

Algoritmos I



SUBROTINAS E PASSAGEM DE PARÂMETROS

Subrotinas



- Uma subrotina (função ou procedimento) consiste em uma porção de código que resolve um problema muito específico, parte de um problema maior (a aplicação final).
- Pode ser definido como um miniprograma dentro de um programa.
- Blocos de Instruções que realizam tarefas específicas. O código de uma subrotina é carregado uma vez e pode ser executado quantas vezes forem necessárias.

Subrotinas



- Um programa em C é uma coleção de funções. Todos os programas se constroem por uma ou mais funções que se integram para criar uma aplicação. Todas as funções contém uma ou mais sentenças C e se criam geralmente para realizar uma única tarefa, tal como imprimir a tela, escrever um arquivo ou mudar a cor da tela. Pode-se declarar e executar um número quase ilimitado de funções em um programa C.

Vantagens de utilização de Subrotinas



- Economia de espaço, reduzindo repetições e tornando mais fácil a programação.
- A possibilidade de reutilizar o mesmo código, sem grandes alterações, em outros programas.
- Proporcionam um meio de dividir um projeto grande em pequenos módulos mais manejáveis.
- Ficam mais organizados.

Tipos de Subrotinas

- Procedimentos: são subrotinas que não possuem retorno. Em C/C++ são conhecidas como funções sem retorno (e será tratado desta forma nestes slides).

```
1  #include <stdio.h>
2  #include <locale.h>
3
4  void mensagem () {
5      printf("Primeira função sem retorno.");
6  }
7
8  int main()
9  {
10     setlocale(LC_ALL, "Portuguese");
11     mensagem();
12
13     return 0;
14 }
```

Função sem retorno

Chamada da função

```
C:\Users\Usuario\Desktop\Teste\bin\Debug\Teste.exe
Primeira função sem retorno.
Process returned 0 (0x0)   execution time : 0.011 s
Press any key to continue.
```

Tipos de Subrotinas

- Funções: são subrotinas que retornam um valor. Em C/C++ são conhecidas como funções com retorno (e será tratado desta forma nestes slides).

```
1  #include <stdio.h>
2  #include <locale.h>
3
4  int soma (int a, int b) {
5      return (a + b);
6  }
7
8  int main()
9  {
10     setlocale(LC_ALL, "Portuguese");
11
12     int a = 5, b = 6;
13
14     printf("%d + %d = %d", a, b, soma(a,b));
15
16     return 0;
17 }
```

Função com retorno

Chamada da função

```
C:\Users\Usuario\Desktop\Teste\bin\Debug\Teste.exe
5 + 6 = 11
Process returned 0 (0x0)   execution time : 0.009 s
Press any key to continue.
```

Estrutura de uma Função



```
tipoDeRetorno nomeDaFunção (listaDeParâmetros) {  
    corpoDaFunção  
    return (expressão);  
}
```

- Onde:
 - tipoDeRetorno: tipo do valor devolvido pela função, e void caso não haja retorno.
 - nomeDaFunção: nome dado a função.
 - listaDeParâmetros: variáveis recebidas por parâmetros.
 - expressão: valor devolvido pela função.

Estrutura de uma Função

```
1  #include <stdio.h>
2  #include <locale.h>
3
4  void leitura (int *x, char *letra) {
5      do {
6          printf("Valor de %s: ", letra);
7          scanf("%d", x);
8          if (x <= 0) { printf("Valor Inválido! Digite novamente.\n"); }
9      } while (x <= 0);
10 }
11
12 int MDC (int a, int b) {
13     while (a != b) {
14         if (a > b) {
15             a = a - b;
16         } else {
17             b = b - a;
18         }
19     }
20     return (a);
21 }
22
23 int main()
24 {
25     setlocale(LC_ALL, "Portuguese");
26     int a, b;
27     leitura(&a, "A");
28     leitura(&b, "B");
29     printf("MDC (%d,%d) = %d", a, b, MDC(a,b) );
30     return 0;
31 }
```

Tipo de retorno

Lista de Parâmetros

Tipo de retorno

Lista de Parâmetros

Retorno da Função

Chamada de Função

Nome da Função



- **Regras dos nomes de funções:**
 - Nomes significativos.
 - Deve iniciar com uma letra ou `_`(sublinhado), e após pode conter tantas letras, números ou sublinhados quantos deseje o programador.
 - Lembre-se que o C/C++ é sensível a letras maiúsculas/minúsculas, ou seja, se for declarada uma função de nome `Media`, e a chamada estiver `media()`, o compilador não irá reconhecer.

