Modularização

““A primeira condição para se realizar alguma coisa, é não querer fazer tudo ao mesmo tempo”

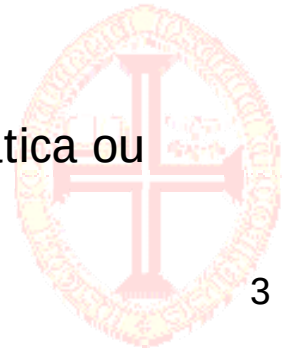
Provérbio popular

Sumári0

- Modularização
- Procedimentos
- Funções
  - Função na linguagem C
  - Expressões computacionais
  - Variáveis locais e globais
- Biblioteca de Funções

Técnicas de desenvolvimento de software

- Programação tradicional
  - Anos 60
  - Utilização de saltos incondicionais (goto)
- Programação modular
  - Anos 70
  - Divisão do programa em módulos
- Programação estruturada
  - Anos 80
  - Estruturas básica
    - Sequencial
    - Selecções
    - Repetição
- Programação orientada a objectos
  - Anos 90
  - Assemblagem de componentes (objectos)
- Programação distribuída
  - Módulos compilados em separado e integrados de forma estática ou dinâmica

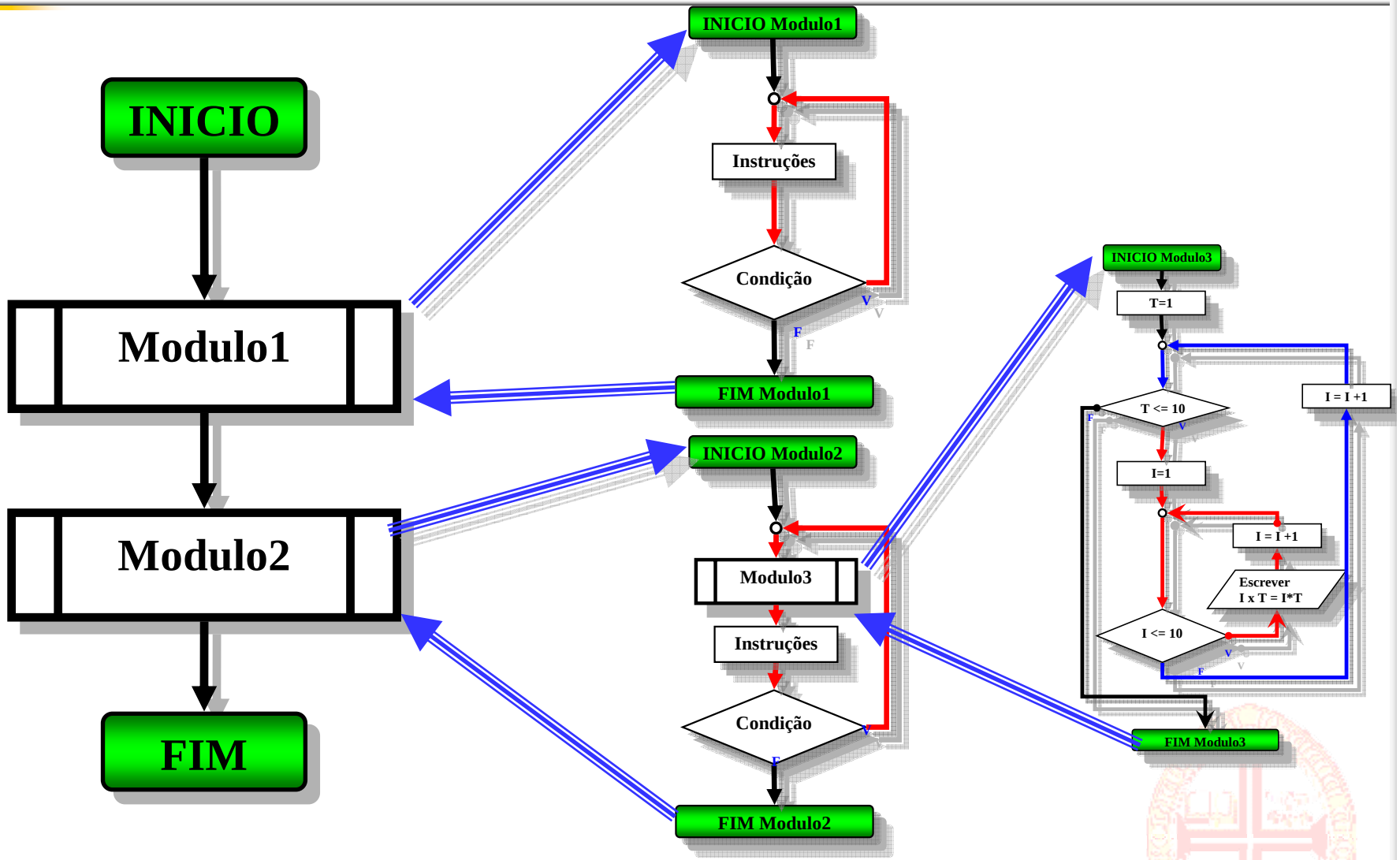


Modularização

- Vantagens
  - Fabricante
    - Trabalho em equipa
    - Investigação aplicada
    - Teste
  - Utilizador
    - Sistemas personalizado
    - Sistema evolutivos
    - Substituição de componentes

Programar consiste em segmentar o problema em módulos
Implementá-los, testá-los e integrá-los.

