Universidade Federal do Piauí
CSHNB

Algoritmos e Programação II

Glauber Dias Gonçalves
ggoncalves@ufpi.edu.br

Conteúdo da Aula

- Funções e Modularização (revisão Algoritmos - I)
  - Estrutura de funções
  - Escopo de variáveis local e global
  - Passagem de parâmetros por cópia de valores

FUNÇÃO

- Módulo do algoritmo com uma tarefa específica
- Objetivo: modularizar ou organizar o algoritmo
  - Estratégia: dividir para conquistar
- Pode ou não retornar um valor
- Envolve três partes básica:
  - Protótipo: no início do algoritmo
  - Declaração: especificação da função
  - Chamada: dentro de alguma função do algoritmo

PROTÓTIPO

tipo nome_da_função (lista_de_parâmetros)

- ***tipo:*** a informação retornada da função;
  - se não retornar nada, seu tipo deve ser void;

- ***nome_da_função***: mesma regra para variáveis

- ***parâmetros***: lista de tipos (e variáveis) que serão passados como argumentos para a função
  - pode ser vazio.

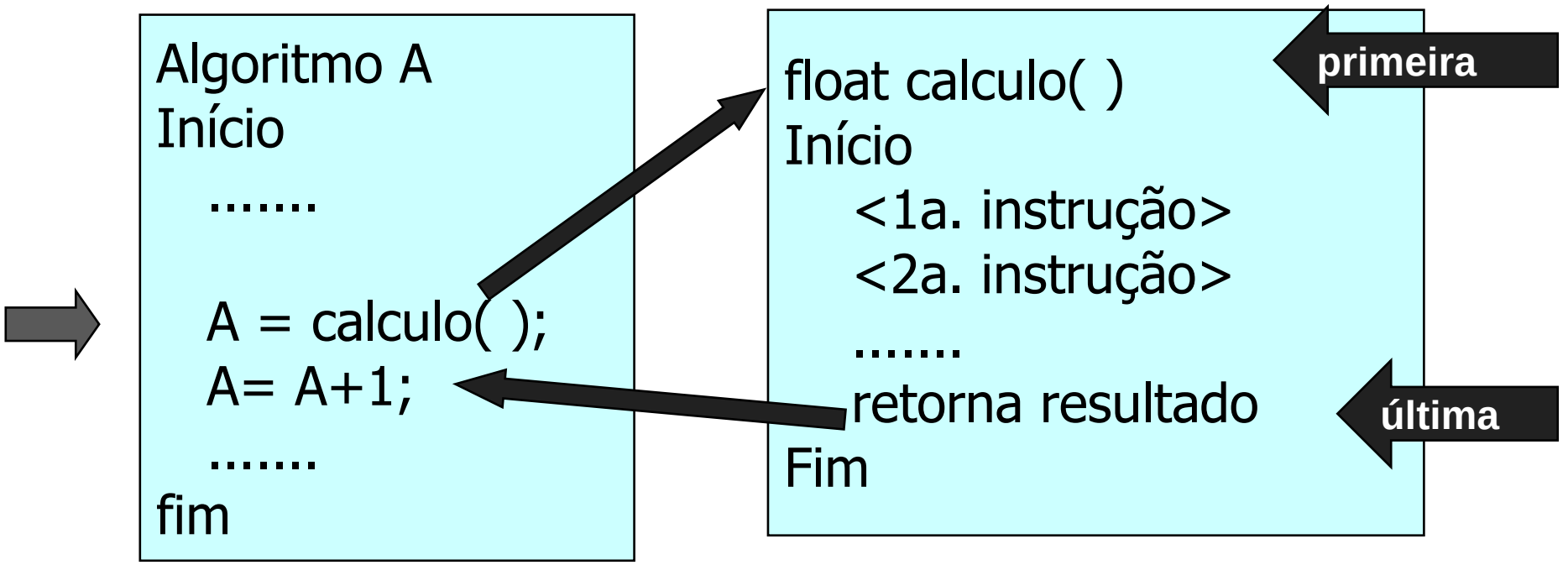
DECLARAÇÃO

```
tipo nome_da_função (lista_de_parâmetros)
    declaração de variáveis
início
    comandos
    returno (expressão)    /*opcional*/
fim
```

- a primeira linha é idêntica ao protótipo
- o ***retorno*** serve para indicar o valor a ser retornado,
  - pode aparecer em qualquer ponto da função
  - pode aparecer em mais de um ponto.