Tipo da Função



- Uma função pode ou não retornar um valor.
- Em funções em que somente se executa uma tarefa, não havendo necessidade de retornar um resultado para ser utilizado por outra função, diz-se que não há retorno da função.

```
void leitura(int *numero){  
    printf("Digite um número inteiro: ");  
    scanf("%d", numero);  
}
```

Tipo da Função



- Caso haja necessidade de retornar um valor , deve-se especificar o tipo do retorno da função. O tipo do retorno depende da variável/valor retornado.

O tipo desta função é int pois a variável retornada (a) é do tipo int.

```
int MDC(int a, int b) {  
    while (a != b) {  
        if (a > b) {  
            a = a - b;  
        } else {  
            b = b - a;  
        }  
    }  
    return (a);  
}
```

- O tipo da função pode ser de um dos tipos básicos do C/C++, um ponteiro a qualquer tipo do C/C++ ou de um tipo struct.

Tipo da Função



- Se o tipo de retorno para uma função é omitido o compilador supõe que o tipo de dado devolvido é int. Mas recomenda-se que mesmo que a função seja int, coloque-se o tipo da mesma, por razões de clareza e consistência.

Retorno de Função



- Uma função pode retornar somente UM valor.
- O valor retornado pode ser qualquer tipo de dado, exceto uma função, um vetor ou uma matriz.
- Não se pode retornar Vetores e Matrizes pois possuem mais que um valor, e uma função retorna somente um valor.
- Uma função pode ter vários return, mas tão logo encontre o primeiro return retorna para a sentença que originou a chamada.

Chamada a uma função

- Para que uma função seja executada, é necessário que seja *chamada* ou *invocada*.

```
1  #include <stdio.h>
2  #include <locale.h>
3
4  void funcao1 () {
5      printf("\nPrimeira Função.\n");
6  }
7
8  void funcao2 () {
9      printf("\nSegunda Função.\n");
10 }
11
12
13 int main()
14 {
15     setlocale(LC_ALL, "Portuguese");
16     printf("Função main(), onde tudo inicia e tudo encerra. \n\n");
17
18     funcao1();
19
20     funcao2();
21
22     return 0;
23 }
```

The diagram illustrates the execution flow of the program. A red line connects the call to `funcao1();` on line 18 of the `main` function to the definition of `funcao1` on line 4. A blue line connects the call to `funcao2();` on line 20 to the definition of `funcao2` on line 8. This visualizes how the program jumps to the function definitions when they are called from within `main`.

Chamada a uma função



- Quando se chama uma função, seja no `main()` ou através de outra função, a função chamada irá executar. Ao seu final, retorna aquela função que a chamou.
- Todo programa inicia no `main()`, e encerra no `main()`.
- Quando tem-se uma função sem retorno (`void`), simplesmente a chamada é feita com o nome da função e o conjunto de parâmetros (caso haja).
 - `leitura(&a);`
- Em uma função com retorno, deve-se considerar que possui um valor, como se fosse uma variável. Então a chamada deve estar vinculada a outra variável, a um `printf`, por exemplo.

Chamada a uma função



- `printf(“%d”, fatorial(num));`
- `if (primo(numero) == 1) { ... }`
- `x = MDC(a,b);`
- `C = fatorial(n) / float(fatorial(k) * fatorial(n-k));`

Chamada a uma função



PRECAUÇÃO

Não se pode definir uma função dentro da outra. Todo código da função deve ser listado sequencialmente durante todo o programa. Antes que apareça o código de uma função, deve surgir a chave de encerramento da função anterior.

Passagem de Parâmetros



- Quando se fala em passagem de parâmetros deve-se, em um primeiro momento, entender o escopo de variáveis.
- Existem variáveis globais, locais e escopo de bloco.
- **Variáveis Globais** são aquelas que são declaradas fora de qualquer função e valem para todo o programa.
- **Variáveis Locais** são as declaradas dentro de alguma função. Quando a função é encerrada a variável é destruída e o espaço retorna para a memória.
- **Variáveis de bloco** são as declaradas dentro de um bloco da função e são válidas somente dentro desse bloco. Ao sair dele são destruídas e o espaço retorna para a memória.
- As variáveis locais tem maior prioridade que as globais.

