Programação em computadores II

Modularização

Prof. Dr. Fábio Rodrigues de la Rocha - utilizando o material do prof. Alexandre Gonçalves

Definição

- Modularização: consiste na subdivisão de um programa em vários módulos (*subrotinas*) específicos visando uma maior clareza e otimização do programa resultante;
- Também chamadas de *subprogramas*, são blocos de instruções que realizam tarefas específicas;
- O código de uma subrotina é carregado uma vez e pode ser executado múltiplas vezes;

Definição

- Como o problema pode ser subdividido em pequenas tarefas, os programas tendem a ficar menores e mais organizados;
- Os programas em geral são executados linearmente, uma linha após a outra, até serem completados;
- Contudo, através de subrotinas, é possível a realização de desvios na execução dos programas;
- Estes desvios são efetuados quando uma função é chamada pelo programa principal

Definição

- A linguagem C possibilita a modularização por meio de funções;
- Um programa escrito em C tem, no mínimo, uma função chamada **main**, por onde a execução começa;
- Existem também muitas outras funções predefinidas na linguagem C, por exemplo, **clrscr()**, **printf()**, **scanf()**. Estas funções são adicionadas aos programas pela diretiva **#include**, no momento da "linkedição";

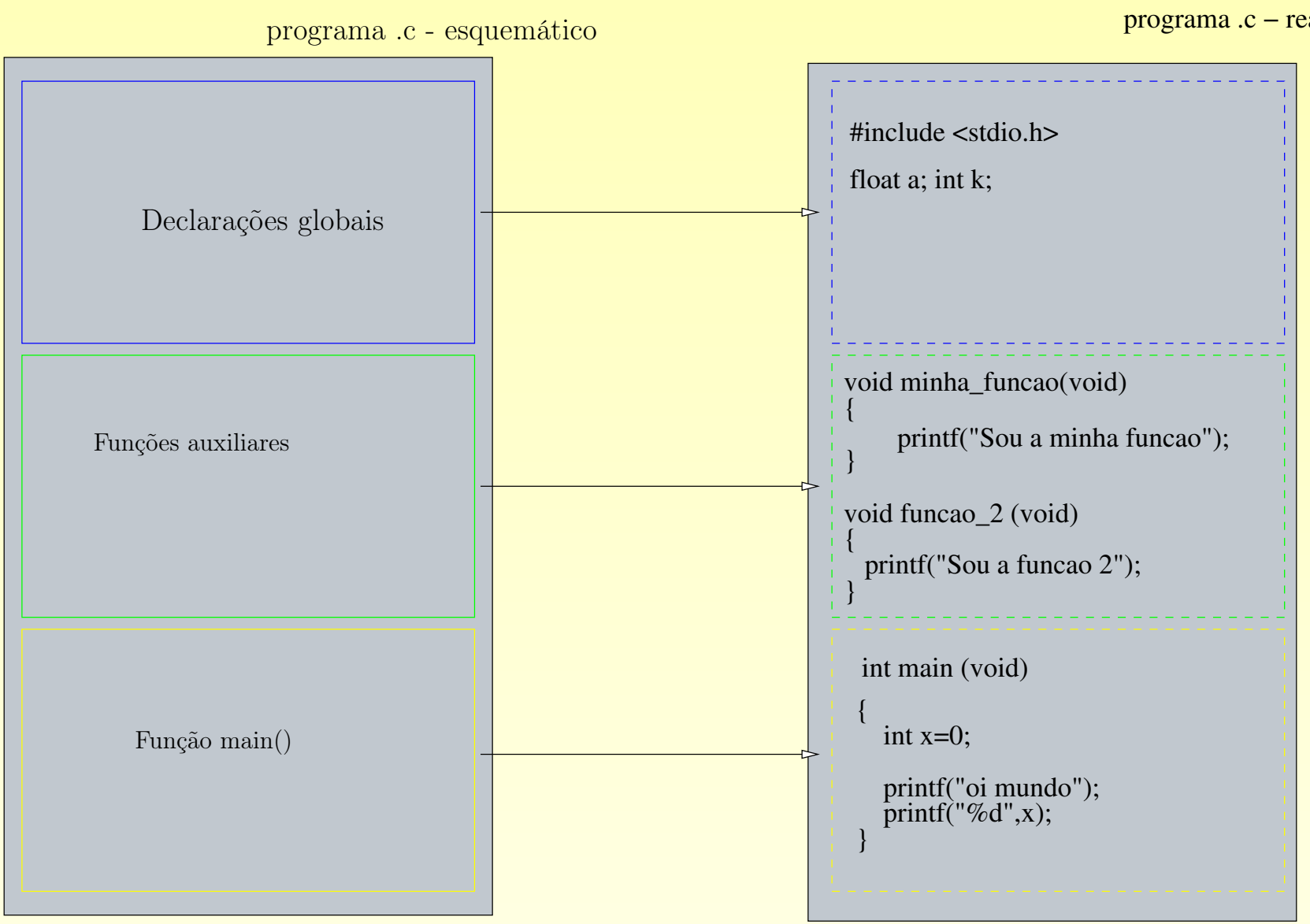
Definição

- Além das funções predefinidas, várias outras podem ser criadas conforme a necessidade;
- As funções às vezes precisam receber valores externos, chamados parâmetros, e também podem devolver algum valor produzido para o ambiente externo, denominado retorno;
- Os parâmetros são representados por uma lista de variáveis colocadas dentro de parênteses, logo após o nome da função;
- Caso haja retorno deve-se incluir o comando **return**;

Definição

- O tipo do valor retornado deverá ser exatamente igual ao tipo informado antes do nome da função;
- Caso não haja retorno, deverá ser digitada a palavra **void**;
- Na linguagem C todos os módulos de um programa são denominados funções. A função que corresponde ao programa principal chama-se **main()**;

Subprogramas em C



Subprogramas em C

- A Linguagem C não admite aninhamento de subprogramas, ou seja, subprograma dentro de subprograma;
- A forma geral de uma função em C é definida abaixo:

```
tipo_de_retorno NOME_SUBROTINA (lista de parametros)
{
     comando1;
     comando2;
     .
     .
}
```

Subprogramas em C

- Uma função não precisa ter parâmetros, neste caso usa-se na lista de parâmetros **(void)** ou ainda **()**;
