RUDIMENTOS DE LINGUAGEM C

2ª série de exercícios

Referência: Curso C da UFMG
(o arquivo .pdf ou as páginas html estão no arquivo "Material de apoio.zip")

---

1. Como compilar programas usando o DevC++?

Uma vez instalado o programa, você pode criar um novo programa através da opção: File -> New Source File. Na janela que aparece, você digita o seu programa. Veja que o DevC++ oferece para você um programa inicial que você pode apagar, se quiser. Uma vez digitado o seu programa, você pode compila-lo e executá-lo através da opção: Execute -> Compile and Run, ou digitando diretamente Cntrl+F10, ou buscando a opção "Compile and Run" na barra de ferramentas. Note que se você fizer isto para um programa que não espera nenhuma entrada do usuário, o programa será executado, terminará sua execução, e a janela onde ele está executando será automaticamente fechada ao seu final. Com isto, você não conseguirá ver a saída do programa. Para evitar que isto aconteça, você pode modificar seus programas incluindo as linhas:

#include <stdlib.h> no início do programa
e
system("pause"); antes do return (0);

como feito no programa abaixo:

```
#include <stdlib.h>
#include <stdio.h>
/* Um Primeiro Programa */
int main ()
{
  printf ("Ola! Eu estou vivo!\n");
  system("pause");
  return(0);
}
```

2. O C é "Case Sensitive"

```
#include <stdlib.h>
#include <stdio.h>
/* soma de N termos da serie harmonica 1+1/2+1/3+1/4+... */
int main ()
{
  int N, i; double s;      // declarações
  printf ("Entre com o numero de termos:\n");
  scanf("%d", &N);
  printf("N = %d", N);     // checando a entrada
  s=0;
  for (i=1;i<=N;i++)s = s + 1.0/(double) i;    // o "cast" (modelador) : passando de inteiro para double
  printf("\nSoma dos %d primeiros termos da serie harmonica = %lf", N, s);  // lf = long float = double
  system("pause");
  return(0);
}
```

```
// 1 byte = 8 bits
// An int variable can store an integer value in the range −32768 to +32767.
// A float number has about seven digits of precision and a range of about 1.E−36 to 1.E+36.
// A float takes four bytes to store.
// A double number has about 13 digits of precision and a range of about 1.E−303 to 1.E+303.
// A double takes eight bytes to store.
```

Nas aplicações comuns, os números com ponto flutuante são todos tratados no C como double.

i++ é equivalente a i = i+1
s += i é equivalente a s = s+i

3. Determinando a precisão de máquina

```
#include <stdlib.h>
#include <stdio.h>
/* encontrando o menor numero positivo eps para o qual 1 + eps != 1  */
int main ()
{
  int i; double eps;

 eps = 1.0; i = 0;
 while ((1.0 + eps) != 1.0) {  eps = eps/2.0;
                               i++; };
  printf("\neps = %g  (%d)", eps, i);
  system("pause");
  return(0);
}
```

(o resultado no meu PC foi eps = $5{,}4\times10^{-20}$, após 64 divisões)

EXERCICIO 1: Faça um programa para obter o maior valor de uma variável double x para o qual $x+1 \neq x$.

4. A estrutura em funções do C

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

/* a função abaixo funciona corretamente para n <= 32767 */
/* F(32767) = 9.3e-10 */
double F( int n )
{
 double sinal;
 sinal = ((n % 2)==0) ? -1.0 : 1.0;
 return(sinal/(double) (n*n));
}

