Índice

Definição

Podemos definir sub-rotinas como sendo blocos de instruções que realizam tarefas específicas. O código de uma sub-rotina é carregado uma vez e pode ser utilizado quantas vezes forem necessárias dentro de seu programa. Dessa maneira, os programas tendem a ficar menores e mais organizados, uma vez que o problema fica organizado em pequenas tarefas, facilitando, por exemplo, a identificação de erros e a manutenção.

Os programas são executados linearmente, uma linha após a outra, do começo ao fim, da esquerda para a direita. Na linguagem C o programa principal começa esta execução pelo main(), independente de onde ele estiver localizado no seu código fonte.

Quando utilizamos sub-rotinas, podemos fazer alguns “desvios” durante a execução natural do programa. Esses desvios são realizados quando uma função ou procedure é chamada pelo programa principal (main).

Observe o exemplo a seguir:

```
#include <stdio.h>
void imprime()
{
   printf("\n\n\nEsta linha foi impressa pela sub-rotina.\n\n");
}
int main()
{
   printf("Aqui iniciamos o programa principal...");
   printf("\n\nApos esta linha estaremos chamando a sub-rotina:");
   imprime();
   printf("Neste ponto voltamos a executar o programa principal\n\n");
   return 0;
}
```

Se você executar este programa perceberá que as linhas da rotina identificada como *main()* serão executadas linearmente até a linha onde encontramos a chamada de uma sub-rotina *imprime()* (que foi escrita no começo do programa, mas que não é executada até esta chamada dentro da sub-rotina *main()*).

O usuário final não consegue ver essa diferença quando ele executa o programa. Seria como se aquela sub-rotina declarada no começo do programa fosse transcrita para o programa principal (main) no momento de sua chamada.

O mais interessante disso é que podemos chamar aquela sub-rotina *imprime()* quantas vezes forem necessárias no decorrer do programa sem ter que reescreve-la, como no exemplo abaixo:

```
#include <stdio.h>

void imprime()
{
   printf("\n\n\nEsta linha foi impressa pela sub-rotina.\n\n");
}

int main()
{
   printf("Aqui iniciamos o programa principal...");
   printf("\n\nApos esta linha estaremos chamando a sub-rotina:");
   imprime();
   printf("Neste ponto voltamos a executar o programa principal\n");
```

```
    printf("Podemos ter diversas linhas de código até que precisemos de executar novamente as mesmas instruções da sub-rotina como aqui:\n");
    imprime();
    printf("E ainda podemos chama-la mais de uma vezseguida:");
    imprime();
    imprime();
    return 0;
}
```

Dentro de uma sub-rotina podemos ter todas as instruções possíveis de ser utilizadas dentro da rotina *main()*, ou seja, podemos declarar variáveis, escrever e ler dados, processando e mostrando seus resultados, até mesmo chamar outras sub-rotinas se necessário.

Veremos agora um exemplo mais prático.

No programa abaixo, o objetivo é calcular a média aritmética das notas de um aluno, onde o aluno possui quatro notas. Inicialmente, mostraremos o programa feito no modo em que escrevemos programas até hoje.

```
#include<stdio.h>
#include<string.h>

int main()
{
  int cont;
  float media,notas[4];
  char nome[40];

  printf("PROGRAMA PARA CALCULAR A MEDIA ARITMETICA DE UM ALUNO\n");
  printf("\nDigite o nome do aluno:");
  gets(nome);
  for(cont=0;cont<4;cont++)
  {
    printf("\nDigite a nota da prova %d: ",cont+1);
    scanf("%f",&notas[cont]);
  }
  media=(notas[0]+notas[1]+notas[2]+notas[3])/4;
  printf("\nA media final do aluno %s e %4.2f.",nome,media);
return 0;
}
```

