# Convertendo Algoritmos para a Linguagem C

## Notas de Aula

Prof. Francisco Rapchan
www.geocities.com/chicorapchan
rapchan@terra.com.br

*O objetivo deste texto é mostrar alguns programas em C, dando uma breve descrição de seu funcionamento para que o leitor possa se "acostumar" com a forma dos programas nessa linguagem. A explicação sobre o funcionamento e uso de cada recurso da linguagem será visto ao longo do curso.*

*O C é uma linguagem de alto nível de uso genérico desenvolvida por programadores para ser usada por programadores. Em outras palavras: não é uma linguagem para uso didático e sim para trabalhar.*

*Ao longo do tempo, grandes sistemas têm sido desenvolvidos em C como, por exemplo, o UNIX, o Windows e o Linux. Essa linguagem foi também usada como base para desenvolver novas linguagens, entre elas a linguagem C++, Java.*

*A primeira versão de C foi criada por **Dennis Ritchie** em 1972 nos laboratórios Bell para ser incluído como um dos softwares a serem distribuídos juntamente com o sistema operacional Unix do computador PDP-11, na equipe coordenada por **Ken Thompson**.*

*Alguns marcos históricos:*

- *1969 – Desenvolvimento do UNIX (em um computador PDP 7 em linguagem Assembly).*
- *1969 – Desenvolvimento da linguagem BCPL (muito próxima do Assembly).*
- *1970 – Denis Ritchie cria a linguagem B a partir do BCPL nos laboratórios da Bell Telephones.*
- *1971 – Primeiro desenvolvimento da linguagem C, sucessora do B (o C é a 2ª letra de BCPL).*
- *1973 – O sistema operacional UNIX é reescrito em linguagem C.*
- *1978 – Primeira edição do livro "The C Programming Language", Kernighan & Ritchie.*
- *1980 – A linguagem C é padronizada pelo American National Standard Institute: surge o ANSI C.*
- *1992 – Surge o C++, uma evolução da linguagem C incorporando conceitos da orientação a objetos.*

**1.** Faça um programa que leia dois números que o usuário do computador digitará no teclado, some-os e mostre o resultado na tela do computador.

| **Portugol** | **C** |
|---|---|
| **algoritmo** "Soma de dois números"<br><br>**var**<br>  A, B, Soma: **inteiro**<br><br>**inicio**<br>  **Leia** (A)<br>  **Leia** (B)<br>  Soma <- A + B<br>  **Escreva** (Soma)<br><br>**Fimalgoritmo** | #include <stdio.h><br><br>**int main**()<br>**{**<br>  **int** A, B, Soma;<br>  **scanf** ("%d",&A);<br>  **scanf** ("%d",&B);<br>  Soma **=** A + B;<br>  **printf** ("Soma: %d \n", Soma);<br><br>  **return** (0)<br>**}** |

Algumas observações.

- Nas linguagens reais como Pascal e C são necessárias algumas **bibliotecas** para que o código executável possa ser executado corretamente. No caso do C precisaremos da biblioteca **stdio.h** que contém informações sobre as funções de leitura do teclado (scanf) e de escrita no vídeo (printf). Para incluir estas bibliotecas, em usamos **#include**.
- Observe que ao final de cada linha do programa C há um ponto-e-vírgula (;).
- Em C, main( ) representa o programa principal que inicia com { e termina com }. Na verdade main ( ) é uma função que retorna um valor inteiro.
- É importante notar que em C letra maiúscula e letra minúscula são diferentes. Assim, uma variável chamada **Soma** é diferente de outra chamada de **soma**.
- O sinal de atribuição em C é o igual **=**. Para indicar o igual em uma decisão usamos **==**.
- A função **printf** é usada em C para escrever dados na tela.
- Em C, o parêntese **{** equivale ao início de um bloco de comandos e **}** equivale ao fim de um bloco de comandos.
- A função **main** do C é a primeira a ser executada pelo programa. Equivale ao programa principal em outras linguagens. Todo programa deve ter uma única função `main` e ela é sempre do tipo **int**. Ao final da função main devemos ter um return (0) que indica ao sistema operacional que o programa terminou bem.
- Tipos em C

| **Tipo** | **Entrada e Saída** | **Descrição** |
|---|---|---|
| int | %i ou %d | inteiro decimal |
| float | %f | Real (ponto flutuante) |
| char | %c | caracter simples |
| | %s | string |