Programação estruturada



Vantagens

- Facilidade de implementação
  - Divisão do problema geral em problemas mais específicos
- Isolamento de erros
  - Estanquicidade dos módulos
- Facilidade de teste
  - Teste especializados para cada componente
- Facilidade de manutenção
  - Facilidade identificação dos módulos defeituosos e correcção dos mesmos
- Desenvolvimento em equipa
  - Especialização e cooperação
- Reutilização de código
  - Reutilização de módulos.

Procedimentos

funções que desempenham tarefas

Exemplo de procedimento

- Construa um programa que faça a seguinte impressão no ecrã

Ecrã

```
**********
Numeros entre 1 e 5
**********
1
2
3
4
5
**********
```

Programa

```
int main(int argc, char* argv[]){
  int i;
  for( i =1; i <= 20 ; i++)
     printf("*");
  printf("\n");

  printf("Numeros entre 1 e 5\n");

  for( i =1; i <= 20 ; i++)
     printf("*");
  printf("\n");

  for( i=1; i <= 5 ; i++)
     printf("%d\n", i);

  for( i =1; i <= 20 ; i++)
     printf("*");
  printf("\n");
}
```

Exemplo procedimento

linha

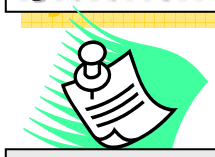
```
**********
```

procedimento Linha

```
linha()
{
 int i
 for(i=1; i <= 20 ; i++)
   cout << "*";
 cout << endl;
}
```

Programa

```
main()
{
 int i
 linha();
 cout <<"Numeros entre 1 e 5" << endl;
 linha();
 for( i=1; i <= 5 ; i++)
   cout << i << endl;
 linha();
}
```



Um programa em C++ tem de possuir sempre uma função main, idependentemente do número e variedade de funções que o programa contenha

Exemplo procedimento

- Construa um programa que faça a seguinte impressão no ecrã

Ecrã

```
***
*****
*******
*****
***
```

main

```
main() {
  linha3x();
  linha5x();
  linha7x();
  linha5x();
  linha3x();
}
```

linha3x

```
linha3x()
{
 int i
 for(i=1; i <= 3 ; i++)
   printf("*");
 printf("\n");
}
```

linha5x

```
linha5x()
{
 int i
 for(i=1; i <= 5 ; i++)
printf("*");
 printf("\n");
}
```

linha7x

```
linha7x()
{
 int i
 for(i=1; i <= 7 ; i++)
   printf("*");
 printf("\n");
}
```

procedimentos parametrizáveis

Ecrã

```
***
*****
*******
*****
***
```

linha

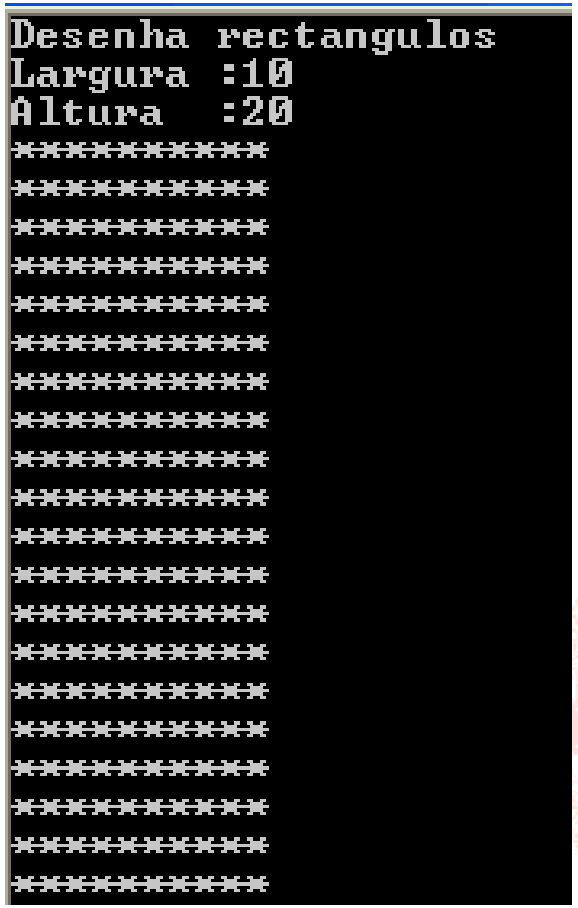
```
linhas(int num){
 int i;
 for( i=0; i < num; i++)
   printf("*");
 printf("\n");
}
```

main

```
main()
{
 linha(3);
 linha(5);
 linha(7);
 linha(5);
 linha(3);
}
```

Exercício

- Escreva um programa que imprima um rectângulo com asteriscos no monitor. A altura e largura deve ser introduzida pelo utilizador



Resolução

```
Desenha rectangulos
Largura :20
Altura  :10
********************
********************
********************
********************
********************
********************
********************
********************
********************
********************
```

linha

```
linhas(int num){
  int i;
  for( i=0; i < num; i++)
     printf("*");
  printf("\n");
}
```

Programa

```
main()
{
 int altura, largura, contador;
 printf("desenha Rectangulos\n");
 printf("Largura :");
 scanf("%d",&largura);
 printf("Altura  :");
 scanf("%d",&altura);
 for( contador =0; contador < altura; contador++ )
  linhas(largura);
}
```

Exercício

- Construa um programa que faça a seguinte impressão no ecrã

```
 Desenha Triangulos
 Altura:10
*
**
***
****
*****
******
*******
********
*********
**********
```

linha

```
linhas(int num){
  int i;
  for( i=0; i < num; i++)
     printf("*");
  printf("\n");
}
```

