Universidade Federal do Piauí
Campus SHNB – Picos -PI
Curso de Sistemas de Informação

MODULARIZAÇÃO
Funções em Linguagem C

Professor: Glauber Dias Gonçalves (gdgnew@gmail.com)
Slides do prof. Leonardo Pereira de Sousa

FUNÇÃO

- Na linguagem C: **Função**
  - trecho de código de um programa projetado para cumprir uma tarefa específica, que pode ou não retornar um valor
  - pode estar no mesmo arquivo ou em arquivos separados

Porque usar funções ?

- Para permitir o reaproveitamento de código já construído(por você ou por outros programadores);
- Para evitar que um trecho de código que seja repetido várias vezes dentro de um mesmo programa;
- Para permitir a alteração de um trecho de código de uma forma mais rápida. Com o uso de uma função é preciso alterar apenas dentro da função que se deseja;

Porque usar funções ?

- Para que os blocos do programa não fiquem grandes demais e, por consequência, mais difíceis de entender;
- Para facilitar a leitura do programa-fonte de uma forma mais fácil;
- Para separar o programa em partes(blocos) que possam ser logicamente compreendidos de forma isolada.

Funções

- Sintaxe da Declaração:

  tipo nome_da_função (lista_de_parâmetros);

  - ***tipo*** é o tipo da informação retornada da função; se a função não retornar nada, seu tipo deve ser void
  - ***parâmetros***: lista de tipos (e variáveis) que serão passados como argumentos para a função; pode ser vazio

Funções

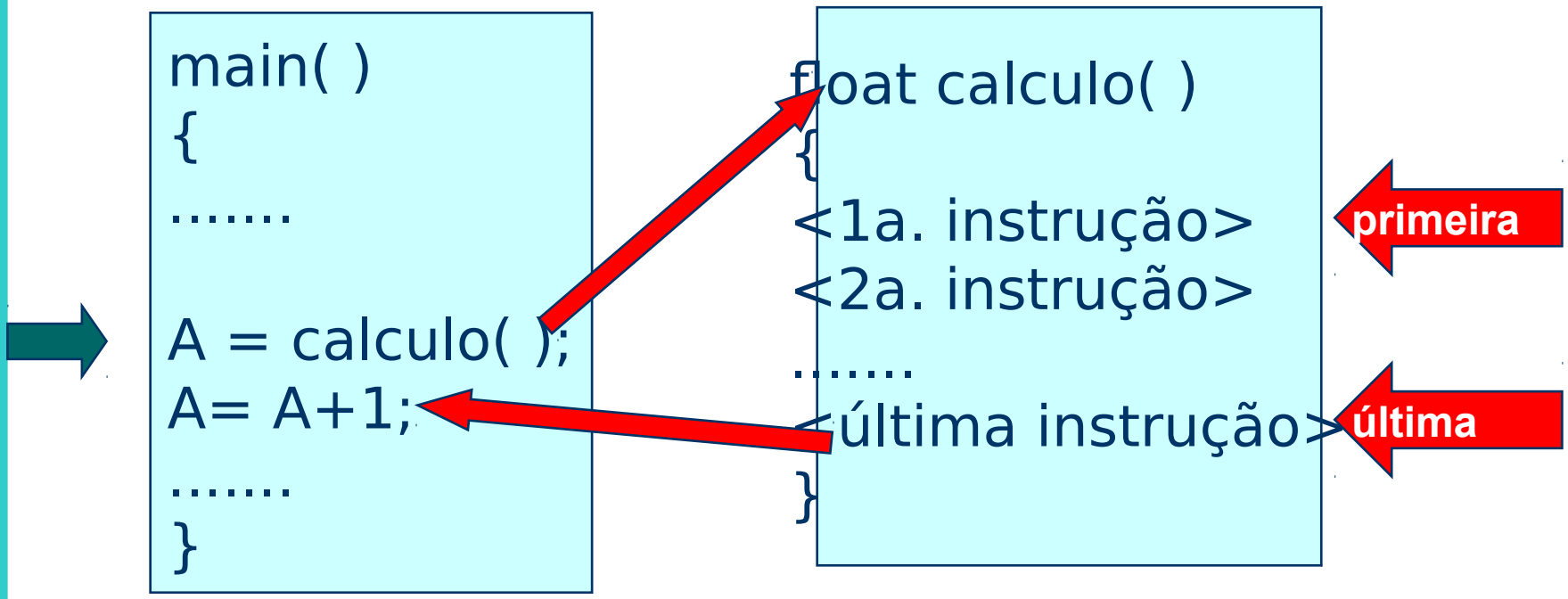
- Sintaxe da Declaração:

```
tipo nome_da_função (lista_de_parâmetros)
{
    declaração de variáveis;
    comandos;
    return (expressão);    /*opcional*/
}
```

  - a primeira linha é idêntica à declaração
  - o ***return*** serve para indicar o valor a ser retornado, se for o caso, e pode aparecer em qualquer ponto da função (não apenas no final).

