

Aula 6

Lógica de Programação e Algoritmos

Prof. Sandro de Araújo

Conversa Inicial

■ Esta aula apresentará a seguinte estrutura de conteúdo:

- Procedimento
- Função
- Declaração de uma função
- Parâmetros
- Passagem de parâmetros

- O objetivo desta aula é conhecer os principais conceitos e aplicações de procedimentos e funções e fazer uma introdução de como declará-los nas construções de algoritmos para resolver problemas computacionais

Procedimento

Procedimento

- Procedimentos são estruturas que juntam um conjunto de comandos, que são executados no momento em que são chamados
- O procedimento é identificado com o nome <identificador> acompanhado de parênteses () e pode possuir ou não parâmetros (Puga e Rissetti, 2016)

Procedimento

- Quando o programa principal chama um procedimento, por meio do seu identificador, o controle do fluxo de execução do programa passa para o procedimento e, no momento em que o procedimento finalizar a tarefa, o controle do fluxo de execução retornará ao programa principal

- Para criar um procedimento em pseudocódigo utiliza-se a seguinte estrutura:

```
procedimento <nome-de-procedimento>
[( <declarações-de-parâmetros> )]
var
    // Seção de declarações internas
início
    // Seção de comandos
fimprocedimento
```

- Na linguagem de programação C, um procedimento é uma função sem retorno e, para criar um procedimento, utiliza-se a seguinte estrutura:

```
void nome<identificador>( )
{
    // Seção de Comandos
    return; /* retorno de uma função void */
}
```

- Exemplo:

- Considere um algoritmo que somará dois números, usando um procedimento, e mostrará o resultado no programa principal

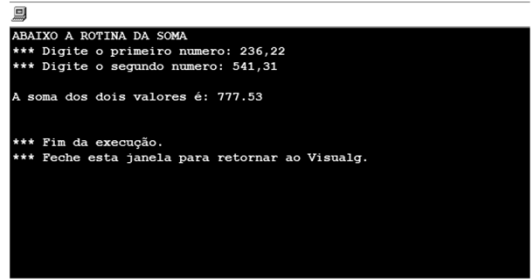
Pseudocódigo

```
algoritmo "CalculaSoma"
procedimento SOMA
var
    resultado,a,b:real
início
    escreva("ABAIXO A ROTINA DA SOMA")
    escreva("*** Digite o primeiro número: ")
    leia(a)
```

```
escreva("*** Digite o segundo numero: ")
leia(b)
resultado<-a+b
escreva("")
escreva("A soma dos dois valores
é:",resultado)
escreva("")
fimprocedimento
```

Pseudocódigo

```
(...)  
início           //programa principal  
    SOMA  
finalgoritmo
```



```
ABAIXO A ROTINA DA SOMA  
*** Digite o primeiro numero: 236,22  
*** Digite o segundo numero: 541,31  
  
A soma dos dois valores é: 777.53  
  
*** Fim da execução.  
*** Feche esta janela para retornar ao Visualg.
```

Linguagem C

```
#include <stdio.h>  
#include <conio.h>  
void SOMA()  
{  
    float resultado, a, b;  
    printf("ABAIXO A ROTINA DA SOMA");  
    printf("\n*** Digite o primeiro número: ");
```

Linguagem C

```
scanf("%f", &a);  
    printf("*** Digite o segundo número: ");  
    scanf("%f", &b);  
    resultado = a + b;  
    printf("\n A soma dos dois valores digitados é:  
    %.2f\n", resultado);  
    return;  
}
```

Linguagem C

```
#include <stdio.h>  
#include <stdlib.h>  
int main() //programa principal  
{  
    SOMA();  
    return 0;  
}
```



```
"C:\Users\Casa\Documents\Sandro\FACULDADES\UNINTER\Linguagem de Programação\Procedimento\bin\Debug\Procedimento.exe"  
ABAIXO A ROTINA DA SOMA  
*** Digite o primeiro numero: 236,22  
*** Digite o segundo numero: 541,31  
  
A soma dos dois valores digitados eh: 777.53  
  
Process returned 0 (0x0)   execution time : 39.248 s  
Press any key to continue.
```