**2.** Faça um programa que leia dois números e mostre se são iguais. Se não forem iguais, mostre o maior.

| **Portugol** | **C** |
|---|---|
| `ALGORITMO "Mostra o maior"`<br><br>`Var`<br>`  numero_um, numero_dois: real`<br>`inicio`<br>`  Leia (numero_um)`<br>`  Leia (numero_dois)`<br><br>`  se numero_um = numero_dois entao`<br>`  escreva ("São iguais")`<br>`  senao`<br>`  Se numero_um > numero_dois entao`<br>`  escreva (numero_um)`<br>`  senao`<br>`  escreva (numero_dois)`<br>`  fimse`<br>`  fimse`<br><br>`fimalgoritmo` | `#include <stdio.h>`<br><br>`int main()`<br>`{`<br>`  float numero_um, numero_dois;`<br><br>`  scanf ("%f",&numero_um);`<br>`  scanf ("%f",&numero_dois);`<br><br>`  if (numero_um == numero_dois)`<br>`  {`<br>`  printf ("Sao iguais");`<br>`  }`<br>`  else`<br>`  {`<br>`  if (numero_um > numero_dois){`<br>`  printf ("O maior: %f", numero_um);`<br>`  }`<br>`  else{`<br>`  printf ("O maior: %f", numero_dois);`<br>`  }`<br>`  }`<br>`  return (0);`<br>`}` |

A instrução if assume em C as duas formas básicas:

```
if (expressão)
   instrução;
```

ou:

```
if (expressão)
   instrução_1;
else
   instrução_2;
```

Em C, os comandos internos à cada estrutura deve começar e terminar com chaves. Assim, para o comando IF:

```
if (<expressão-booleana>)
{
     comando1;
     ...
     comandoN;
}
else
{
     comando1;
     ...
     comandoN;
}
```

Caso, dentro do comando **if**, haja apenas um comando, então não é necessário usar chaves { }.

Exemplo:

```
if (numero_um == numero_dois)
  printf ("Sao iguais");
else
   if (numero_um > numero_dois)
      printf ("O maior: %f", numero_um);
   else
      printf ("Maior: %f", numero_dois);
```

**3.** Faça um algoritmo que mostre todos os números de 1 até 5.

| **Portugol** | **C** |
|---|---|
| `Algoritmo "Números"`<br><br>`Var numero : inteiro`<br>`inicio`<br><br>`   numero <- 1;`<br><br>`   Enquanto numero <= 5 faca`<br>`      Escreva (numero)`<br>`      numero <- numero + 1`<br>`   fimenquanto`<br><br>`Fimalgoritmo` | `/* Programa números */`<br><br>`#include <stdio.h>`<br><br>`int main(){`<br>`   int numero; /* contador */`<br><br>`   numero = 1; // iniciando a variável`<br><br>`   while (numero <= 5)`<br>`   {`<br>`      printf ("Numero: %i \n",numero);`<br>`      numero = numero + 1;`<br>`   }`<br>`   return (0);`<br>`}// fim do programa` |

- Os textos entre **/*** ***/** e os textos após // são comentários em C. Portanto há duas formas de delimitar (indicar) os comentários em C: usando **/*** ***/** ou //. A forma // também é usada em C++ (versão da linguagem C que dá suporte à orientação a objetos.
- O **\n** dentro do printf indica que deve pular para a próxima linha depois de escrever o número.

