Introdução à Computação II – AULA 05

BCC Noturno - EMA896115B

Prof. Rafael Oliveira
olivrap@gmail.com

Universidade Estadual Paulista
“Júlio de Mesquita Filho”
UNESP

Rio Claro 2014 (Sem 2)

Recordando – definição de funções

- Podemos definir nossas próprias funções
- Cada função tem
  - Nome que deve ser único
  - Tipo, que o tipo do valor retornado pela função
  - Parametros
    - Que por sua vez têm tipos também
- Definir protótipo antes de usar ou definir a função

Recordando – variáveis

- Locais: só existem no escopo da função em que foi definida
  - Diferentes funções podem ter variáveis locais com mesmo nome
  - Referem-se a variáveis diferentes.
- Globais: são variáveis compartilhadas por todas as funções
  - Todas usam a mesma variável
  - Não deve haver variável local com o mesmo nome de uma variável global

Comandos repetidos

- Escrever todos os números de 1 a 100
- Escrever todos os números de 1 até um valor N
- Calcular o fatorial de um número inteiro
- Calcular o valor do seno pela série de Taylor, até que o erro seja menor do que um valor ϵ

Função print_100

```
void print_100();

void print_100()
{
    printf("1\n");
    printf("2\n");
    printf("3\n");
    printf("4\n");
      ...
    printf("100\n");
}
```

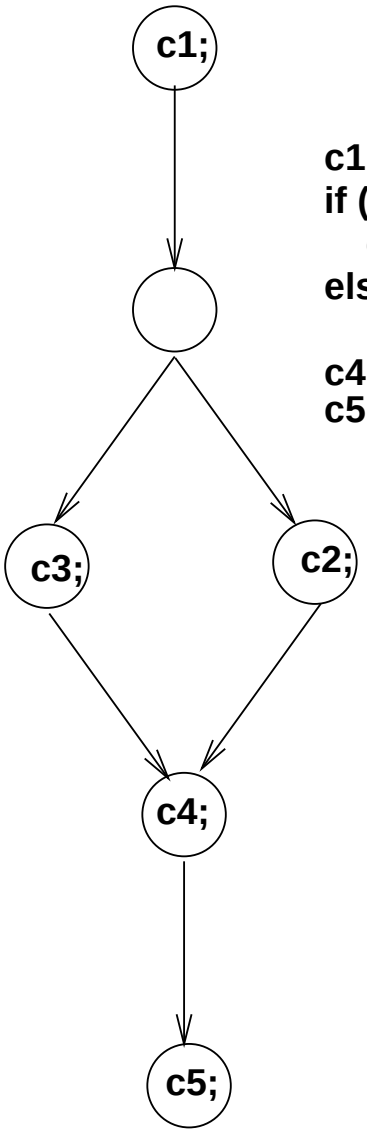
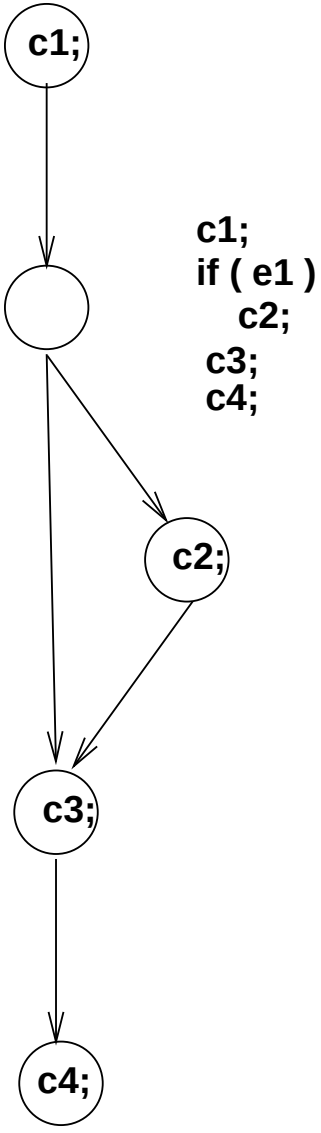
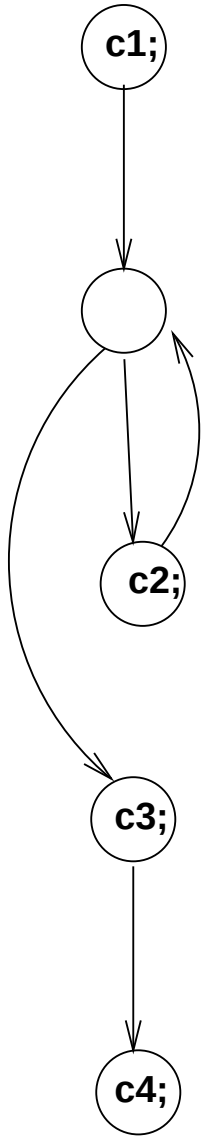
Função print_n

```
void print_n(int k);

void print_n(int k)
{
    printf("1\n");
    printf("2\n");
    printf("3\n");
    printf("4\n");
   ???
}
```

Função fatorial

```
int fatorial(int);

int fatorial(int k)
{
   ???
}
```

Comando `while`

- O comando *while* permite repetir a execução de alguns comandos enquanto uma condição for verdadeira
- ```
  comando antes;
  while ( condição )
  {
      C1;
      C2;
      C3;
  }
  comando depois;
  ```
- Executa comando antes; C1-C2-C3 (0 ou mais vezes); comando depois
  ```

Representação diagramática – sequência

