VOLTAR

Programação Estruturada (Modular)

Apresentar conceitos sobre programação estruturada e aplicar tais conceitos utilizando a linguagem C.

NESTE TÓPICO

- Introdução
- Exemplo 1
- Exemplo 2
- Tipos de Módulos

Introdução

A maioria das linguagens de programação permite a construção de programas estruturados, que consiste na divisão de grandes tarefas de computação em partes menores, a fim de utilizar seus resultados parciais para compor o resultado final desejado. Dessa forma, há uma diminuição da extensão dos programas, de forma que qualquer alteração poderá ser feita mais rapidamente, caso seja necessária. A modularização é uma técnica utilizada para desenvolver algoritmos, na qual se divide o problema em pequenas partes denominadas módulos, sendo estes também conhecidos também como sub-rotinas, sub-programas ou sub-algoritmos.

Ao desenvolvermos um programa, muitas vezes precisamos utilizar uma pequena rotina repetidamente em mais de um local do mesmo, quando fazemos isso, estamos ocupando mais espaço de memória, aumentando a complexidade do algoritmo, aumentando o tamanho do programa. Para evitar tudo isso, podemos modularizar o programa.

A principal vantagem da modularização é possibilitar o reaproveitamento de código, já que podemos utilizar um módulo várias vezes, eliminando assim a necessidade de escrever o mesmo código em situações repetitivas.

Ao trabalhar com essa técnica, pode ser necessário dividir um módulo em outras tantas quantas forem necessárias, buscando uma solução mais simples de uma parte do problema maior. As vantagens da programação estruturada são:

- Deixar o programa mais legível e fácil de entender.
- Encontrar e corrigir erros com mais facilidade.
- Facilitar os testes, diminuindo a probabilidade de erros.
- Reduzir o tempo e custo da programação.
- Possibilitar o reaproveitamento de partes do programa (módulos).

Para exemplificar a técnica de programação estruturada ou modular, abaixo segue os exemplos de um mesmo programa construído de duas maneiras diferentes: sem modularização e com modularização.

Exemplo 1

O programa consiste em uma calculadora básica que realiza as operações de adição, subtração, multiplicação e divisão. Nesse caso, o programa foi criado sem modularização.

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
main ()
{
  int opcao=0;
  float A, B, R;

  while (opcao!=5)
  {
         system("cls");
         printf("\n1-adicao");
         printf("\n2-subtracao");
         printf("\n3-multiplicacao");
         printf("\n4-divisao");
         printf("\n5-sair");
         printf("\n\n-Escolha um num:");
         scanf("%d",&opcao);

         switch(opcao)
        {
             case 1:
             {
                    printf("\nDigite o primeiro numero: ");
                    scanf("%f",&A);
                    printf("\nDigite o segundo numero: ");
                    scanf("%f",&B);
                    R=A + B;
                    printf ("\n\nO resultado e=%0.2f",R);
                    getch();
                    break;
             }
            case 2:
            {
                    printf("\nDigite o primeiro numero: ");
                    scanf("%f",&A);
                    printf("\nDigite o segundo numero: ");
                    scanf("%f",&B);
                    R=A - B;
                    printf ("\n\nO resultado e=%0.2f",R);
                    getch();
                    break;
             }

           case 3:
             {
                    printf("\nDigite o primeiro numero: ");
                    scanf("%f",&A);
                    printf("\nDigite o segundo numero: ");
                    scanf("%f",&B);
                    R=A * B;
                    printf ("\n\nO resultado e=%0.2f",R);
                    getch();
                    break;
             }
           case 4:
           {
                    printf("\nDigite o primeiro numero: ");
                    scanf("%f",&A);
                    printf("\nDigite o segundo numero: ");
                    scanf("%f",&B);
                    R=A / B;
                    printf ("\n\nO resultado e=%0.2f",R);
                    getch();
                    break;
            }
           default:
                    printf("\nEsta operacao nao existe");
         }
    }
    system ("PAUSE");
```