Agora, separaremos em uma sub-rotina a função que calcula a média do aluno do programa principal. Veja como ficaria o mesmo programa:

```
#include<stdio.h>
#include<string.h>

void calculamedia()
{
  int cont;
  float media,notas[4];
  char nome[40];
  printf("\nDigite o nome do aluno:");
  gets(nome);
```

```
  for(cont=0;cont<4;cont++)
  {
    printf("\nDigite a nota da prova %d: ",cont+1);
    scanf("%f",&notas[cont]);
  }
  media=(notas[0]+notas[1]+notas[2]+notas[3])/4;
  printf("\nA media final do aluno %s e %4.2f.",nome,media);
}

int main()
{
printf("PROGRAMA PARA CALCULAR A MEDIA ARITMETICA DE UM ALUNO\n");
    calculamedia();

  return 0;
}
```

A princípio, podemos verificar que o programa principal praticamente ficou sem função nenhuma. Ele apenas existe apenas para iniciar o programa quando mandamos executar. No momento em que a sub-rotina calculamedia é chamada o programa desvia sua execução para a sub-rotina que é processada.

Mas lembre-se uma variável declarada no programa principal não é “vista” na sub-rotina, por isso levamos todas as declarações das variáveis para dentro da sub-rotina.

Porém, imagine se você fosse criar várias sub-rotinas. Ficaria complicado, num programa mais extenso, a distribuição das variáveis dentro de cada sub-rotina. Para isso, utilizaremos um recurso que chamamos de **passagem de parâmetros**.

Passagem de parâmetros

Definiremos **parâmetros** como sendo as variáveis (ou valores) existentes em uma sub-rotina que serão transmitidas a outra sub-rotina.

Por exemplo, se em nosso exemplo anterior quiséssemos criar uma sub-rotina para realizar o cálculo da média do aluno, separando assim ainda mais o programa inicial, precisaríamos ter o valor das notas nas duas sub-rotinas. Então precisamos informar para a sub-rotina que fará a soma das notas quais serão os valores a serem somados.

Para isso, deveremos informar na declaração da sub-rotina que tipos de valores serão recebidos e por qual nome será reconhecido dentro dela aquele valor recebido. Seria semelhante à declaração de uma variável com seu valor já iniciado, ou seja, uma constante. Assim, na sub-rotina declaramos as variáveis nos parênteses separados por vírgulas. Lembre-se que podemos ter diferentes tipos de dados na subrotina, em sua declaração, e que temos que colocar a chamada em ordem correta para não haver conflito nos tipos de dados. Também devemos passar parâmetros para todas as variáveis declaradas na sub-rotina.

Exemplo:

void somavalores(int a, float b)

No exemplo acima teríamos uma sub-rotina que receberia dois valores inteiros e que, para a sub-rotina, seriam identificados como a (o primeiro) e b (o segundo).

Na sub-rotina que contém a chamada para essa sub-rotina, colocaríamos assim:

somavalores(23,49.03);

Então, esses valores (23 e 49.03) seriam transmitidos para a sub-rotina somavalores e seriam interpretados como a=23 e b=49.03. Esses valores podem ser substituídos por variáveis que possuam o mesmo tipo de dados.

Veja o exemplo completo

```
#include<stdio.h>

void somavalores(int a,  float b)
{
   float soma;
   soma=a+b;
  printf(“\n%d”,a);
  printf(“\n%f”,b);
  printf(“\nA soma dos números é %f”,soma);
}

int main()
{
   int num1;
  float num2;
   printf(“\nDigite o primeiro numero: “);
  scanf(“%d”,&num1);
  printf(“\nDigite o segundo numero: “);
  scanf(“%f”,&num2);
  printf(“\n%d”,num1);
  printf(“\n%f”,num2);
  somavalores(num1,num2);
   return 0;
}
```

Aqui podemos ver claramente qual o valor da leitura dos valores num1 e num2 no programa principal *main()*, a sua passagem para a função somavalores (dentro dos parênteses), como eles foram recebidos (como a e b inteiros) e ainda exibimos o valor (a e b que são os mesmos de num1 e num2) na sub-rotina mostravalores.