Programa

```
main()
{
int altura, contador;
  printf("desenha Triangulos\n");
  printf("Altura  :");
  scanf("%d",&altura);
  for( contador =0; contador < altura; contador++ )
    linhas(contador + 1);
}
```

Exercício

- Escreva um programa que imprima uma caixa com asteriscos no monitor. A altura e largura deve ser introduzida pelo utilizador

```
 Desenha Caixas
 Largura:10
 Altura:5
**********
*        *
*        *
*        *
**********
```

```
 Desenha Caixas
 Largura:40
 Altura:10
****************************************
*                                      *
*                                      *
*                                      *
*                                      *
*                                      *
*                                      *
*                                      *
*                                      *
****************************************
```

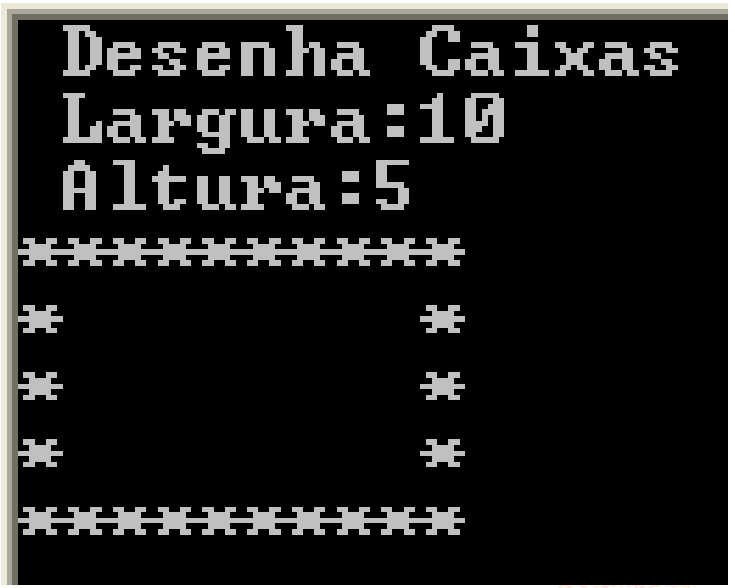
Resolução

linha

```
linhas(int num){
  int i;
  for( i=0; i < num; i++)
     printf("*");
  printf("\n");
}
```

interior

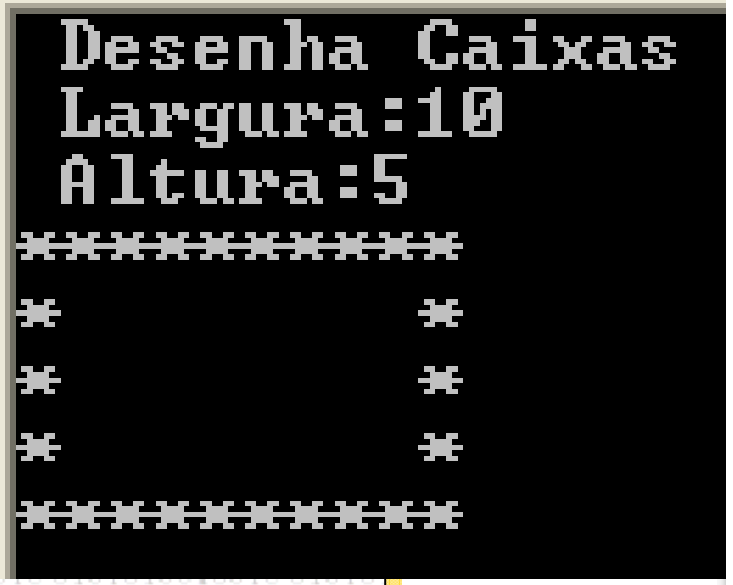
```
interior(int num)
{
  int i;
  printf("*");
  for( i=1; i < num-1; i++)
     printf(" ");
  printf("*\n");
}
```



Resolução

programa

```
main()
{
int altura, largura, contador;
  printf("desenha Rectangulos\n");
  printf("Largura :");
  scanf("%d",&largura);
  printf("Altura  :");
  scanf("%d",&altura);
  linhas(largura);
  for( contador = 1; contador < altura-1; contador++ )
      interior(largura);
  linhas(largura);
}
```



Resolução

linha

```
linhas(int num){
  int i;
  for( i=0; i < num; i++)
     printf("*");
  printf("\n");
}
```

interior

```
interior(int num)
{
int i;
  printf("*");
  for( i=1; i < num-1; i++)
     printf(" ");
  printf("*\n");
}
```

caixa

```
caixa( int largura, int altura)
{
int contador;
  linhas(largura);
  for( contador = 1; contador < altura-1; contador++ )
    interior(largura);
  linhas(largura);
}
```

```
Desenha Caixas
Largura:10
Altura:5
**********
*        *
*        *
*        *
**********
```

Resolução

programa

```
main()
{
  int altura, largura;
  printf("desenha Rectangulos\n");
  printf("Largura :");
  scanf("%d",&largura);
  printf("Altura  :");
  scanf("%d",&altura);
  caixa( largura, altura);
}
```

```
 Desenha Caixas
 Largura:10
 Altura:5
**********
*        *
*        *
*        *
**********
```

Exercício

- Construa um programa que faça a seguinte impressão no ecrã

Ecrã

```
Triângulos
Altura  :20
-
--
---
----
-----
------
-------
--------
---------
----------
###########
##########
#########
########
#######
######
#####
####
###
##
#
```