Funções

- A ativação faz com que o controle seja transferido para o trecho chamado (primeira instrução) e executa até o fim do trecho (última instrução)
- Ao final da função, o controle volta para instrução seguinte à chamada



Variáveis e Funções

- Uma função pode fazer referência a variáveis:
  - declaradas localmente: variáveis locais da unidade
  - declaradas globalmente: variáveis globais do programa

Variáveis Locais

- Espaço de memória é alocado na entrada da execução da função e liberado na saída
- Podem ser declaradas dentro de qualquer bloco de código (variáveis locais ao bloco)
- Só podem ser usadas pela função à qual pertencem
- São variáveis automáticas: o valor é perdido quando a função termina e não guardam o valor anterior

Variáveis Globais

- Declaradas fora das funções
- Podem ser usadas em qualquer função
- Devem ficar no início do arquivo, antes da declaração da função main( )

Funções

- Formas de utilização
  - Sem retorno de valor
  - Com retorno de valor
- Exemplo
  - Fatorial

Fatorial (sem retorno de valor)

```
#include <stdio.h>
int n;
void fat( );
main( ) {
        scanf(“%d”, &n);
        fat( );
}
void fat( ) {
        int i, f=1;
        for (i=1; i<=n; i++)
            f=f*i;
        printf(“Fatorial = %d\n”, f);
}
```

Comando Return

- Atribui o valor de uma expressão qualquer à função, retornando este resultado para trecho do programa que chamou a função, podendo ser atribuído a uma variável, usado em algum comando ou fazer parte de alguma expressão.

- Causa uma saída imediata da função na qual ele se encontra, fazendo com que a execução retorne para o ponto do programa que chamou a função.

- Pode aparecer mais de uma vez na função (sendo que apenas um será executado a cada ativação do módulo).

Fatorial (com retorno de valor)

```
#include <stdio.h>
int n;
int fat( );
main( ) {
        int r;
        scanf("%d", &n);
        r=fat( );
        printf("Fatorial = %d\n", r);
}
int fat( ) {
        int i, f=1;
        for (i=1; i<=n; i++)
            f=f*i;
        return f;
}
```

Chamada por valor - 01

```
#include <stdio.h>
   int quadrado (int x) {
        return (x * x); }

void saudacao () {
        printf ("Ola!\n"); }

void despedida () {
        printf ("Fim do programa.\n"); }
```

Chamada por valor - 01

```
int main () {
    int numero, resultado;
    saudacao ();

    printf ("Digite um numero inteiro: ");
        scanf ("%d", &numero);
    resultado = quadrado (numero);
    printf ("O quadrado de %d eh %d.\n", numero,
    resultado);


    despedida ();
}
```

Chamada por valor - 02

```
#include <stdio.h>
/* Multiplica 3 numeros */

void mult (float a, float b, float c) {
    printf ("%6.2f",a*b*c); }

    int main ()
    {
        float x, y;
        x = 23.5;
        y = 12.9;
        mult (x, y, 3.87);
    }
```

Atividade 01

- Fazer uma aplicação para calcular a formula de Baskara:

Resolução

```
#include<stdio.h>
#include<conio.h>
#include <math.h>/*função necessaria para calcular a raiz quadrada que é sqrt()
                   e para usar o powf() elevado a um numero */

void CalculaEquacao(float a, float b, float c);
```

Resolução

```
void main()
{
 float a,b,c;

 printf ("Este programa calcula Equacoes do tipo: ax^2+bx+c=0\n");
 printf ("Digite um valor para a:");
 scanf ("%f", &a);
 printf ("Digite um valor para b:");
 scanf ("%f", &b);
 printf ("Digite um valor para c:");
 scanf ("%f", &c);

 CalculaEquacao(a,b,c); /*chama a função que calcula*/
 getch();
}
```

Resolução

```
void CalculaEquacao(float a, float b, float c)
{
 float delta, x1, x2;
 delta = pow(b,2) - 4 * a * c;
 if (a != 0) /*Se a for diferente de zero temos uma equação do 2 grau*/
 {
  if (delta >= 0)
  {
   /* Caso delta>=0 temos raizes reais(x1 e x2), calculada por: */
   x1 = (-b+sqrt(delta))/(2*a);
   x2 = (-b-sqrt(delta))/(2*a);
   printf("\nx1:%f",x1);
   printf("\nx2:%f",x2);
  }
```

Resolução

```
else
{
 printf ("Nao foi possivel calcular x1 e x2,pois sao numeros imaginarios");
}
}/*fim do ultimo if*/


  else
  {
    printf ("Esta equação não tem solução!!!");
  }
}
```

Atividades

- Usar funções para as questões 06-10 da lista de atividade02