int main ()
{
   int n;
   double s, a, epsilon;
  /* somando a serie F(n) até que o termo geral fique menor do que epsilon */
  printf("\nEntre com o erro maximo epsilon:\n");
  scanf("%lf", &epsilon);
  printf("   epsilon = %g\n", epsilon);
  if(epsilon < 1.0e-9){ printf("\n Este programa funciona somente se epsilon >= 1.0e-9\n");
                        system("pause");
                        return(1);
                       }
  s = 0.0;
  n = 1;
  do {  a = F(n);
        s += a;
        n++; } while (fabs(a) >= epsilon);
  printf("\nsoma dos %ld primeiros termos = %g \n", n, s);
  system("pause");
  return(0);
}
```

(para epsilon = 10^{-6}, a soma dos primeiros 1002 termos é 0,822468; essa série converge para $\pi^2/12$ =0,822467)

EXERCICIO 2: Modifique o programa acima para que ele some termos da série até que a diferença na soma ao se acrescentar um novo termo seja menor do que epsilon.

EXERCICIO 3: Modifique o programa acima para que ele some termos da série até que a diferença relativa na soma ao se acrescentar um novo termo seja menor do que epsilon.

DESAFIO (muito difícil !!!) : Modifique o programa acima de modo que ele funcione até a precisão de máquina. A dificuldade é que voce não poderá usar inteiros no cálculo da função, porque o número de termos pode exceder o valor máximo que uma variável inteira pode guardar. E lembre-se de que, para x suficientemente grande, o computador vai fazer 1 + x = x !!!

5. Os tipos de dados no C

Tipo	Num de bits	Formato para leitura com scanf	Intervalo	
			Inicio	Fim
char	8	%c	−128	127
unsigned char	8	%c	0	255
signed char	8	%c	−128	127
int	16	%i	−32.768	32.767
unsigned int	16	%u	0	65.535
signed int	16	%i	−32.768	32.767
short int	16	%hi	−32.768	32.767
unsigned short int	16	%hu	0	65.535
signed short int	16	%hi	−32.768	32.767
long int	32	%li	−2.147.483.648	2.147.483.647
signed long int	32	%li	−2.147.483.648	2.147.483.647
unsigned long int	32	%lu	0	4.294.967.295
float	32	%f	3,4E-38	3.4E+38
double	64	%lf	1,7E-308	1,7E+308
long double	80	%Lf	3,4E-4932	3,4E+4932

códigos de formato do printf :

Código	Formato
%c	Um caracter (char)
%d	Um número inteiro decimal (int)
%i	O mesmo que %d
%e	Número em notação científica com o "e"minúsculo
%E	Número em notação científica com o "e"maiúsculo
%f	Ponto flutuante decimal
%g	Escolhe automaticamente o melhor entre %f e %e
%G	Escolhe automaticamente o melhor entre %f e %E
%o	Número octal
%s	String
%u	Decimal "unsigned" (sem sinal)
%x	Hexadecimal com letras minúsculas
%X	Hexadecimal com letras maiúsculas
%%	Imprime um %
%p	Ponteiro

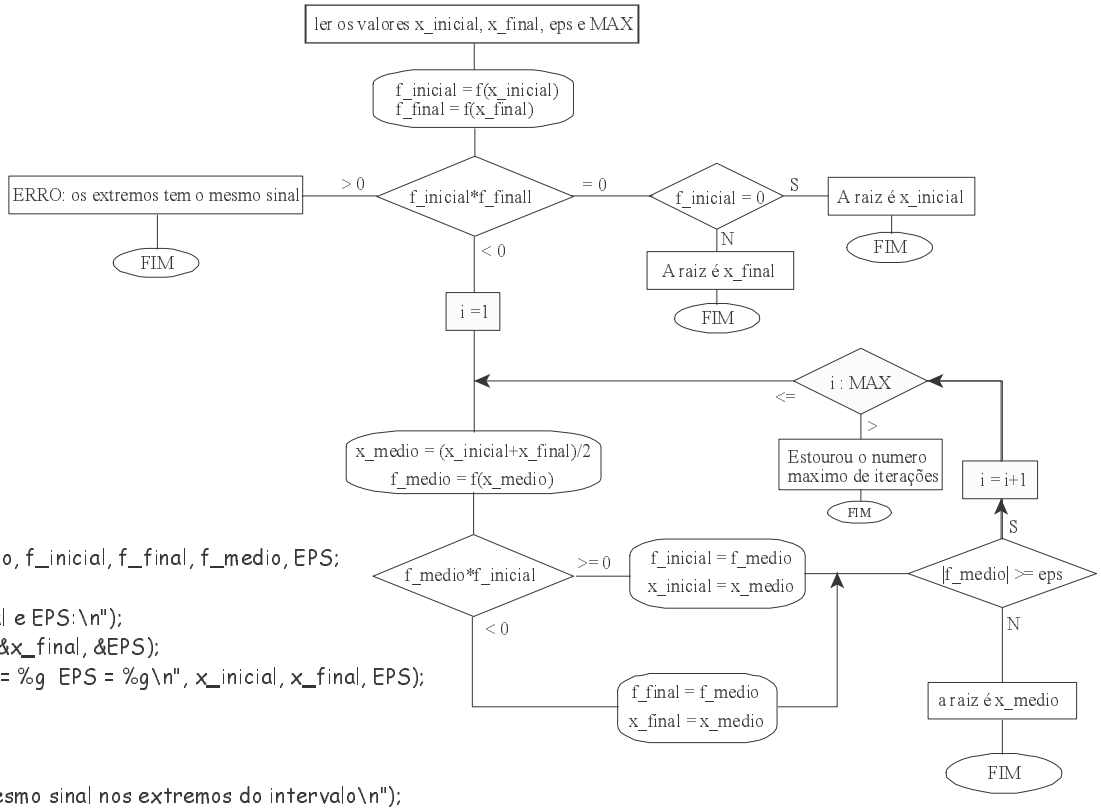
É possível também indicar o tamanho do campo, justificação e o número de casas decimais. Para isto usa-se códigos colocados entre o **%** e a letra que indica o tipo de formato.