Função

Função

- A função é um tipo especial de procedimento. Também conhecida como sub-rotina, ela é um conjunto de instruções construídas para cumprir uma tarefa específica e agrupada numa unidade

- Para criar uma função em pseudocódigo utiliza-se a seguinte estrutura:
 - função<nome-função>[(<declarações-parâmetros>)]:<tipo-de-dado>
 // Seção de declarações internas
início
 // Seção de comandos
 retorne valor_de_retorno
fimfuncao

- Para criar uma função na linguagem C, utiliza-se a seguinte estrutura:
 - <tipo> nome<identificador>
 (<tipo>parâmetro,<tipo>parâmetro, ...)
 {
 // Seção de comandos
 return valor_de_retorno;
 }

- Exemplo:
 - Considere um algoritmo que terá uma rotina da qual recebe um número do tipo inteiro e calcula o seu quadrado

Pseudocódigo

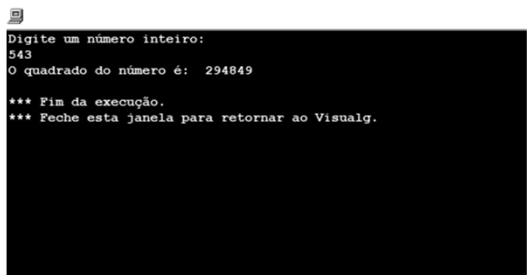
- algoritmo "CalculaQuadrado"
 - var
 n1, a, resultado: inteiro
 função calcula_quadrado(a : inteiro): inteiro
 var retorna: inteiro

Pseudocódigo

```
início  
  retorna <- a*a  
  retorne retorna  
fimfuncao
```

Pseudocódigo

```
início  
  escreva ("Digite um número inteiro: ")  
  leia (n1)  
  escreva ("O quadrado do número é: ",  
    calcula_quadrado(n1))  
finalgoritmo
```



```
Digite um número inteiro:  
543  
O quadrado do número é: 294849  
  
*** Fim da execução.  
*** Feche esta janela para retornar ao Visualg.
```

Linguagem C

```
int quadrado(int n1);  
int main()  
{  
  int número, resultado;  
  printf("Digite um número inteiro: \n");  
  scanf("%d", &número);
```

Linguagem C

```
printf("O quadrado do número é: %d",  
  quadrado(numero));  
return 0;  
}  
int quadrado(int n1)
```

Linguagem C

```
{  
  resultado = n1*n1;  
  return resultado;  
}
```

```

"C:\Users\Casa\Documents\Sandro\FACULDADES\UNINTER\Linguagem de Programação\CalculaQuadrado"
Digite um numero inteiro:
543
O quadrado do numero eh: 294849
Process returned 0 (0x0)   execution time : 2.934 s
Press any key to continue.

```

Declaração de uma função

Declaração de uma função

- Na chamada de uma função, o compilador necessita que sejam informados corretamente o tipo de retorno e os parâmetros da função, para que ele possa manipulá-los

Declaração de uma função

Indica o tipo do valor de retorno (saída).
Toda função deve ter um tipo declarado, que pode ser char, int, entre outros.

Indica o tipo e os valores de entrada que serão manipulados pela função.

```

int soma( int x, int y )
{
    int z;
    z = x + y;
    return z;
}

```

Corpo da função
Seção de comandos que serão usados para resolver o problema.