Passagem de Parâmetros

Variável Global

Variável Local

```
1  #include <stdio.h>
2  #include <locale.h>
3
4  int x = 10;
5
6  int calculo (int a, int b) {
7      int resultado;
8      resultado = a + b;
9      return resultado;
10 }
11
12
13
14 int main()
15 {
16     setlocale(LC_ALL, "Portuguese");
17     int soma = 0, i;
18
19     for (i=1; i<=5; i++) {
20         soma = soma + calculo (i, i+2);
21     }
22
23     printf("Resultado: %d", soma);
24
25     return 0;
26 }
```

Passagem de Parâmetros



- Parâmetros são informações passadas para as subrotinas realizarem operações sobre elas.
- A Passagem de Parâmetros ocorre na chamada da subrotina, mas para isto, sua definição deve estar preparada especificando o tipo e o nome dos parâmetros. O número de parâmetros e a ordem entre eles deve ser a mesma na chamada e na definição da subrotina.

Passagem de Parâmetros



- O nome dos parâmetros corresponde ao nome interno que eles possuem na subrotina e não afetam as variáveis do programa principal, com exceção quando os parâmetros são passados por referência.
- Os parâmetros podem ser passados por valor ou por referência, sendo que:
 - **por valor:** uma cópia do valor é passado para o parâmetro da subrotina.
 - **por referência:** uma referência a posição de memória da variável do programa principal (ponteiro) é passada para o parâmetro

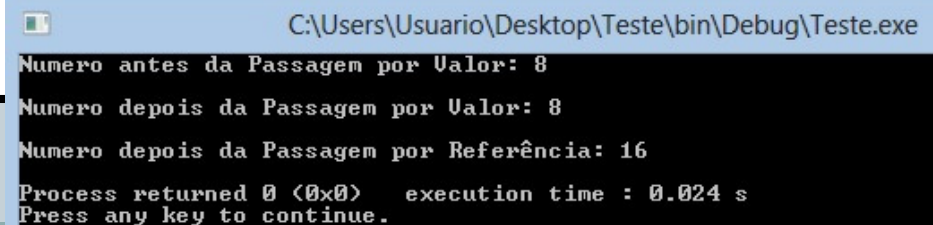
Passagem de Parâmetros



- Quando altera-se o valor de um parâmetro passado por valor, esta alteração não afeta a variável no programa principal.
- Quando altera-se o valor de um parâmetro passado por referência, a variável correspondente no programa principal será alterada. Para passagem de parâmetros por referência coloca-se o símbolo & antes do nome da variável.

Passagem de Parâmetros

```
1  #include <stdio.h>
2  #include <locale.h>
3
4  void parametroValor (int numero) {
5      numero = numero * 2;
6  }
7
8  void parametroReferencia (int *numero) {
9      *numero = *numero * 2;
10 }
11
12 int main()
13 {
14     setlocale(LC_ALL, "Portuguese");
15
16     int numero = 8;
17
18     printf("Número antes da Passagem por Valor: %d\n", numero);
19     parametroValor(numero);
20     printf("\nNúmero depois da Passagem por Valor: %d\n", numero);
21     parametroReferencia(&numero);
22     printf("\nNúmero depois da Passagem por Referência: %d\n", numero);
23
24     return 0;
25 }
```



```
C:\Users\Usuario\Desktop\Teste\bin\Debug\Teste.exe
Numero antes da Passagem por Valor: 8
Numero depois da Passagem por Valor: 8
Numero depois da Passagem por Referência: 16
Process returned 0 (0x0)   execution time : 0.024 s
Press any key to continue.
```

Lembretes



- Todo programa em C/C++ inicia no `main()`, encerra no `main`.
- Se a função tem retorno, a chamada é como se fosse uma variável, vale um valor. Portanto, deve estar ligada a um `printf`, um `if`, um `for`, uma variável, por exemplo.
- Se não possui retorno é simplesmente chamada.
- Passagem de parâmetro e retorno de função são duas situações completamente diferentes. A passagem de parâmetros trata da variável, e o retorno da função faz com que a chamada da função valha um valor.

Lembretes



- A diferença principal entre passagem de parâmetros por valor e por referência (&) é que quando é passada uma variável por valor, a mesma pode ser alterada na função onde é chamada, mas ao final da execução da função não houve modificação do valor. No caso de passagem por referência(&) qualquer alteração realizada na variável na função onde foi chamada, ao finalizar a execução da função, modifica o valor.
- Vetores e Matrizes (arrays) já trabalham com passagem por referência, portanto, não é necessário colocar o &.
- Pode-se retornar somente UM valor, então não é permitido retornar vetores e matrizes.