**4.** Faça um algoritmo que mostre a soma dos 10 primeiros números inteiros.

| **Portugol** | **C** |
|---|---|
| `algoritmo "Soma dez primeiros números"`<br><br>`var soma, N: inteiro`<br><br>`inicio`<br>`  soma <- 0`<br><br>`  para N de 1 ate 10 faca`<br>`  soma <- soma + N`<br>`  fimpara`<br><br>`  escreva (soma)`<br><br>`fimalgoritmo` | `#include <stdio.h>`<br>`// Soma dos 10 primeiros números`<br>`int main()`<br>`{`<br>`  int N, soma;`<br>`  soma = 0;`<br><br>`  for (N = 1; N <= 10; N = N + 1)`<br>`  {`<br>`  soma = soma + N;`<br>`  }`<br>`  printf ("Total: %i \n",soma);`<br>`  return (0);`<br>`}` |

A forma geral do laço **for** é a seguinte:

```
for (iniciação do contador ; condição de parada; incremento)
{
    comandos
}
```

Como é muito comum o incremento de um em um, em C, expressões de incremento como N = N + 1 podem ser substituídas por N++.

```
for (N = 1; N <= 10; N++)
```

Pode existir mais de uma expressão de *iniciação* e de *incremento* na estrutura **for**. Estas expressões devem ser separadas por vírgula (,). Mas atenção: **não pode** haver mais de uma expressão de *condição*. Exemplo:

```
for(i=0, j=10; i<10; i++, j--)
```

Também não é necessário especificar todos os parâmetros do laço **for**. Por exemplo, no trecho abaixo, a ausência de condições de iniciação, parada e incremento, causarão repetição infinita (loop infinito).

```
for(;;)
{
   printf("Este loop rodará eternamente!\n");
}
```

**5.** Leia números e some-os até o usuário digitar zero. Ao final mostre a soma dos números lidos. Use **repita-até**.

| **Portugol** | **C** |
|---|---|
| `algoritmo "semnome"`<br><br>`var N, soma : real`<br><br>`Inicio`<br>`  soma <- 0`<br><br>`  repita`<br>`  Leia (N)`<br>`  soma <- soma + N`<br>`  ate N = 0`<br><br>`  escreva (soma)`<br><br>`fimalgoritmo.` | `#include <stdio.h>`<br>`// Soma números até digitar zero`<br>`int main()`<br>`{`<br>`  float N, soma;`<br>`  soma = 0;`<br><br>`  do`<br>`  {`<br>`  scanf ("%f", &N);`<br>`  soma = soma + N;`<br>`  }`<br>`  while (N != 0);`<br><br>`  printf ("Total: %f",soma);`<br><br>`  return (0);`<br>`}` |

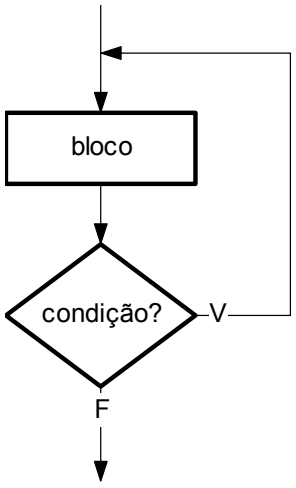
O **do – while** é uma estrutura básica de repetição condicional. Permite a execução de um bloco de instruções repetidamente. Sua sintaxe é a seguinte:

```
do{
  bloco
}while(condição);
```

onde: *condição* é uma expressão lógica ou numérica.

*bloco* é um conjunto de instruções.

Esta estrutura faz com que o bloco de instruções seja executado pelo menos uma vez. Após a execução do bloco, a condição é avaliada. Se a condição é **verdadeira** o bloco é executado outra vez, caso contrário a repetição é terminada. O fluxograma desta estrutura é mostrado abaixo.



**6.** Faça um programa que leia o número do mês (de 1 até 12) e mostre quantos dias ele tem.

```
#include <stdio.h>
int main ()
{
  int num;

  printf ("Digite um numero: ");
  scanf ("%i",&num); /* pega um caracter do teclado */

  switch (num)
  {
    case 1:
    case 3:
    case 5:
    case 7:
    case 8:
    case 10:
    case 12:
       printf ("\n\n Mes com 31 dias.\n");
       break;
    case 2:
       printf ("\n\n Mes com 28 ou 29 dias.\n");
       break;
    case 4:
    case 6:
    case 9:
    case 11:
       printf ("\n\n Mes com 30 dias.\n");
       break;
    default:
       printf ("\n\n Digitou mes errado.\n");
  }
  return (0);
}
```