CHAMADA

- Transfere controle para o função chamada
- Executa até o fim da função
- retorna o controle de volta para o local de chamada.



ESCOPO DE VARIÁVEIS EM FUNÇÕES

- Uma função pode usar variáveis:
  - Locais: válidas apenas dentro da função onde foi declarada
  - Globais: válidas em todas as funções do algoritmo

VARIÁVEIS LOCAIS

- Espaço de memória é alocado no início da execução da função e liberado no final;
- Podem ser declaradas em qualquer parte do bloco que compõe a função;
- Só podem ser usadas pela função à qual pertencem;
- Valores são perdidos quando a função termina.

VARIÁVEIS LOCAIS

```
Algoritmo A
   real x
Início
   .......

   x ← calculo( );
   x ← x+1;
   .......
fim
```

Válida apenas dentro do Algoritmo A

VARIÁVEIS GLOBAIS

- Declaradas fora de funções;
- Podem ser usadas em qualquer função;
- Devem ficar fora do algoritmo
  - Antes da declaração da função principal

VARIÁVEIS GLOBAIS

```
float resultado

Algoritmo A
Início
   .......

   calculo( );
   escreve(resultado)
   .......
fim
```

variável global

```
inteiro calculo( )
Início
    resultado *= 10
    retorna 0
Fim
```

RETORNO DA FUNÇÃO

- Formas de utilização
  - Sem retorno de valor;
  - Com retorno de valor.

- Exemplo
  - Fatorial na linguagem C

FATORIAL (SEM RETORNO DE VALOR)

```
inteiro n

fatorial ( )

Programa testaFatorial
Início
        leia(n)
        fatorial ( )
        retorna 0
Fim
```

```
fatorial ( )
        inteiro i, fat = 1
        para i de 1 até n faça
                fat *= i
        fim_para
        escreva( fat )
fim
```

COMANDO RETORNO

- Retorna um resultado para o local de chamada da função
  - pode ser armazenado em uma variável
  - pode ser usado em algum comando ou expressão
- Finaliza a função no local do retorno
- Pode aparecer mais de uma vez na função
  - apenas um será executado a cada chamada da função

FATORIAL (COM RETORNO DE VALOR)

```
inteiro n

inteiro fatorial ( )

Programa testaFatorial
        inteiro fat
Início
        leia(n)
        fat ← fatorial ( )
        escreva( fat )
        returna 0
Fim
```

```
inteiro fatorial ( )
        inteiro i, fat = 1
        se n = 0 então
                retorna 0
        para i de 1 até n faça
                fat *= i
        fim_para
        retorna fat
fim
```

Codificação na Linguagem C

FATORIAL (SEM RETORNO DE VALOR)

```c
#include <stdio.h>

int n; //Variavel global
void fatorial(void); //Prototipo da função fatorial

int main(){
    printf("Digite o valor de n: ");
    scanf("%d",&n);

    fatorial(); //Chamada da funcao fatorial

    return 0;
}//fim da funcao main

//Criacao da funcao fatorial
void fatorial(){
    int i, fat = 1;
    for(i=1; i<=n; i++){//A funcao fatorial utiliza o valor de n
        fat *= i;
    }//fim do for
    printf("Fatorial = %d\n",fat);
}//fim da funcao fatorial
```

