Linguagem e Técnicas de Programação

Prof. Msc: Marlus Dias Silva
(64)98133-7874

Sumário

- Programação Modular
- Subprogramação
- Subprograma
- Parâmetros
- Procedimentos
- Funções
- Exercício

Programação Modular

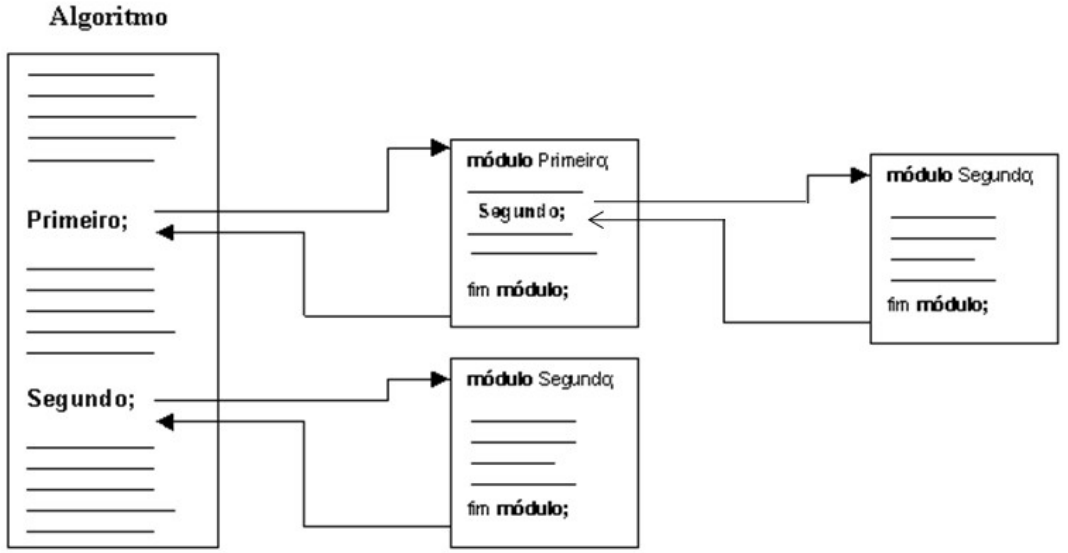
- A arte de programar consiste na arte de organizar e dominar a complexidade dos sistemas (Dijkstra, 1972).
- Um aspecto da programação estruturada é a decomposição de um algoritmo em módulos, usando a técnica denominada **“programação modular”**(Staa, 2000).
- O objetivo da programação modular é diminuir a complexidade dos programas, usando a estratégia de **“dividir para conquistar” (dividir problemas complexos em problemas menores).**
- Usando essa estratégia, um algoritmo é dividido em partes menores, chamadas de **módulos.** Cada módulo tem um único ponto de entrada, e sua execução termina em um único ponto de saída, contendo no seu fluxo de execução: sequências de ações, seleções e iterações. Cada módulo é implementado por um subalgoritmo (subprogramas) específico, passível de ser construído e testado de forma independente. Os diferentes subprogramas são posteriormente integrados em um só programa.
- A modularização tem o objetivo de facilitar a compreensão, o desenvolvimento, o teste e a manutenção dos sistemas. Programas que utilizam subprogramação resultam mais confiáveis e flexíveis.

subprogramação

- Um **subprograma** (as vezes chamado de sub-rotina) consiste de um trecho de código com estrutura semelhante à de um programa, que é executado somente quando acionado por outro trecho de código. Esse acionamento costuma-se denominar chamada ao subprograma.
- Um subprograma deve executar um única tarefa, claramente definida. Um programa, ao utilizar um subprograma para executar uma tarefa, não deve-se preocupar em como essa tarefa será executada. A execução correta de um subprograma deve ser assegurada sempre que esse seja adequadamente chamado.
- A utilização de subprograma é uma técnica de programação que visa:
  - A definição de trechos de **código menores**, mais fáceis de serem construídos e testados;
  - A **diminuição do tamanho dos programas**, pela eliminação de **redundância**, ao evitar que códigos semelhantes sejam repetidos dentro de um programa;
  - A **construção mais segura de programas complexos**, pela utilização de unidades menores (os subprogramas) já construídas e testadas;
  - A **reutilização de código** em um programa ou em programas diferentes.

Subprograma

- Todo subprograma é identificado através de um nome. Esse nome deve representar claramente a tarefa a ser executada pelo subprograma.
- A chamada a um subprograma é feita pelo seu nome, por um comando especídico ou utilizando diretamente seu resultado. Um subprograma pode ser chamado e executado diversas vezes, em diferentes pontos de um programa.
  - Adicionalmente, um subprograma também pode conter chamadas a outros subprogramas.



Parâmetros

- Os valores que um subprograma necessita receber para poder realizar sua tarefa, ou os valores que produz e que devem ser visíveis externamente após concluída sua execução, devem ser sempre armazenados em parâmetros. Parâmetros são espaços de armazenamento que permitem a comunicação do subprograma com o mundo externo.

- Os parâmetros que aparecem na declaração dos subprogramas são chamados **parâmetros formais** porque, durante a execução, na chamada dos subprogramas, são substituídos por varáveis ou valores do mesmo tipo, muitas vezes com nomes totalmente diferentes.

```
Subprograma Fatorial
    Parâmetro : número (inteiro)
```

- O parâmetro formal número não provoca a reserva de espaço de memória. Ele simplesmente indica que, ao ser chamado o subprograma Fatorial, deve ser fornecido um número inteiro para a sua execução.

Parâmetros

- Os parâmetros utilizados na chamada de um subprograma, chamados de **parâmetros reais**, substituem os formais durante sua execução. Os parâmetros reais devem sempre concordar em quantidade e tipo com os respectivos parâmetros formais, na ordem em que esses foram definidos. Podem ser fornecidos como **parâmetros reais** nomes de variáveis, valores literais ou resultados de expressões. As variáveis utilizadas como parâmetros reais devem ter sido declaradas no programa que chama o subprograma.

```c
#include <stdio.h>
int soma(int,int);
int main(){
    int x,y;
    printf("Informe um valor \n");
    scanf("%d",&x);
    printf("Informe um valor \n");
    scanf("%d",&y);

    int resultado = soma (x,y);
    printf("O valor da Soma é %d\n",resultado);
    return 0;
}
int soma(int a,int b){
    return (a+b);
}
```

Procedimento

- Um **procedimento** é um subprograma que executa uma determinada tarefa com ou sem a utilização de parâmetros e não retorna nada para o programa principal.
  - Na linguagem de programação C a palavra reservada **void** indica que um procedimento não retorna nada.

```c
#include <stdio.h>
void linguagem(int);
int main(){
    linguagem(10);
    return 0;
}
void linguagem(int dia){
    printf("Essa semana a aula de linguagem será no dia %d \n",dia);
}
```

Função

- Uma função é um subprograma que devolve um valor, resultante de um cálculo ou da execução de uma determinada tarefa, ao programa que o chamou por meio de seu nome.

```c
#include <stdio.h>
int soma(int,int);
int main(){
    int x,y;
    printf("Informe um valor \n");
    scanf("%d",&x);
    printf("Informe um valor \n");
    scanf("%d",&y);

    int resultado = soma (x,y);
    printf("O valor da Soma é %d\n",resultado);
    return 0;
}
int soma(int a,int b){
    return (a+b);
}
```

Exercício

1) Construa um subprograma que, recebendo como parâmetro quatro números inteiros, devolva ao módulo que o chamou a soma dos três maiores números dentre os quatro recebidos.