```
72.  }
```

No exemplo acima, percebe-se que existe redundância com relação a trechos de códigos que se repetem várias vezes. Agora, imaginando outra situação em que um programa com cinco mil linhas foi construído de forma não estruturada, ou seja, toda codificação foi feita dentro do programa principal (main) e com vários trechos de códigos que se repetem. Isso não é bom, pois pode demandar tempo e trabalho desnecessário quando da manutenção do programa.

O programa da calculadora apresentado anteriormente permite que seja aplicada a técnica de modularização, pois nos 4 módulos de cálculo existem instruções que realizam as mesmas tarefas. Por exemplo: a entrada e a saída são efetuadas com as mesmas variáveis. A solução é definir as variáveis A, B e R como globais e construir mais dois módulos, uma para entrada e a outra para saída. Os quatro módulos atuais serão diminuídos em número de linhas, pois tudo o que se repete nos módulos será retirado. Observe a declaração das variáveis A, B e R como globais e a definição e chamada de dois novos módulos, entrada e saída. No exemplo abaixo é mostrado, como comentários, os módulos que serão utilizados no programa, sendo que, mais adiante, será explicado como eles são definidos na linguagem C.

Exemplo 2

O programa consiste em uma calculadora básica que realiza as operações de adição, subtração, multiplicação e divisão. Nesse caso, o programa foi criado com modularização.

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>

float A, B, R;

/*********************************************
Definicao do MODULO DE ENTRADA
*/*********************************************

/*********************************************
Definicao do MODULO ADICAO
*/*********************************************

/*********************************************
Definicao do MODULO DE SUBTRACAO
*/*********************************************

/*********************************************
Definicao do MODULO DE MULTIPLICACAO
*/*********************************************

/*********************************************
Definicao do MODULO DE DIVISAO
*/*********************************************

/*********************************************
Definicao do MODULO DE SAIDA
*/*********************************************


main ()
{
     int opcao=0;

     while (opcao!=5)
     {
          system("cls");
          printf("\n1-adicao");
          printf("\n2-subtracao");
          printf("\n3-multiplicacao");
          printf("\n4-divisao");
          printf("\n5-sair");
          printf("\n\n-Escolha um num:");
          scanf("%d",&opcao);

          switch(opcao)
          {
               case 1:
              {
                   /*********************************************
                   Utilizacao do MODULO DE ENTRADA
                  */*********************************************

                   /*********************************************
                   Utilizacao do MODULO ADICAO
                  */*********************************************

                   /*********************************************
                   Utilizacao do MODULO DE SAIDA
                  */*********************************************

                   break;
            }
               case 2:
              {
                   /*********************************************
                   Utilizacao do MODULO DE ENTRADA
                  */*********************************************

                   /*********************************************
```

```
                    Utilizacao do MODULO SUBTRACAO
                   */*********************************************

                    /*********************************************
                    Utilizacao do MODULO DE SAIDA
                   */*********************************************

                       break;
                }
                   case 3:
                  {
                       /*********************************************
                       Utilizacao do MODULO DE ENTRADA
                      */*********************************************

                       /*********************************************
                       Utilizacao do MODULO MULTIPLICACAO
                      */*********************************************

                       /*********************************************
                       Utilizacao do MODULO DE SAIDA
                      */*********************************************

                       break;
                }
                   case 4:
                  {
                       /*********************************************
                       Utilizacao do MODULO DE ENTRADA
                      */*********************************************

                       /*********************************************
                       Utilizacao do MODULO DIVISAO
                      */*********************************************

                       /*********************************************
                       Utilizacao do MODULO DE SAIDA
                      */*********************************************

                       break;
                }
                 default:
                       printf("\nEsta operacao nao existe");
            }
     }
    system ("PAUSE");
  }
```

Tipos de Módulos

Basicamente, há 2 tipos de módulos: procedimentos e funções. Entre esses dois tipos de módulos existem algumas diferenças, mas o conceito é mesmo para ambas.

O importante no uso prático desses dois tipos de módulo é distinguir as diferenças entre eles e como utilizá-los no momento mais adequado.

Procedimentos

Um procedimento é um bloco de programa contendo início e fim, identificado por um nome, por meio do qual será referenciado em qualquer parte do programa principal ou em outro procedimento. A característica principal de um procedimento é que não há retorno de valor ou variável, pois, no momento da execução do programa, somente as instruções definidas dentro do escopo do procedimento são executadas. Quando um procedimento é chamado por um programa, ele é executado até o seu término e a execução do programa volta exatamente para a primeira linha de instrução, após a linha que fez a chamada do procedimento.