Função Linha

```
linhas(int num, char ch){
 int i;
 for( i=0; i < num; i++)
   printf("%c",ch);
 printf("\n");
}
```

Programa

```
main() {
 int altura, contador;
  printf("desenha Triangulos\n");
  printf("Altura  :");
  scanf("%d",&altura);
  for( contador =0; contador < altura/2; contador++ )
   linhas(contador + 1 , '-');
  for( contador = altura/2; contador >=0 ; contador-- )
   linhas(contador + 1 , '#');
}
```

Funções

funções que calculam valores

Programa Factorial

- Construa um programa que calcule o factorial de um número introduzido pelo utilizador
  - n! = n * (n-1) * (n-2) * * 2 * 1
  - 6! = 6* 5 * 4 *3 *2 *1

```
numero :10
10 ! = 3628800
```

Programa

```
main()
{
   int contador, numero, total=1;
   printf("numero :");
   scanf("%d",&numero);
   for( contador = numero ; contador > 1 ; contador--)
     total *= contador;
   printf("%d ! = %d",numero, total);
   getch();
}
```

Factoriais

Combinações

$$\binom{m}{n} = \frac{m!}{n!(m-n)!}$$

$$CR_m^n = \binom{m+n-1}{n} = \frac{(m+n-1)!}{n!(m-1)!}$$

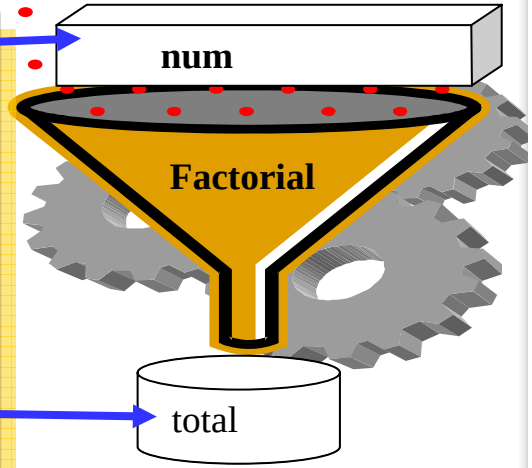
Seno

$$sen(x) = \frac{x}{1!} - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \frac{x^{11}}{11!} + \ldots$$

Funções que retornam um valor

Função Factorial

```
long factorial(int num)
{
  int contador;
  long total = 1;
  for( contador = num ; contador > 1 ; contador--)
          total *= contador;
  return total;
}
```

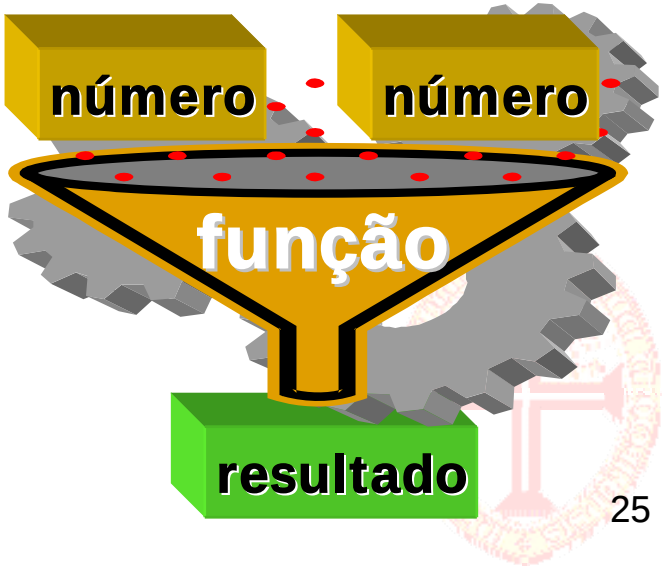


Programa

```
main()
{
   int contador, numero, total=1;
   printf("numero :");
   scanf("%d",&numero);
   printf("%d ! = %d",numero, factorial(numero));
   getch();
}
```

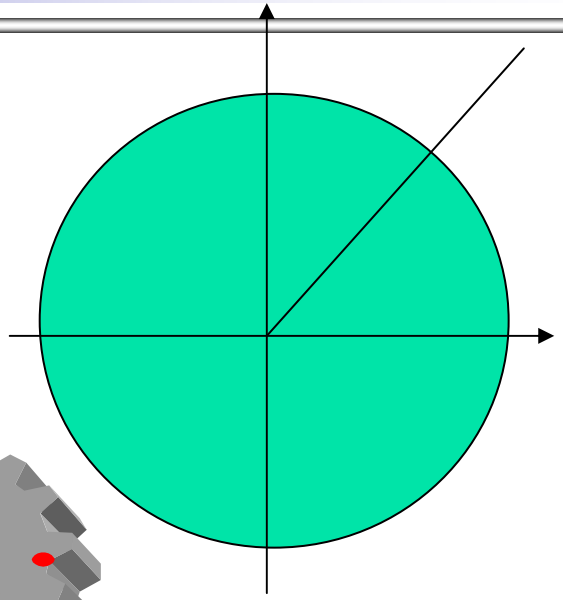
Biblioteca Math.h

- Matemáticas
  - sqrt
    - Raíz quadrada
    - x = sqrt(12)
  - exp
    - exponencial (base e)
    - x = exp(2)
  - log
    - logaritmo de base natural
    - x = log(1)
  - log10
    - logaritmo de base 10
    - x = log10(4.5)
  - pow
    - potência
    - x = pow (10 , 20)
  - abs , fabs
    - valor absoluto