- uma função pode ter ou não retorno. Caso não tenha retorno, escreve-se **void** antes do nome da subrotina;

Subprogramas em C

- O retorno de uma função é um valor que será considerado o resultado da função. Como exemplo pense numa função chamada seno que tenha sido utilizada da seguinte forma:

```
int main (void)
{
    float a;
    a = seno(45); // o retorno será 0.7071
    return 0;
}
```

- a subrotina seno calculará o seno de 45 (0.7071) e o seu valor numérico será o resultado/retorno da função. Este valor será copiado para a variável **a**, de forma semalhante ao que ocorre se escrevermos:

```
int main (void)
{
    float a;
    a = 0.7071;
```

Retorno de ua função

- Existem duas formas de retorno de uma função ao módulo que a chamou: retorno natural e retorno explícito;
- No retorno natural o fluxo retorna logo após a execução do último comando do corpo da função;
- No retorno explícito, o retorno ocorre através da execução do comando return;
- return;
- return a;
- return ++a;
- return a+b; return (a+b);

Retorno de ua função

- No exemplo seguinte, uma função retorna **int** e tem dois parâmetros, cada qual também do tipo **int**;
- o comando **return** faz com que a função termine e o seu retorno/resposta é o valor contido neste;

```
int soma (int x, int y)
{
   return (a+y);
}
int main (void)
{
    int a;
    a =  soma(10,20)
    return 0;
 }
```

Retorno de ua função

- No exemplo seguinte, uma função retorna **int** e tem dois parâmetros, cada qual também do tipo **int**;
- o comando **return** faz com que a função termine e o seu retorno/resposta é o valor contido neste;

```
void mostra_mensagem (char s[], int idade)
{
   printf("O nome é: %s e a idade é: %d\n);
}
int main (void)
{
    int a;
    mostra_mensagem ("Ana", 35);
    return 0;
 }
```

Escopo de variáveis

- **Variáveis globais**: declaradas fora dos subprogramas. Recebem esse nome porque podem ser utilizadas em qualquer ponto do programa, incluindo dentro dos subprogramas;
- **Variáveis locais**: declaradas dentro de um subprograma e existirá enquanto este estiver sendo executada;

```c
#include <stdio.h>
int global=50;
void minha_funcao_0 (void)
{
    int x=20;
    printf(" global %d e a local %d\n",global,x);
}
void minha_funcao_1 (void)
{
    int x=200;
    printf(" global %d e a local %d \n",global,x);
}
int main (void)
{
  global = 100;
  minha_funcao_0();
  global = 1000;
  minha_funcao_1();
  return 0;
}
```