No exemplo seguinte, vamos alterar o programa da média de notas do aluno iniciado nas páginas anteriores para utilizarmos a passagem de parâmteros.

```
#include<stdio.h>
#include<string.h>

void somamedia(float n1, float n2, float n3,  float n4, char aluno[40])
{
   float media;
   media=(n1+n2+n3+n4)/4;
   printf("\nA media final do aluno %s e %4.2f.",aluno,media);
}

void calculamedia(char ident[40])
{
   int cont;
   float notas[4];
   for(cont=0;cont<4;cont++)
```

```
    {
      printf("\nDigite a nota da prova %d: ",cont+1);
      scanf("%f",&notas[cont]);
    }
    somamedia(notas[0],notas[1],notas[2],notas[3],ident);
}

int main()
{
    char nome[40];

    printf("PROGRAMA PARA CALCULAR A MEDIA ARITMETICA DE UM ALUNO\n");
    printf("\nDigite o nome do aluno:");
    gets(nome);
    calculamedia(nome);
return 0;
}
```

As sub-rotinas ainda podem ser classificadas de duas formas: como sendo **procedures** ou como **funções**.

Procedures

Procedures, ou procedimentos, são sub-rotinas que utilizamos onde ela não devolve nenhum valor para a sub-rotina que a chamou. Em todos exemplos utilizados até agora realizamos chamadas de sub-rotinas e até mesmo passamos parâmetros para elas, porém quem realizou a chamada da sub-rotina não recebe nenhuma informação devolta. Nesse caso, apelidamos a sub-rotina de **procedure** ou procedimento.

Na linguagem o que caracteriza uma sub-rotina como sendo uma **procedure** é a palavra **void** colocada antes do nome da sub-rotina.

Funções

Funções são sub-rotinas especiais que, diferentemente das procedures, devem retornas algum valor para a sub-rotina que realiza a chamada.

As funções não podem começar com a palavra **void**, e sim devem começar com o tipo de dados que ela enviará no retorno para quem a chamou e ainda devem conter a instrução **return** no final com o valor a ser retornado (que deve possuir o mesmo tipo de dados especificado em sua declaração).

Por exemplo:

```
#include<stdio.h>

int soma(int a, int b);
{
   int resultado;
  resultado = a + b;
```

```
  return resultado;
}
int main()
{
   int n1,n2;
   int resposta;
   printf("\nDigite o valor do 1 número : ");
  scanf("%d",&n1);
   printf("\nDigite o valor do 2 número : ");
  scanf("%d",&n2);
   resposta=soma(n1,n2);
   printf("\nA soma dos números e %d",resposta);
return 0;
}
```

Poderíamos ainda não utilizar a variável resposta, alterando o programa principal no final para

printf("\nA soma dos números e %d",soma(n1,n2));

Vale a pena lembrar que uma mesma função ou procedimento podem ser chamados diversas vezes no decorrer do programa, assim poderíamos ter o programa anterior dessa forma:

```
#include<stdio.h>

int soma(int a, int b)
{
   int resultado;
  resultado = a + b;
  return resultado;
}
int main()
{
   int n1,n2,n3,n4;
   int resposta,diferenca;
   n4=3;
   n3=-9;
   printf("\nDigite o valor do 1 número : ");
  scanf("%d",&n1);
   printf("\nDigite o valor do 2 número : ");
  scanf("%d",&n2);
   resposta=soma(n1,n2);
   printf("\nA soma dos números e %d",resposta);
   printf("\n\ne a soma de 2 e 9 é %d",soma(2,9);
   diferenca=soma(n3,n4);
   printf("\nainda temos a soma de %d e %d como %d",n3,n4,diferenca);
return 0;
}
```

Exercícios:

1) Escreva um programa que leia um número inteiro qualquer e uma função que retorne para o programa principal o valor **1** (um) se o número digitado for positivo e **0** (zero) se o número digitado for negativo.

2) Faça um programa que leia dois números inteiros e escreva uma função que mostre os números inteiros existentes entre eles, retornando o a soma deles para o programa principal e mostrando ao final esse resultado calculado.

3) Faça uma função que receba 3 números informados pelo usuário no programa principal como parâmetros sendo horas, minutos e segundos e retorne e mostre a quantidade de segundos totais para o programa principal. (exemplo: 2 horas 40 minutos e 10 segundos retornariam 9610 segundos).