Biblioteca Math.h

- Trigonométricas
  - sin
  - cos
  - asin
  - acos
  - tan
  - atan



Exercícios

Transforme as seguintes expressões matemáticas em expressões computacionais escritas em C

$$\log(N+3) * \sqrt{2\pi * N} * \left(\frac{N}{e}\right)^{N}$$

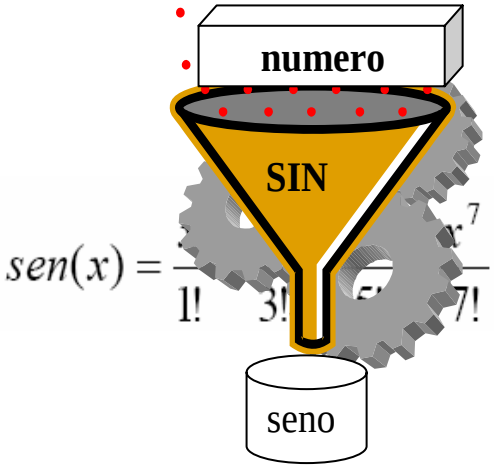
log10(N+3) * sqrt(2*3.14*N)* pow(n/2.8,N)

$$\frac{x^5 - \ln\left[e^{y-x^2}\right]}{\left|\sin(x) + arctg\left(\sqrt{y * x}\right)\right|}$$

(pow(x,5) – log(exp(y- pow(x,2))) / fabs(sin(x) + atan(sqrt(y*x)))

Exemplo Caixa Negra

- Construa um programa que calculo o seno de um nº dado pelo utilizador



$$sen(x) = \frac{x}{1!} - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \frac{x^{11}}{11!} + \ldots$$

Ecrã

```
numero:3.14
seno(3.140000) = 0.001593
```

Programa

```
main()
{
   double num;
   printf("numero:");
   scanf("%lf",&num);
   printf("seno(%lf) = %lf ", num, sin(num));
   getch();
}
```

Funções em C

Protótipo de funções em C

```
tipo_de_retorno nome_da_funcao ( [argumentos])
{
        Instruções;
        return <expressão>;
}
```

- Retorno
  - Qualquer tipo de dados
    - void (vazio)
- Nome da função
  - Regras gerais de nomes
- Argumentos
  - Zero ou mais

```
double sin( double x)
bool E_maior( double x, double y)
char getch()
void clrscr()
void gotoxy(int x, int y)
      void textcolor(int cor)
      void textbackground(int cor)
      void cprintf( . . .)
```

Void = VAZIO

Funções e Procedimentos

- Funções – calculam um valor
  - Retornam o valor calculado
  - long factorial(int numero)

- Procedimentos – Executam a sequência de tarefas
  - Terminam quando executam todas as instruções
  - void linha(int num, char ch)

Quando se omite o tipo de retorno de uma função o compilador assume que retorna o tipo int

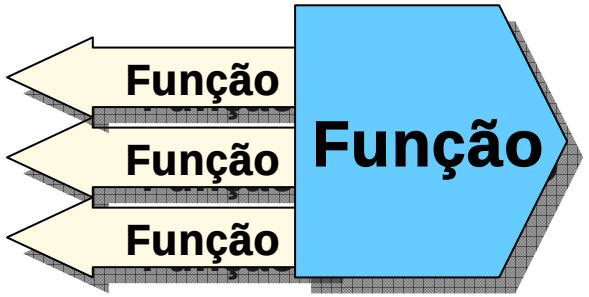
```
int main(int argc, char* argv[])
```

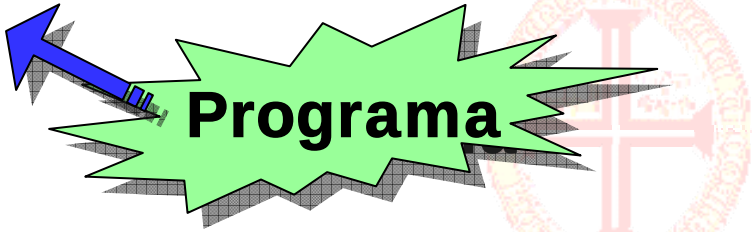
Características das funções

- Cada função tem de ter um nome único que serve para fazer a sua invocação no programa
  - Regras para o nome iguais aos nomes das variáveis
- Uma função pode ser invocada a partir de outra função
- Uma função deve realizar UMA ÚNICA TAREFA bem definida
- Uma função deve comportar-se como uma caixa negra.
  - Não interessa como funciona, o que interessa é o resultado final e sem efeitos colaterais
- O código de uma função deve ser independente do resto do programa
  - Reutilização
- Uma função pode receber parâmetros que alterem o seu comportamento
  - Adaptação
- Uma função pode retornar UM e UM SÓ valor

Exercício

- Pedir a data de admissão dois empregados e imprimir as datas na forma dd-mm-aa;
  - Robustez do código
    - Validação do input
  - Modularização
    - Validar um uma data
      - Validar um ano (>1900)
      - Validar um mês (1 –12)
      - Validar um dia (30, 31, 28, 29)
    - Escrever uma data na forma dd-mm-aa
    - Pedir duas datas e escreve-las





Valida Ano e mês

- Retorno
  - Lógico
- Nome
  - anoValido
- Argumentos
  - Ano
    - inteiro

Validação do ano

```
bool anoValido(int ano){
   if( ano < 1900 )
        return false;
   return true;
}
```