Um inteiro indica o tamanho mínimo, em caracteres, que deve ser reservado para a saída. Se colocarmos então **%5d** estamos indicando que o campo terá cinco caracteres de comprimento *no mínimo*. Se o inteiro precisar de mais de cinco caracteres para ser exibido então o campo terá o comprimento necessário para exibi-lo. Se o comprimento do inteiro for menor que cinco então o campo terá cinco de comprimento e será preenchido com espaços em branco. Se se quiser um preenchimento com zeros pode-se colocar um zero antes do número. Temos então que **%05d** reservará cinco casas para o número e se este for menor então se fará o preenchimento com zeros.
O alinhamento padrão é à direita. Para se alinhar um número à esquerda usa-se um sinal - antes do número de casas. Então **%-5d** será o nosso inteiro com o número mínimo de cinco casas, só que justificado a esquerda.
Pode-se indicar o número de casas decimais de um número de ponto flutuante. Por exemplo, a notação **%10.4f** indica um ponto flutuante de comprimento total dez e com 4 casas decimais. Entretanto, esta mesma notação, quando aplicada a tipos como inteiros e strings indica o número mínimo e máximo de casas. Então **%5.8d** é um inteiro com comprimento mínimo de cinco e máximo de oito.

códigos de formato do scanf :

Código	Formato
%c	Um único caracter (char)
%d	Um número decimal (int)
%i	Um número inteiro
%hi	Um short int
%li	Um long int
%e	Um ponto flutuante
%f	Um ponto flutuante
%lf	Um double
%h	Inteiro curto
%o	Número octal
%s	String
%x	Número hexadecimal
%p	Ponteiro

6. Exemplo: encontrando uma solução de f(x) = 0 num dado intervalo pelo método da bisecção

A listagem abaixo é para a função $f(x) = x^3 - 2x^2 + x - 1$, que tem uma raiz no intervalo [0,2].




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

int MAX = 200;

double F( double x )
{
 return(-1.0+x*(1.0+x*(-2.0+x)));
}

int main ()
{
  int i;
  double x_inicial, x_final, x_medio, f_inicial, f_final, f_medio, EPS;

  printf("\nEntre x_inicial, x_final e EPS:\n");
  scanf("%lf %lf %lf", &x_inicial, &x_final, &EPS);
  printf("   x_inicial = %g  x_final = %g  EPS = %g\n", x_inicial, x_final, EPS);
  f_inicial = F(x_inicial);
  f_final = F(x_final);
  if ((f_inicial*f_final) > 0){
   printf("\n ERRO: f(x) tem o mesmo sinal nos extremos do intervalo\n");
   system("pause");
   return(1);
  }
  if (fabs(f_inicial) < EPS){
   printf("\n A raiz eh x = %g\n", x_inicial);
   system("pause");
   return(0);
  }
  if (fabs(f_final) < EPS){
   printf("\n A raiz eh x = %g\n", x_final);
   system("pause");
   return(0);
  }
  for(i=1; i<=MAX; i++){
   x_medio = (x_inicial+x_final)/2.0;
   f_medio = F(x_medio);
   if ((f_medio*f_inicial) >= 0.0){
    x_inicial = x_medio;
    f_inicial = f_medio;
   } else { x_final = x_medio;
            f_final = f_medio;
          };
   if (fabs(f_medio) < EPS){ printf("\n A raiz eh x = %g (%d iteracoes)\n", x_medio, i);
                             system("pause");
                             return(0);
                           }
  }
  printf("\ O numero maximo de iteracoes (%d) foi excedido \n", MAX);
  system("pause");
  return(2);
}
```