A forma geral para se definir um procedimento é a seguinte:

void nome-do-procedimento (lista de parâmetros){

corpo do procedimento (instruções)

}

Exemplo 3

A seguir, apresentamos a definição do Módulo de Entrada citado no exemplo da calculadora. Quando este procedimento for chamado e executado, será solicitada a entrada ao usuário das variáveis A e B.

```
1.  void Entrada ( )
2.  {
3.          printf("\nDigite o primeiro numero: ");
4.          scanf("%f",&A);
5.          printf("\nDigite o segundo numero: ");
6.          scanf("%f",&B);
7.  }
```

Funções

Uma função também é um bloco de programa, como são os procedimentos, contendo início e fim e identificada por um nome, pelo qual também será referenciada em qualquer parte do programa principal. A principal diferença entre um procedimento e uma função está no fato de uma função retornar um determinado valor, sendo que o tipo de dado (int, float, char) de tal valor deve ser definido na criação da função. A forma geral para se definir uma função é a seguinte:

tipo-de-dado nome-da-função (lista de parâmetros){

corpo da função (instruções)

return (variável/valor);

}

Exemplo 4

A seguir, apresentamos a definição do Módulo Adição citado no exemplo da calculadora. Quando esta função for chamada e executada, será feito o cálculo de adição das variáveis A e B, o qual será atribuído à variável R, que será o retorno da função.

```
1.  float  Adicao ( )
2.  {
3.          R = A + B;
4.          return (R);
5.  }
```

Utilização dos Procedimentos e Funções

A definição de procedimentos e funções só faz sentido se eles forem utilizados em alguma parte do programa. Os procedimentos e funções, em geral, são utilizados dentro do programa principal. Entretanto, eles também podem ser chamados em outros módulos.

A forma geral da utilização de um procedimento é a seguinte:

nome-do-procedimento (lista de parâmetros);

Para utilizarmos uma função, normalmente, devemos declarar uma variável que receberá o retorno da função. Então, a forma geral da utilização de uma função é a seguinte:

Variável = nome-da-função (lista de parâmetros);

Exemplo 5

Para exemplificarmos a utilização dos procedimentos e funções, abaixo segue o programa da calculadora escrito de forma estruturada ou modular. Observe que a entrada e saída de dados foram transformadas em procedimentos e as operações matemáticas em funções.

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>

float A, B, R;

void Entrada ( )
{
        printf("\nDigite o primeiro numero: ");
        scanf("%f",&A);
        printf("\nDigite o segundo numero: ");
        scanf("%f",&B);
}

float  Adicao ( )
{
        R = A + B;
        return (R);
}
float  Subtracao ( )
{
        R = A - B;
        return (R);
}
float  Multiplicacao ( )
{
        R = A * B;
        return (R);
}
float  Divisao ( )
{
        R = A / B;
        return (R);
}

main ()
{
  int opcao=0;
  float result;

  while (opcao!=5)
  {
  system("cls");
  printf("\n1-adicao");
  printf("\n2-subtracao");
  printf("\n3-multiplicacao");
  printf("\n4-divisao");
  printf("\n5-sair");
  printf("\n\n-Escolha um num:");
  scanf("%d",&opcao);

  switch(opcao)
   {
     case 1:
     {
          Entrada ();
          result = Adicao ();
          printf ("\n A + B = %f", result);
          break;
     }
case 2:
     {
          Entrada ();
          result = Subtracao ();
          printf ("\n A - B = %f", result);
          break;
     }
case 3:
     {
          Entrada ();
          result = Multiplicacao ();
```