- Retorno
  - Lógico
- Nome
  - mesValido
- Argumentos
  - mes
    - inteiro

Validação do mes

```
bool mesValido(int mes)
{
 return mes >0 && mes <= 12;
}
```

Valida dia

- Retorno
  - Lógico
- Nome
  - diaValido
- Argumentos
  - Ano
    - Inteiro
  - Mês
    - inteiro
  - Dia
    - Inteiro

Validação do ano

```
bool diaValido(int ano, int mes, int dia)
{
  enum meses{JAN=1, FEV, MAR, ..., SET, OUT, NOV, DEZ};
  int ultimo_dia;
  switch(mes)
   {
    case NOV:     case ABR:     case JUN:     case SET:
          ultimo_dia=30;
          break;
    case FEV:
          if( (ano%4==0 && ano%100!=0) || ano%400==0)
            {  ultimo_dia=29;  }
          else
            {  ultimo_dia=28;  }
          break;
    default:
          ultimo_dia=31;
   }
  return (dia > 0) && (dia <= ultimo_dia);
}
```

Função para validar a data

- Retorno
  - Lógico
- Nome
  - dataValida
- Argumentos
  - Ano
    - Inteiro
  - Mês
    - inteiro
  - Dia
    - Inteiro

Validação data

```
bool dataValida(int dia, int mes, int ano)
{
 return diaValido(dia,mes,ano) &&
        mesValido(mes) &&
        anoValido(ano);
}
```

Função para escrever a data

- Retorno
  - nenhum
- Nome
  - escreveData
- Argumentos
  - Ano
    - inteiro
  - Mês
    - inteiro
  - Dia
    - inteiro

05-02-74
01-11-99
20-11-04

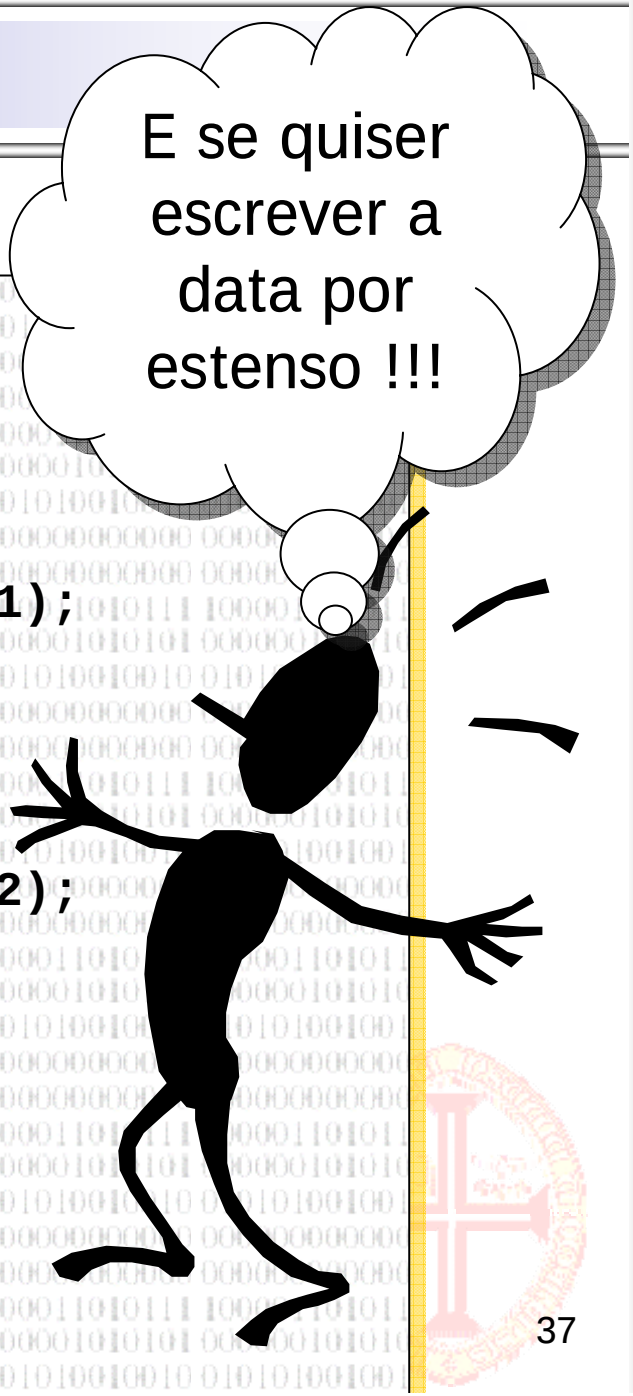
Escreve data

```
void escreveData(int dia, int mes, int ano){
 ano = ano % 100;
 printf("%02d - %02d - %02d", dia,mes,ano);
}
```