Declaração de uma função

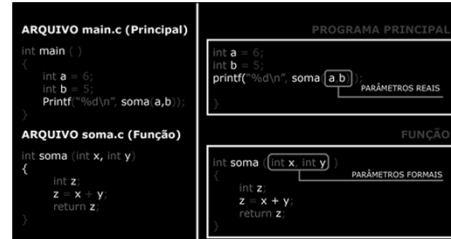
- O valor de retorno da função se dá com o comando return. Esse comando é sempre o último a ser executado por uma função, finalizando o bloco de instruções
- O tipo void representa o "sem retorno", ou seja, um retorno com conteúdo indeterminado

Parâmetros

Parâmetros

- Os parâmetros podem ser divididos em duas categorias
 - Formais: que correspondem aos parâmetros utilizados na descrição da função
 - Reais: que correspondem aos parâmetros especificados na instrução de chamada

Parâmetros



Parâmetros

- Os parâmetros formais só existem para o programa no momento da execução da função. Após sua execução eles deixam de existir
- Diferentemente, os parâmetros reais podem ser usados na chamada da função e também em outros momentos no programa principal

Parâmetros

```
int soma (int x, int y);
int main()
{
    int a = 6, b = 5, c;
    c = a - b; // Parâmetros reais usados para subtração
    printf("Soma = %d\n", soma(a,b));
    //Parâmetros reais
```

Parâmetros

```
printf("Subtração = %d\n\n", c);
system("pause");
return 0;
}
```

Parâmetros

```
int soma (int x, int y) // Parâmetros formais
{
    int z;
    z = x + y;
    return z;
}
```

```
"C:\Users\Casa\Documents\Sandro\FACULDADES\UNINTER\Linguagem de Programação
Soma = 11
Subtracao = 1
Pressione qualquer tecla para continuar. . . .
```

Passagem de parâmetros

Passagem de parâmetros

- **Por valor** – uma cópia do parâmetro é feita, um valor da expressão é calculado e o valor resultante é passado para a execução da função

Passagem de parâmetros

- **Por referência** – o endereço de um parâmetro é passado na chamada da função. Com isso, a função pode modificar a variável diretamente

Exemplo de passagem por valor

```
void soma_mais_1(int num);
int main()
{
    int a = 8;
    printf("Antes da função: a = %d\n",a);
    // Impressão de "a"
    soma_mais_1(a); // A função recebe o
    parâmetro de "a"
```

Exemplo de passagem por valor

```
printf("Depois da função: a = %d\n",a); //
Impressão de "a"
system("pause");
return 0;
}
```


Exemplo de passagem por valor

```
void soma_mais_1(int num){ // Cópia do dado de "a" em "num"
    num = num + 1;
    printf("Dentro da função: a = %d\n", num);
}
```

Exemplo de passagem por valor

```
"C:\Users\Casa\Documents\Sandro\FACULDADES\UNINTER\Linguagem de Programação\P_Valor"
Antes da funcao: a = 8
Dentro da funcao: a = 9
Depois da funcao: a = 8
Pressione qualquer tecla para continuar. . .
```

Exemplo de passagem por valor

```
void soma_mais_1(int num);
int main()
{
    int a = 8;
    printf("Antes da função: a = %d\n", a);
    // Impressão de "a"
    soma_mais_1(a); // A função recebe o parâmetro de "a"
}
```

Exemplo de passagem por valor

```
printf("Depois da função: a = %d\n", a); // Impressão de "a"
system("pause");
return 0;
```

Exemplo de passagem por valor

```
}
void soma_mais_1(int num){ // Cópia do dado de "a" em "num"
    num = num + 1;
    printf("Dentro da função: a = %d\n", num);
}
```

```
"C:\Users\Casa\Documents\Sandro\FACULDADES\UNINTER\Linguagem de Programação\P_Valor"
Antes da funcao: a = 8
Dentro da funcao: a = 9
Depois da funcao: a = 8
Pressione qualquer tecla para continuar. . .
```

- **Passagem por valor**

```
Antes da funcao: a = 8  
Dentro da funcao: a = 9  
Depois da funcao: a = 8
```

- **Passagem por referência**

```
Antes da funcao: a = 8  
Dentro da funcao: a = 9  
Depois da funcao: a = 9
```

Passagem de parâmetros

- Na passagem por valor, o parâmetro formal comporta-se como uma variável local e as alterações feitas nesta variável não terão efeito sobre o parâmetro real
- Na passagem por referência, o parâmetro formal comporta-se como se fosse uma variável global, em que todas as alterações feitas, nesta variável, são feitas no parâmetro real