NOTE que o programa não corresponde exatamente aos passos do fluxograma.

O fluxograma ajuda a estabelecer os passos necessários, e a implementação é feita de modo a utilizar as facilidades da linguagem.

Caso fosse necessário, poderíamos fazer um fluxograma que correspondesse exatamente ao programa fonte.

EXERCICIO 4: Modifique o trecho de leitura para que o programa continue pedindo os dados até que f(x) tenha sinais opostos nos extremos do intervalo.

EXERCICIO 5: Refaça o loop de iterações usando o comando while.

EXERCICIO 6: Teste o programa para encontrar as raízes de:

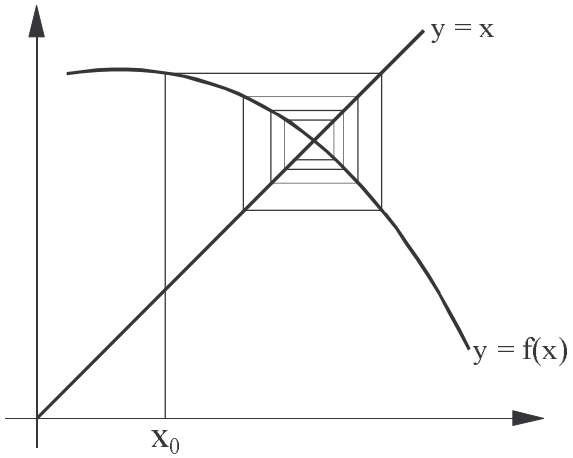
(a) $f(x) = x^4 + 10x^3 + x^2 + 5x + 1$, no intervalo $[-1, 0]$.

(b) $f(x) = x - e^{-x}$, no intervalo $[0,1]$

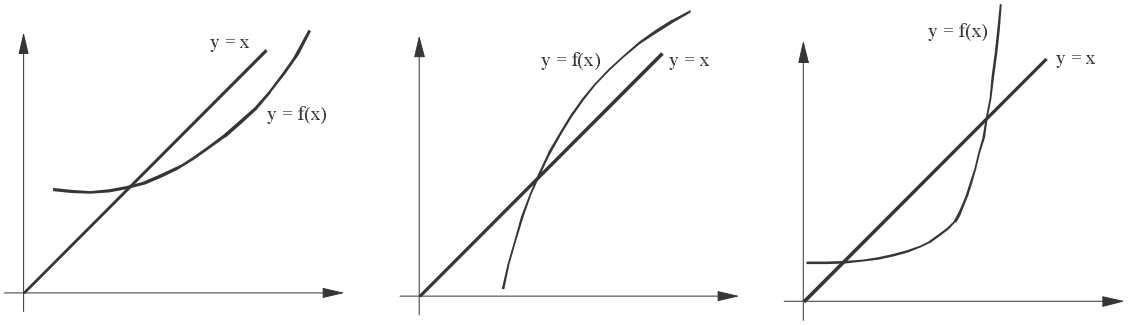
(c) $f(x) = x - 5\tan(x)$, no intervalo $[\pi, 3\pi/2]$

7. Encontrando o ponto fixo de x = f(x), a partir de um "chute" inicial

Observe, no gráfico ao lado, como as iterações $x_{n+1} = f(x_n)$, a partir de um valor inicial x_0, convergem para o ponto no qual $x = f(x)$.



EXERCICIO 7: Verifique se essas iterações vão convergir nos casos abaixo:



EXERCICIO 8: Faça um programa em C implementando essas iterações para $x = e^{-x}$. O usuário deve especificar o erro máximo absoluto e um número máximo de iterações.

---

8. O truque do maior valor e do menor valor

EXERCICIO 9: Faça um programa em C que estime o maior e o menor valor de uma função f(x) no intervalo $[x_1, x_2]$, amostrando esse intervalo em passos dx.

EXERCICIO 10: Usando o programa do Exercício 9, faça outro que determina o maior (ou o menor) valor de uma função f(x) no intervalo $[x_1, x_2]$, com uma dada precisão.