Programa principal

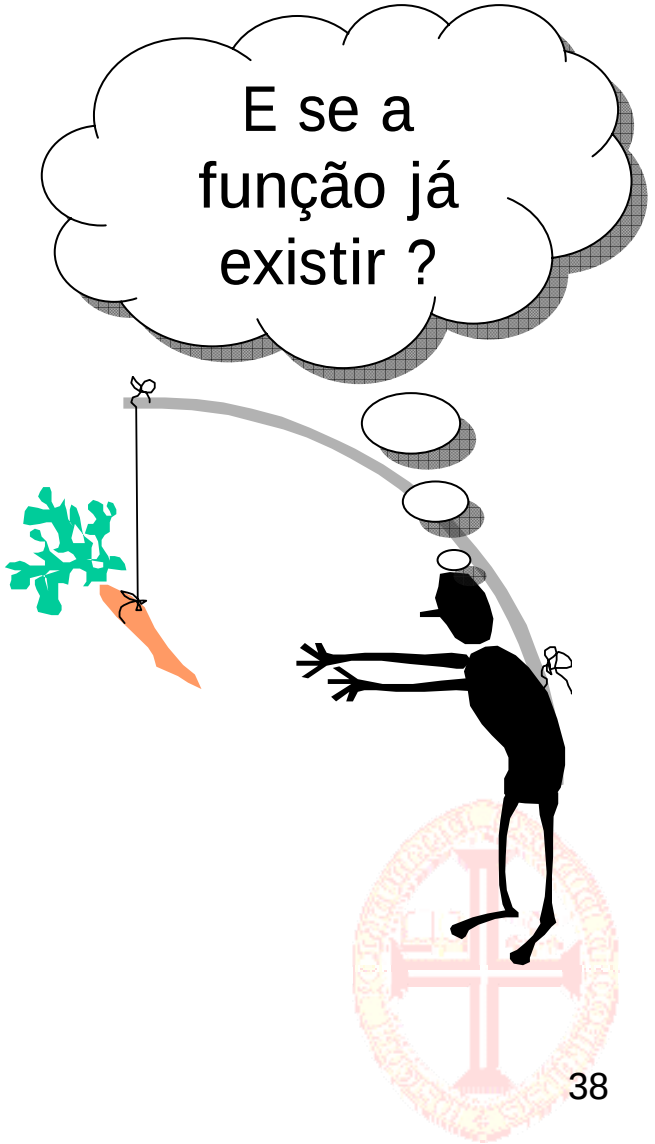
Programa principal

```
int main(int argc, char* argv[]) {
int ano1, mes1, dia1;
   int ano2, mes2, dia2;
   do{
     printf(" 1 data (aaaa mm dd):");
     scanf("%d %d %d" , &ano1, &mes1, &dia1);
   }while(!dataValida(dia1, mes1,ano1));

   do{
     printf(" 2 data (aaaa mm dd):");
     scanf("%d %d %d" , &ano2, &mes2, &dia2);
   }while(!dataValida(dia2, mes2,ano2));

   escreveData(dia1, mes1,  ano1);
   printf("\n");
   escreveData(dia2, mes2,  ano2);
  getch();
   return 0;
}
```



Conselhos úteis

- Cada função deve fazer apenas uma coisa.
- Identificar o tipo de retorno
- Escolher um nome elucidativo da sua finalidade
- Identificar os argumentos
- Encontrar um algoritmo que transforme os argumentos no retorno.
- Testar a função



Tópicos especiais em funções

Escopo dos símbolos

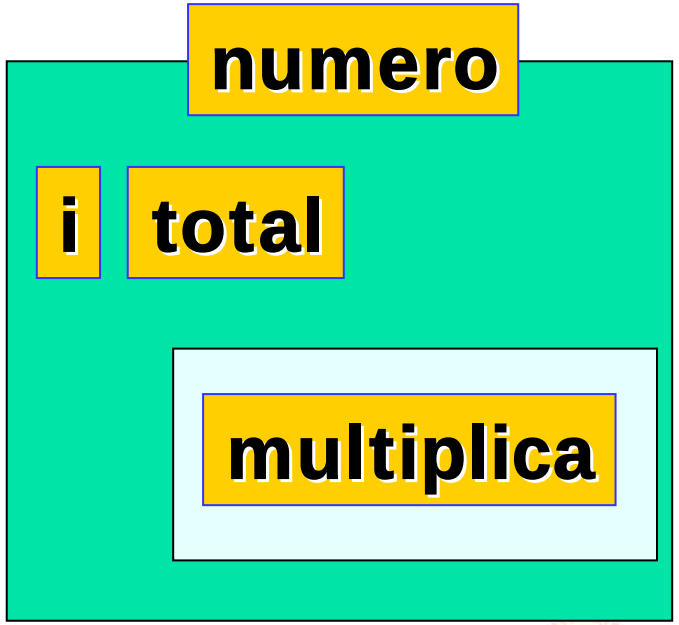
Um símbolo (função, variável, constante) é visível desde o momento em que é declarado até que o bloco onde está definido termine.

- Locais
  - Definidas dentro de um bloco
    - Função
    - ciclo
  - Criadas quando se declaram
  - Destruídas quando o bloco termina
- Globais
  - Definida fora das funções
  - Criadas quando se declaram
  - Destruídas no final do programa