FATORIAL (COM RETORNO DE VALOR)

```c
#include <stdio.h>

int n; //Variavel global
int fatorial(void); //Prototipo da função fatorial

int main(){
    int resultado;//Variavel que recebe o resultado
    printf("Digite o valor de n: ");
    scanf("%d",&n);

    resultado = fatorial(); //Chamada da funcao fatorial

    printf("Resultado = %d",resultado);
    return 0;
}//fim da funcao main

//Criacao da funcao fatorial
int fatorial(){
    int i, fat = 1;
    for(i=1; i<=n; i++){//A funcao fatorial utiliza o valor de n
        fat *= i;
    }//fim do for

    return fat;//Retornando o valor da variavel fat
}//fim da funcao fatorial
```

PASSAGEM DE PARÂMETRO(S)

- Transferência de informação entre funções
  - Função principal para as demais funções e vice-versa
  - Entre as demais funções

- Utilizaremos agora: passagem de parâmetros por cópia
  - Utiliza variáveis locais e retorno de funções
  - Útil para evitar ‘variáveis globais’
  - Alternativa à passagem de parâmetro por referência
  - Não serve para vetores e matrizes

ATIVIDADE DE FIXAÇÃO

- Modificar o código da função fatorial em C com retorno de valor para receber o valor n via passagem de parâmetro.

  - Fazer rascunho no slide seguinte e testar!

ATIVIDADE DE FIXAÇÃO

```c
#include <stdio.h>

int n; //Variavel global
int fatorial(void); //Prototipo da função fatorial

int main(){
    int resultado;//Variavel que recebe o resultado
    printf("Digite o valor de n: ");
    scanf("%d",&n);

    resultado = fatorial(); //Chamada da funcao fatorial

    printf("Resultado = %d",resultado);
    return 0;
}//fim da funcao main

//Criacao da funcao fatorial
int fatorial(){
    int i, fat = 1;
    for(i=1; i<=n; i++){//A funcao fatorial utiliza o valor de n
        fat *= i;
    }//fim do for

    return fat;//Retornando o valor da variavel fat
}//fim da funcao fatorial
```

FATORIAL COM PASSAGEM DE PARÂMETRO

```c
#include <stdio.h>

int fatorial(int); //Prototipo da função fatorial

int main(){
    int resultado, n;
    printf("Digite o valor de n: ");
    scanf("%d",&n);

    resultado = fatorial(n); //Chamada da funcao fatorial

    printf("Resultado = %d",resultado);
    return 0;
}//fim da funcao main

//Criacao da funcao fatorial
int fatorial(int n){
    int i, fat = 1;
    for(i=1; i<=n; i++){//A funcao fatorial utiliza o valor de n
        fat *= i;
    }//fim do for

    return fat;//Retornando o valor da variavel fat
}//fim da funcao fatorial
```

ATIVIDADE DE FIXAÇÃO

- Faça a função soma que some dois números inteiros recebidos como parâmetros e imprima o resultado

- Faça um algoritmo que lê dois números inteiros utilize a função soma para somá-los e mostrar o resultado

ATIVIDADE DE FIXAÇÃO

- Faça uma função que recebe os parâmetros peso, altura e calcula o IMC (peso dividido pelo quadrado da altura)

- Faça um algoritmo que lê peso e altura de uma pessoa e utiliza a função acima para calcular o IMC. Se o IMC estiver acima de 25 escreva a mensagem: “acima do peso”

ATIVIDADE DE FIXAÇÃO

- Fazer uma função para mostrar as raízes de uma equação quadrática com o seguinte protótipo:

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

$$\Delta = b^2 - 4.a.c$$

$$x = \frac{-b \pm \sqrt{\Delta}}{2a}$$

SUMÁRIO

- Funções:
  - Recurso para modularizar ou organizar o algoritmo
  - Estratégia: dividir para conquistar
  - Pode ou não retornar um valor

- Envolve três passos:
  - **Protótipo**: no início do algoritmo
  - **Chamada:** dentro de alguma função do algoritmo
  - **Declaração:** trecho do algoritmo com especificação da função

- Lista de Atividade 3 (discussão na aula amanhã)