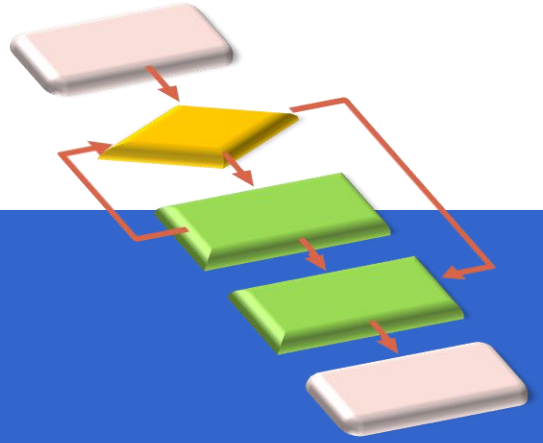
INSTITUTO DE COMPUTAÇÃO



Algoritmos II

MODULARIZAÇÃO
Linguagem C

Prof.ª Vanessa de Oliveira Campos

Em C

- Um subprograma deve ser declarado antes de ser usado.

  <tipo da subprograma> <nome subprograma> ([<lista de parâmetros formais>|void])

  - Subprograma que não precisa devolver valor: tipo **void**.
  - Subprograma sem parâmetros: **( )** ou **(void)**.

Em C

Exemplos:

- dois subprogramas sem devolução de valores e sem parâmetros:

```
void funcao1( )
void funcao2(void)
```

- um subprograma sem devolução de valores com dois parâmetros:

```
void variasletras(int vezes, char carac)
```

- um subprograma com devolução de um valor inteiro e dois parâmetros:

```
int acha_maior(int num1, int num2)
```

Em C

- Para devolver o valor de uma função:

  return <valor | variável | constante | expressão | função>;

- Subprogramas de tipo **void**:
  - executados como procedimentos;
  - chamados como comandos.

Em C

Exemplo:

```
void leia_vetor (int vet[], int qtd)
{
     int i;
     for (i = 0; i < qtd; i++)
     {
          printf("\nElemento [%d]: " , i + 1);
          scanf ("%d", vet + i);
     }
}
```

- Chamada do subprograma:

```
leia_vetor(vetor, MAX);
```

Em C

Exemplo:

```
int soma_vetor(int vet[], int qtd)
{
      long soma = 0;
      int i;
      for (i = 0; i < qtd; i++)
            soma += vet[i];
      return soma;
}
```

- Chamada do subprograma:

```
printf("Soma dos elementos do vetor: %d",soma_vetor(vetor, MAX));
```

Em C

Exemplos:

- vetor como parâmetro:

```
void mostra_vetor(int vetor[ ], int qtd)
```

```
void mostra_vetor(int *vetor, int qtd)
```

Em C

- Um protótipo de um subprograma é uma pré-declaração e informa para o compilador seu tipo, seu nome e sua lista de parâmetros.

Exemplos:

```
void mostra_menu( void )
```

```
float maior( float, float )
```

```
int fatorial( int )
```

```c
#include <stdio.h>
int ndivisores(int); // prototipo de subprograma
int main () // programa principal
{
    int num;
    printf("Número a processar(positivo): ");
    scanf("%d", &num);
    if (num > 0)
        printf("Número %d tem %d divisor(es)", num, ndivisores(num));
    else
        printf("Número inválido. ");
    return 0;
}
int ndivisores (int numero) // função que calcula a quantidade de divisores
{
    int i, lim, cont = 1; // variáveis locais / contabiliza o número 1
    lim = numero / 2; //exceto o próprio, só tem divisores até a metade do número
    for (i = 2; i <= lim ; i++)
        if (numero % i == 0) // se é divisor, conta
            cont++;
    if (numero > 1)
        cont++; // contabiliza o próprio número
    return cont; // devolve valor e encerra o processamento
}
```

```c
#include <stdio.h>
#include <stdlib.h>
void complexo2 (float *r, float *t); /* definicao do prototipo */
void main ()
{
   float a, b;
   printf ("Entre com um numero complexo (2 números inteiros): ");
   scanf("%f %f", &a, &b);
   complexo2 ( &a, &b);
   printf("O quadrado do número complexo é %f + i %f\n", a, b);
}
void complexo2 (float *r, float *t)
{
   float real;
   real = (*r * *r) - (*t * *t);
   *t = 2 * *r * *t;
   *r = real;
}
```

Exercícios de Fixação

1) Faça o algoritmo de uma função que receba três valores numéricos inteiros e retorne aquele de maior valor.

Exercícios de Fixação

2) Faça o algoritmo de uma função lógica que receba três valores inteiros e retorne VERDADEIRO se eles podem ser os valores dos lados de um triângulo, ou FALSO em caso contrário.

Exercícios de Fixação

3) Faça o algoritmo de um procedimento que troque entre si os valores das duas variáveis reais passadas como parâmetro.