Símbolos Locais

Símbolos locais

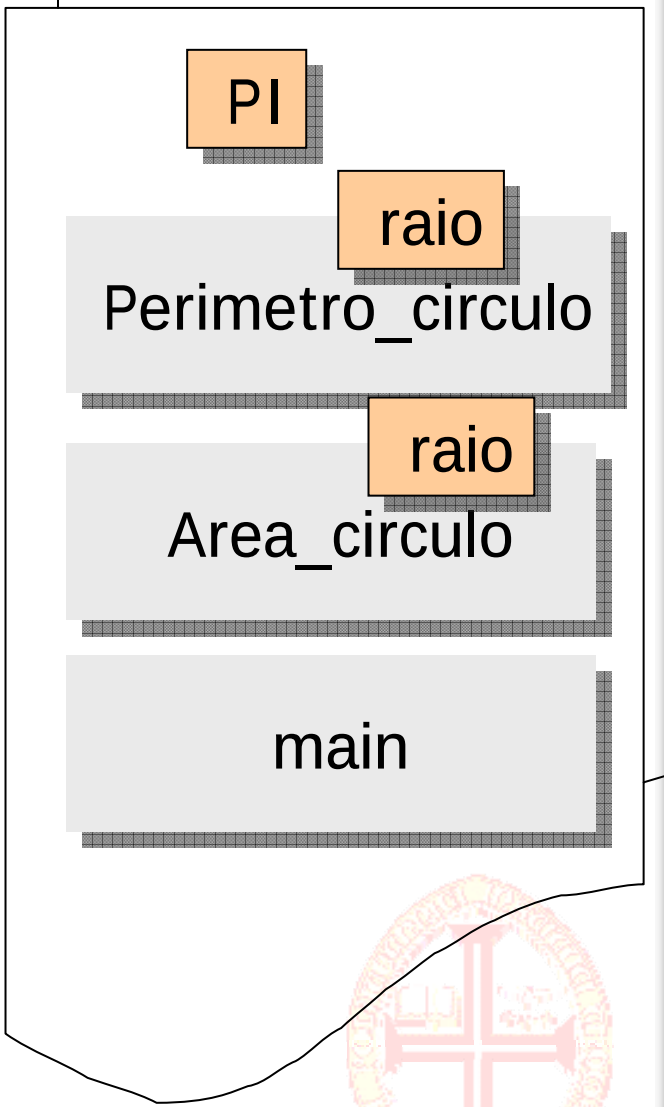
```
long factorial(int numero){
  int i;
  long total = 1;
  for( i = numero ; i > 1 ; i--)
  {
          int multiplica = i
          total *= multiplica;
  }
  return total;
}
```



Símbolos Globais

Simbolos globais

```
const double PI=3.1415;
//-----------------------------------
double perimetro_circulo(double raio)
{
  return 2 * PI * raio;
}
//-----------------------------------
double area_circulo(double raio)
{
  return  PI * raio * raio;
}
//-----------------------------------
main()
{
 . . .
}
```



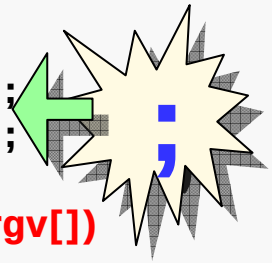
Localização das funções

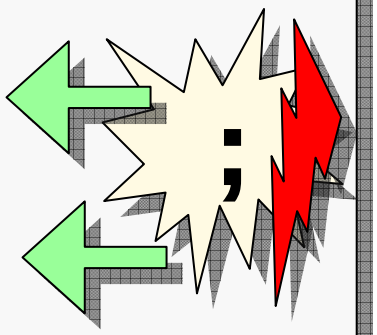
Localização das funções

- Estrutura geral de um programa em C++
  - #include <biblioteca>
  - Declaração de funções;
  - Função principal (main)
  - Implementação de funções

Exemplo

```
#include <xxxxx.h>
#include <xxxxx.h>
tipo func1( tipo arg1 , . . .);
tipo func2( tipo arg1 , . . .);

int main(int argc, char* argv[])
{
  . . .
}

tipo func1(tipo arg1,....)
{
....
}
tipo func1(tipo arg1,....)
{
....
}
```






```
#include <iostream.h>

int max(int x, int y);

int main(int argc, char* argv[])
{
  int a=5,b=10, c;
  c = max( a, b );
 . . .
 return 0;
}

int max(int x, int y)
{
  if( x> y) {
    return x;
  }
  return y;
}
```

Funções de biblioteca

Matemtica.h

```
long factorial(int n);
double exponencial(int x);
```

Matemtica.cpp

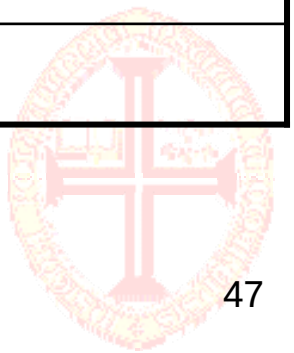
```
#include <math.h>
#include "Matematica.h“

double exponencial (int x)
{
 . . .
}
long factorial(int numero)
{
 . . .
 }
```

programa.cpp

```
#include <math.h>
##include <iostream.h>
#include "Matematica.h“

int main(int argc, char* argv[])
{
 long x=factorial(5);
 double y = exponencial(2);
}
```

Algumas Funções Padrão do c++

Prototipo	Função
double sin(double x);	sin computes the sine of the input value. Angles are specified in radians.
double acos(double x);	acos returns the arc cosine of the input value.
void randomize(void);	Initializes random number generator.
int random(int num);	random returns a random number between 0 and (num-1). random(num) is a macro defined in stdlib.h. Both num and the random number returned are integers.
double log(double x);	Calculates the natural logarithm of x.
double log10(double x);	log10 calculates the base ten logarithm of x.
double exp(double x);	Calculates the exponential e to the x.



Resumo

- Vantagens da modularização de programas
- Formato genérico de uma função
  - Tipo de retorno
  - Nome da função
  - Parâmetros de passagem
- Instrução return
- Funções como **caixas negras**
- Funções independentes do programa
  - Bibliotecas de funções
- Algumas Funções do C++

Exercício

Exercício:

Calcule o valor de Sen(x) pela formula abaixo indicada.

$$sen(x) = \frac{x}{1!} - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \frac{x^{11}}{11!} + \ldots$$

Exercício

Construa um programa que apresente as seguintes opções até que o utilizador escolha 'o'.