```
72.            printf ("\n A * B = %f", result);
73.            break;
74.       }
75.        case 4:
76.       {
77.            Entrada ();
78.            result = Divisao ();
79.            printf ("\n A / B = %f", result);
80.            break;
81.       }
82.
83.       default:printf("\nEsta operacao nao existe");
84.     }
85.     }
86.  system("PAUSE");
87.  }
```

Existe, também, outra forma de estruturarmos um programa por meio de módulos, os quais podem ser definidos após o programa principal (main). Neste caso, como os módulos são criados após o programa principal, então deverá haver um protótipo para cada módulo logo após a seção include. O protótipo consiste no cabeçalho do módulo.

Exemplo 6

Para ilustrarmos melhor esta outra forma para estruturarmos o programa, abaixo segue o mesmo exemplo da Calculadora, no qual os módulos são criados após o programa principal.

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>

float A, B, R;

/*****************************
Prototipos dos modulos
*****************************/
void Entrada ();
float Adicao ();
float Subtracao ();
float Multiplicacao ();
float Divisao ();

/*****************************/

main ()
{
  int opcao=0;
  float result;

  while (opcao!=5)
  {
       system("cls");
       printf("\n1-adicao");
       printf("\n2-subtracao");
       printf("\n3-multiplicacao");
       printf("\n4-divisao");
       printf("\n5-sair");
       printf("\n\n-Escolha um num:");
       scanf("%d",&opcao);

       switch(opcao)
       {
           case 1:
           {
                Entrada ();
                result = Adicao ();
                printf ("\n A + B = %f", result);
                break;
            }
           case 2:
           {
                Entrada ();
                result = Subtracao ();
                printf ("\n A - B = %f", result);
                break;
            }
           case 3:
           {
                Entrada ();
                result = Multiplicacao ();
                printf ("\n A * B = %f", result);
                break;
            }
           case 4:
           {
                Entrada ();
                result = Divisao ();
                printf ("\n A / B = %f", result);
                break;
            }

           default:
                printf("\nEsta operacao nao existe");
        }
    }
}
void Entrada ( )
{
```

```
72.         printf("\nDigite o primeiro numero: ");
73.         scanf("%f",&A);
74.         printf("\nDigite o segundo numero: ");
75.         scanf("%f",&B);
76.  }
77.
78.  float  Adicao ( )
79.  {
80.         R = A + B;
81.         return (R);
82.  }
83.  float  Subtracao ( )
84.  {
85.         R = A - B;
86.         return (R);
87.  }
88.  float  Multiplicacao ( )
89.  {
90.         R = A * B;
91.         return (R);
92.  }
93.  float  Divisao ( )
94.  {
95.         R = A / B;
96.         return (R);
97.  }
```

Exemplo 7

Neste exemplo, o programa contém os seguintes módulos:

1. Um *procedimento* para a leitura da variável n.
2. Um *procedimento* que calcula e mostra o dobro do valor de n.
3. Uma *função* que verifica se o número n é par ou ímpar. Se n é par, então a função retorna 1. Se n é ímpar, a função retona 0 (zero).

Observação: os módulos são chamados no programa principal (main) e a variável n é declarada como global.

```
#include <stdio.h>
#include <stdlib.h>
int n;

void entrada (){
    printf ("\n Digite o valor de n=");
    scanf ("%d", &n);
}

void dobro (){
    n = n*2;
    printf ("\n Dobro de n=%d",n);
}
int paridade (){
     if (n%2==0)
         return (1);
     else return (0);Prática de Programação
}
main () {
        int par_impar;

        entrada ();
        dobro();
        par_impar = paridade ();

        if (par_impar==1)
             printf ("\n O numero e par");
        else printf ("\n O numero e impar");

        system ("pause");
}
```

Agora que você já estudou essa aula acesse a plataforma AVA, resolva os exercícios e verifique o seu conhecimento. Caso fique alguma dúvida, leve a questão ao Fórum e divida com seus colegas e professor.

Quiz

Exercício

Programação Estruturada (Modular)

INICIAR

Quiz

Exercício Final

Programação Estruturada (Modular)

INICIAR

Referências

MIZRAHI, V. V. *Treinamento em linguagem C.* São Paulo: Pearson, 2008.

SCHILDT, H. C – *Completo e Total.* São Paulo: Pearson, 2006.

Avalie este tópico

ANTERIOR
Alocação dinâmica de memória

Índice

PRÓXIMO
Passagem de parâmetros

Ajuda?
(https://ava.uninove.br/seu/AVA/topico/topico.php?idCurso=)