endl;
  else
    cout<<"NAO"<<endl;

  return 0;
}
```