Modularização de Programas (Funções em C)

SSC 304 - Introdução à Programação para Engenharias

Profa. Dra. Elisa Yumi Nakagawa

Slides inicialmente preparados pela Profa. Dra. Simone Souza

FUNÇÕES

- Linguagens de programação têm à sua disposição várias funções pré-definidas:
- EXEMPLO (Linguagem *C*)
  - pow(X,2) - quadrado de X
  - sqrt (X) - raiz quadrada de X
  - sin (X) - seno de X

FUNÇÕES

- Linguagens de programação têm à sua disposição várias funções pré-definidas:
- EXEMPLO (*Linguagem c*)
  - pow (X,2) - quadrado de X
  - sqrt (X) - raiz quadrada de X
  - sin (X) - seno de X

FUNÇÕES

- Linguagens de programação têm à sua disposição várias funções pré-definidas:
- EXEMPLO (*Linguagem C*)
  - pow (X,2) - quadrado de X
  - sqrt (X) - raiz quadrada de X
  - sin (X) - seno de X

FUNÇÕES

- Funções Pré Definidas podem ser usadas diretamente em expressões:

- Exemplo:

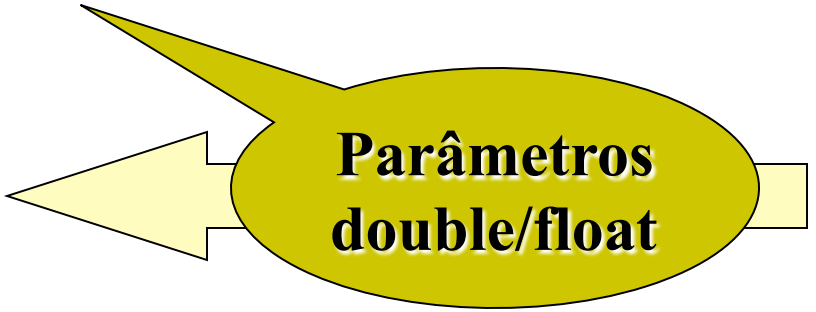
```
h= sqrt (a + pow (y,2) + 2 * sin (y))
```

FUNÇÕES

- Funções Pré Definidas podem ser usadas diretamente em expressões:
- Exemplo:

```
h= sqrt (a + pow(y,2) + 2 * sin (y))
```

Esses parâmetros devem ser valores conhecidos



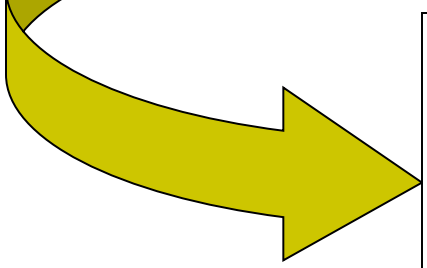
FUNÇÕES

Se houver necessidade, o programador pode **definir** suas próprias **FUNÇÕES**

```
inteiro fat(inteiro x)
inicio
inteiro i,f=1;
para i=x até 1 faça
    f=f*i;
fim-para;
  retorna (f);
fim.
```

```
int fat(int x){
   int i, f=1;
   for(i=x; i>1;i--){
        f=f*i;
   }
   return f;
}
```

```
inteiro fat(inteiro x)
inicio
inteiro i,f=1;
para i=x até 1 faça
    f=f*x;
fim-para;
  retorna (f);
fim.
```



```
int fat(int x){
   int i, f=1;
   for(i=x; i>1;i--){
        f=f*i;
   }
   return f;
}
```

```
inteiro fat(inteiro x)
inicio
inteiro i,f=1;
para i=x até 1 faça
     f=f*x;
fim-para;
  retorna (f);
fim.
```

```
int fat(int x){
   int i, f=1;
   for(i=x; i>1;i--){
        f=f*i;
   }
   return f;
}
```

UTILIZAÇÃO DA FUNÇÃO FATORIAL - Exemplo

- Dados dois números N e K, calcular a Combinação

$$C_{N,K} =$$

```
Algoritmo fat
inteiro fat(inteiro x)
inicio
   inteiro i,f=1;
   para i=x até 1 faça
     f=f*i;
   fim-para;
  retorna (f);
fim.

Algoritmo principal.
inicio.
   inteiro n,k,c;
   leia(n,k);
   c = fat(n)/(fat(k)*fat(n-k));
  escreva(c);
fim.
```