Funções sem passagem de parâmetros e sem retorno

```c
#include <stdio.h>
void soma(void) {
  int a, b, s;
  printf("Digite o primeiro nmero: ");
  scanf("%d",&a);
  printf("\nDigite o segundo nmero: ");
  scanf("%d",&b);
  s = a + b;
  printf("\nResultado: %d",s);
}

int main (void)
{
  soma();
  getchar();
  return 0;
}
```

Funções com passagem de parâmetros e sem retorno

```c
#include <stdio.h>
void media(float a, float b) {
  float s;
  s = (a + b) / 2;
  printf("\nResultado: %.2f", s);
}

int main (void)
{
  float a, b;
  printf("Digite o primeiro nmero: ");
  scanf("%f",&a);
  printf("\nDigite o segundo nmero: ");
  scanf("%f",&b);
  media(a, b);
  getchar();
  return 0;
}
```

Funções sem passagem de parâmetros e com retorno

```c
#include <stdio.h>
float multiplicacao() {
  float multiplicando, multiplicador, produto;
  printf("Digite o multiplicando: ");
  scanf("%f",&multiplicando);
  printf("\nDigite o multiplicador: ");
  scanf("%f",&multiplicador);
  produto = multiplicando * multiplicador;
  return produto;
}
int main (void)
{
  float resposta;
  resposta = multiplicacao();
  printf("\nResultado: %.2f", resposta);
  getchar();
  return 0;
}
```

Funções com passagem de parâmetros e com retorno

```c
#include <stdio.h>
float divisao(float dividendo, float divisor) {
  float q;
  q = dividendo / divisor;
  return q;
}
int main (void)
{
  float n1, n2, resposta;
  printf("Digite o dividendo: ");
  scanf("%f",&n1);
  printf("\nDigite o divisor: ");
  scanf("%f",&n2);
  resposta = divisao(n1, n2);
  printf("\nResultado: %.2f", resposta);
  getchar();
  return 0;
}
```

Modos de passagem de parâmetros

- Por valor; - forma que foi utilizada até este ponto
- Por referência;

Passagem por valor

```
1  #include <stdio.h>
2  int dobro(int a, int b) {
3    int soma;
4    a = a * 2;
5    b = b * 2;
6    soma = a + b;
7    return soma;
8  }
9  int main (void) {
10   int x, y, res;
11   printf("Digite o 1. nmero: ");
12   scanf("%d",&x);
13   printf("\nDigite o 2. nmero: ");
14   scanf("%d",&y);
15   res = dobro(x, y);
16   printf("\nResultado: %d", res);
17   getchar();
18   return 0;
19 }
```

Passagem por valor

- Passagem de parâmetro por referência significa que os parâmetros passados para a função correspondem a endereços de memória ocupados por variáveis;
- Dessa maneira, toda vez que for necessário acessar determinado valor, isso será feito por meio de referência, ou seja, apontamento ao seu endereço;

Passagem por referência

```
1  #include <stdio.h>
2  int dobro(int a, int b) {
3    int soma;
4    a = a * 2;
5    b = b * 2;
6    soma = a + b;
7    return soma;
8  }
9  int main (void) {
10   int x, y, res;
11   printf("Digite o 1. nmero: ");
12   scanf("%d",&x);
13   printf("\nDigite o 2. nmero: ");
14   scanf("%d",&y);
15   res = dobro(x, y);
16   printf("\nResultado: %d", res);
17   getchar();
18   return 0;
19 }
```

Passagem por referência

- A linguagem C não permite que vetores e matrizes sejam passados na íntegra como parâmetros para uma função;
- Portanto, deve-se passar apenas o endereço da posição inicial do vetor ou da matriz;
- O endereço é obtido utilizando-se o nome do vetor (ou da matriz) sem o índice entre colchetes;

Passagem por referência

```
1  #include <stdio.h>
2  void soma_linhas(float m[][5], float v[]) {
3    for (int i=0; i<3; i++) {
4      for (int j=0; j<5; j++) {
5        v[i] += m[i][j];
6      }
7    }
8  }
9  int main (void)
10 {
11   float m[3][5], v[3];
12   for (int i=0; i<3; i++) {
13     v[i] = 0;
14     for (int j=0; j<5; j++){
15       printf("Informe um valor para a %d. linha e %d. coluna: ",i+1,j+1);
16       scanf("%f",&m[i][j]);
17     }
18   }
19   soma_linhas(m, v);
20   for (int i=0; i<3; i++) {
21     printf("\nSoma da linha %d: %.2f",i+1, v[i]);
22   }
23   getchar();
24   return 0;
25 }
```

Exercícios:

1) Faça um programa contendo uma sub-rotina que receba um valor inteiro e retorne um inteiro indicando se é par ou ímpar;

2) Crie uma função que receba três números inteiros, **a**, **b** e **c**, sendo **a** maior que 1 e **c** maior que **b**. A função deverá somar todos os inteiros entre **b** e **c** que sejam divisíveis por **a**, ou seja, módulo igual a zero, e retornar o resultado para a função principal.