Observe que em C os comandos podem estar todos em uma mesma linha (o ponto e vírgula ; é quem define o final do comando). Assim, mesmo programa poderia ser escrito de uma forma mais compacta:

```
#include <stdio.h>
int main ()
{
  int num;
  printf ("Digite um numero: ");
  scanf ("%i",&num); /* pega um caracter do teclado */

  switch (num)
  {
    case 1: case 3: case 5: case 7: case 8: case 10: case 12:
       printf ("\n\n Mes com 31 dias.\n");
       break;
    case 2:
       printf ("\n\n Mes com 28 ou 29 dias.\n");
       break;
    case 4: case 6: case 9: case 11:
       printf ("\n\n Mes com 30 dias.\n");
       break;
    default:
       printf ("\n\n Digitou mes errado.\n");
  }
  return (0);
}
```

**Exercícios**

1. Faça um programa que leia um número e mostre se é primo ou não.

```
#include <stdio.h>

int main ()
{
   int n, c, qtde;
   scanf ("%i",&n);
   qtde = 0;

   for (c=1; c<=n; c++)
      if ((n % c) == 0)
         qtde = qtde + 1;
   if (qtde > 2)
      printf ("Não e primo\n");
   else
      printf ("E primo\n");

   return (0);
}
```

2. Faça um programa que leia um número e mostre a soma dos seus divisores.

```
#include <stdio.h>

int main ()
{
   int n, c, soma;
   scanf ("%i",&n);
   soma = 0;

   for (c=1; c<=n; c++)
      if ((n % c) == 0)
         soma = soma + c;

   printf ("Soma dos divisores de %i = %i\n",n,soma);

   return (0);
}
```

3. Leia o valor de N salários e mostre quantos são acima de 1000,00 e quantos são abaixo de 400,00.

```
#include <stdio.h>

int main ()
{
   int n, c, acima, abaixo;
   float salario;

   scanf ("%i",&n);

   acima = abaixo = 0;   // Observe!!! Esta atribuindo zero a todos.

   for (c=1; c<=n; c++) {
      printf ("Salario %i: ",c); scanf("%f",&salario);
      if (salario > 1000.00) acima++;
      if (salario < 400.00)  abaixo++;
   }
   printf ("Total: %i Acima de 1000,00 e %i abaixo de 400,00\n",acima,abaixo);

   return (0);
}
```

4. Leia N números e mostre: o maior número lido; o menor número lido; a média dos números lidos.

```
#include <stdio.h>

int main ()
{
   int qtde, c;
   float num, maior, menor, soma;

   printf ("Quantidade de números: "); scanf("%i",&qtde);
   printf ("Numero 1: "); scanf("%f",&num);
   soma = maior = menor = num;

   for (c=2; c<=qtde; c++) {
      printf ("Numero %i: ",c); scanf("%f",&num);
      soma += num;        // Observe!!! Equivale a soma = soma + num;
      if (num > maior)
         maior = num;
      if (num < menor)
         menor = num;
   }
   printf ("Maior: %f\nMenor: %f\nSoma: %f\n",maior, menor, soma);

   return (0);
}
```

5. Some os números que o usuário digitou até que ele digite zero. Ao final mostre: quantidade de números digitados; a soma e a média dos números.

```
#include <stdio.h>

int main ()
{
   int qtde, c;
   float num, soma;

   soma = qtde = 0;
   num  = 1;

   while (num != 0)
   {
      qtde++;
      printf ("Numero %i: ",qtde); scanf("%f",&num);
       soma += num;
   }

   printf ("Quantidade: %i\nSoma: %f\nMedia: %f\n",qtde-1, soma, soma/(qtde-1));

   return (0);
}
```

6. Leia um número e mostre a raiz quadrada dele. Use a seguinte relação matemática: $\sqrt[b]{a} = a^{1/b} = e^{(1/b \log(a))}$.

```
#include <stdio.h>
#include <math.h>     //essa biblioteca é necessária para as funções exp() e log()

int main ()
{
   float num, raiz;
   printf ("Numero: "); scanf("%f",&num);

   raiz = exp(1.0 / 2 * log(num));   //ou seja: raiz (num) = e^(1/2 log(num))

   printf ("%f\n",raiz);
   return (0);
}
```

**Observação:** Ao compilar com **gcc** no Linux é necessário indicar o uso da biblioteca matemática:

```
$ gcc teste.c –o teste -lm
```