Linguagem Algoritmica

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

int fat (int x){
 int i,f=1;
 for(i=x;i>1;i--){
        f=f*i;
 }
 return f;
}

int main(){
  int n,k;
  float c;

  printf("FORNECA O VALOR DE N: "); scanf("%d",&n);
  printf("FORNECA O VALOR DE K: "); scanf("%d",&k);
  c= (float) fat(n)/(fat(k)*fat(n-k));
  printf("C(%d,%d)=%.2f",n,k,c);

return 0;
}
```

Return

- Funções retornam um resultado que deve ser do mesmo tipo para o qual a função foi declarada anteriormente.

<**tipo_retornado**> <nome_função>(<lista_dos_parametos>)

int fatorial (int n)

- Comando **return** é responsável por encerrar a execução da função e retornar o valor daquele tipo.

return result;

Return

- Se um tipo não é especificado para uma função, o tipo **int** será o *default* da função.

```
int all_add( int a, int b)
{
    int c;
    ....
    return  (a+b+c);
}
```

⇔

```
 all_add( int a, int b)
{
    int c;
    ....
    return  (a+b+c);
}
```

Return

- Valor retornado pela função é convertido, se necessário, para o tipo adequado.

```
float add( int a, int b)
{
    int soma;
    soma  = a+b;
    return  soma;
}
```

Return

- Recomenda-se que a função tenha um único **return** visando facilitar a compreensão da função pelo programador.
- Se não for possível ou se o código se tornar mais fácil de entender, o uso de mais de um **return** é permitido.

Return

- **Exemplo:**

```
int absoluteValue (int x) {
    if(x>=0)
        return x;
    else
        return –1*x;
}
```

Void

- Palavra reservada **void** na declaração de uma função indica que se trata de uma função que não retorna nenhum valor.
- Uso de **void** ao invés de uma lista de parâmetros, indica que a função não utiliza argumentos (lista de parâmetros).
- Trechos abaixo são equivalentes.

void func () ⇔ void func (void)

Declarações de funções

- Programas maiores teriam grande quantidade de linhas inicialmente dedicadas às funções até se chegar à função main().
- Além disso, ficaria cada vez mais difícil manter as funções na ordem correta de acordo com sua chamada pelo programa principal.
- Linguagem C permite que os **protótipos de funções** sejam declarados antes do programa principal.

Declarações de funções

- **Protótipo da função** indica ao compilador o número e o tipo de argumentos que devem ser passados para a função e o tipo de valor que deve ser retornado pela função.

<tipo_retornado> <nome_função>(<lista_dos_parametos>);

- Lista dos parâmetros apresenta os tipos separados por vírgula, onde os identificadores são opcionais.

float func(int, float); OU float func(int x, float y);

Programa para verificar se um número é *primo*

```
inteiro primo(inteiro n)
inicio.
inteiro c, i;
c = 1;  //true
i = 2;  //divisor do número
 enquanto ( (i < n ) E c )então
       se (n % i == 0) então //resto da divisão for zero
  c = 0;
       senão
       i = i +1;
       fim-se;
   fim-enquanto;
retorne(c);
fim.
```

```
algoritmo principal
inicio.
inteiro n;
   leia (n);
   se primo(n) então
       escreva("O Numero e Primo" );
   senão
       escreva("O Numero nao e Primo");
   fim-se;
fim.
```

```
#define<stdio.h>
#define<stdlib.h>

int primo(int);

int main(){
  int n;
  printf("forneca o numero: "); scanf("%d",&n);
  if (primo(n))
    printf("%d eh primo",n);
  else
    printf("%d nao eh primo",n);
  return 0;
}

int primo(int n){
 int c, i;
 i=2;
 c=1;
 while((i<n) && (c)){
  if(n%i == 0)
      c=0
  else
      i = i+1;
 return c;
}
```

```
#define<stdio.h>
#define<stdlib.h>

int primo(int);
```

Declaração da função

```
int main(){
  int n;
  printf("forneca o numero: "); scanf("%dz",&n);
  if (primo(n))
    printf("%d eh primo",n);
  else
    printf("%d nao eh primo",n);
  return 0;
}
```

```
int primo(int n){
 int c, i;
 i=2;
 c=1;
 while((i<n) && (c)){
  if(n%i == 0)
      c=0
  else
      i = i+1;
 return c;
}
```

Definição da Função