3) Elabore um programa contendo uma sub-rotina que receba as três notas de um aluno e uma letra como parâmetros. Se a letra for A, a sub-rotina deverá calcular a média aritmética das notas do aluno; se for P, deverá calcular a média ponderada, com pesos 5, 3, e 2. A média calculada deverá ser devolvida ao programa principal para, então, ser apresentada.

Exercícios:

4) Elabore um programa que receba e armazene as notas da 1a prova, 2a prova e trabalho final de 15 alunos. Note que cada linha da estrutura representa um aluno, enquanto que as colunas representam as três notas. Para tal, utilizando um procedimento calcule e apresente:

- A média das 3 notas para cada aluno;
- A média de cada uma das notas (1a, 2a e 3a) para todos os alunos.

5) Elabore um programa que receba um vetor de n posições e através de uma subrotina faça a ordenação apresentando o resultado no programa principal;

Exercícios:

6) Faça um programa contendo uma sub-rotina que receba dois valores numéricos e um símbolo. Este símbolo representará a operação que se deseja efetuar com os números. Se o símbolo for + deverá ser realizada uma adição, - subtração, * multiplicação e / divisão. O resultado deverá ser apresentado no programa principal;

7) Foi realizada uma pesquisa entre quinze habitantes de uma região. As seguintes informações foram coletadas: idade, sexo, salário, número de filhos. Faça uma subrotina que leia esses dados armazenando-os em vetores. Depois, crie subrotinas que calculem a média salarial dos habitantes, a menor e a maior idade do grupo, a média de filhos e a média salarial por sexo. Utilize uma subrotina para cada cálculo;

Exercícios:

8) Crie um programa que:
Receba o preço de 4 produtos e armazene-os em um vetor; Receba a quantidade estocada de cada um desses produtos em cinco armazéns diferentes, utilizando uma matriz 5x4. O programa deverá calcular através de sub-rotinas e apresentar no corpo principal as seguintes informações:

- A quantidade de produtos estocados em cada um dos armazéns;
- A quantidade de cada um dos produtos estocados em todos os armazéns juntos;
- O preço do produto que possui maior estoque em todos os armazéns;
- O armazém que possui o menor estoque; O valor (preço x qtde) de cada armazém;

Exercícios:

9) Um banco possui a necessidade de um programa que calcule o provável rendimento de determinada aplicação financeira. Para tal, deve-se levar em consideração a tabela de rendimentos abaixo, bem como as informações de imposto e taxa de administração (essas duas informações são aplicadas sobre o rendimento mensal). Nesse sentido, o programa deve requisitar o montante desejado para aplicação, o tipo da aplicação e a quantidade de meses previstos. Considere ainda que aplicações com prazos superiores a 6 meses não terão imposto de renda. Ao final, a função de cálculo de rendimento deve retornar o resultado para a apresentação.

Exercícios:

Código	Valor mínimo	rendimento	Taxa adm.	Imposto
1	1000,00	0,8% a.m.	2 % a.m	10% a.m.
2	5000,00	0,9% a.m.	1,5% a.m	12% a.m.
3	20000,00	0,95% a.m.	1 % a.m.	15% a.m.

Exercícios:

10) Elabore um programa que crie duas matrizes em memória e sub-rotinas para carregar as matrizes, preencher as matrizes com conteúdo informado pelo usuário, verificar a compatibilidade e multiplicar as matrizes.

Na função main() deve apenas possuir a definição das matrizes assim como as chamadas das sub-rotinas.

Utilize para a definição das matrizes dimensões máximas, por exemplo, 20 x 20, assim como uma matriz de 2x2 para guardar as dimensões das duas matrizes. Desse modo, apesar dos parâmetros serem fixos a matriz de dimensão permite saber qual a real dimensão de cada matriz.

Exercícios:

Para resolver o exercício pense na estruturação/modularização do programa sem se preocupar com o desenvolvimento do código de cada função, ou seja, as funções e variáveis devem ser declaradas mas o código não deve ser escrito.

A função principal main() deve sim conter as variáveis e as chamadas das funções previamente especificadas. Considere o conceito de fluxo de execução de funções caso haja a necessidade de chamar uma função a partir de outra. Testes e verificações antes da chamada de uma determinada função devem ser especificados.