```
c1;
c2;
c3;
c4;
```

Representação diagramática – if




```
c1;
if ( e1 )
    c2;
else
    c3;
c4;
c5;
```




```
c1;
if ( e1 )
    c2;
c3;
c4;
```

Representação diagramática – while




```
c1;
while ( e1 )
    c2;
 c3;
 c4;
```

Voltando ao fatorial

```
int fatorial(int k)
{
int r;

}
```

Voltando ao fatorial

```
int fatorial(int k)
{
int r;

   r = k;
}
```

Voltando ao fatorial

```
int fatorial(int k)
{
int r;

   r = k;
   while ( k > 1 )
   {

   }
   return r;
}
```

Voltando ao fatorial

```
int fatorial(int k)
{
int r;

   r = k;
   while ( k > 1 )
   {
       k = k - 1;
       r = r * k;
   }
   return r;
}
```

Executando o fatorial

```
int fatorial(int k)
{
int r;

   r = k;
   while ( k > 1 )
   {
       k = k - 1;
       r = r * k;
   }
   return r;
}
```

- Valor de k: 4
- Valor de r: 4
- Valor da condição: ???

Executando o fatorial

```
int fatorial(int k)
{
int r;

    r = k;
    while ( k > 1 )
    {
        k = k - 1;
        r = r * k;
    }
    return r;
}
```

- Valor de k: 4
- Valor de r: 4
- Valor da condição: verdadeira (1)

Executando o fatorial

```
int fatorial(int k)
{
int r;

   r = k;
   while ( k > 1 )
   {
       k = k - 1;
       r = r * k;
   }
   return r;
}
```

- Valor de k: ~~4~~ 3
- Valor de r: 4
- Valor da condição:

Executando o fatorial

```
int fatorial(int k)
{
int r;

   r = k;
   while ( k > 1 )
   {
       k = k - 1;
       r = r * k;
   }
   return r;
}
```

- Valor de k: ~~4~~ 3
- Valor de r: ~~4~~ 12
- Valor da condição:

Executando o fatorial

```
int fatorial(int k)
{
int r;

   r = k;
   while ( k > 1 )
   {
        k = k - 1;
        r = r * k;
   }
   return r;
}
```

- Valor de k: ~~4~~ 3
- Valor de r: ~~4~~ 12
- Valor da condição: verdadeira

Executando o fatorial

```
int fatorial(int k)
{
int r;

    r = k;
    while ( k > 1 )
    {
        k = k - 1;
        r = r * k;
    }
    return r;
}
```

- Valor de k: ~~4~~ ~~3~~ 2
- Valor de r: ~~4~~ ~~12~~ 24
- Valor da condição:

Executando o fatorial

```
int fatorial(int k)
{
int r;

   r = k;
   while ( k > 1 )
   {
        k = k - 1;
        r = r * k;
   }
   return r;
}
```

- Valor de k: ~~4~~ ~~3~~ 2
- Valor de r: ~~4~~ ~~12~~ 24
- Valor da condição: verdadeira

Executando o fatorial

```
int fatorial(int k)
{
int r;

   r = k;
   while ( k > 1 )
   {
        k = k - 1;
        r = r * k;
   }
   return r;
}
```

- Valor de k: ~~4~~ ~~3~~ ~~2~~ 1
- Valor de r: ~~4~~ ~~12~~ ~~24~~ 24
- Valor da condição:

Executando o fatorial

```
int fatorial(int k)
{
int r;

    r = k;
    while ( k > 1 )
    {
        k = k - 1;
        r = r * k;
    }
    return r;
}
```

- Valor de k: ~~4~~ ~~3~~ ~~2~~ 1
- Valor de r: ~~4~~ ~~12~~ ~~24~~ 24
- Valor da condição: falsa

Executando o fatorial

```
int fatorial(int k)
{
int r;

    r = k;
    while ( k > 1 )
    {
        k = k - 1;
        r = r * k;
    }
    return r;
}
```

- Valor de k: ~~4~~ ~~3~~ ~~2~~ 1
- Valor de r: ~~4~~ ~~12~~ ~~24~~ 24
- Valor da condição: falsa
- retorna o valor de r

O que vai no while

- Dentro do “corpo” do while pode ir qualquer comando
-
```
while ( condição 1 )
{
    if ( condição 2 )
       C1;
    else
       C2;
}
```

Exemplo - série de Taylor

- Computar o seno pela série de Taylor
- Usar a função fatorial
- Utilizar os 10 primeiros termos da série
-

$$sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} ...$$

Série de taylor

```
double seno(double x)
{
int cont; // conta termo
double numerador;
double termo; // termo corrente
double s; // valor do seno
int denominador;
}
```

Série de taylor – inicialização

```
double seno(double x)
{
int cont; // conta termo
double numerador;
double termo; // termo corrente
double s; // valor do seno
int denominador;
     cont = 1;
     numerador = x;
     denominador = 1;
     s = 0.0;
}
```

Série de taylor – computar termo

```
double seno(double x)
{
int cont; // conta termo
double numerador;
double termo; // termo corrente
double s; // valor do seno
int denominador;
     cont = 1;
     numerador = x;
     denominador = 1;
     s = 0.0;
     while (cont <= 10 )
     {
         // computar termos
     }
}
```

Série de taylor – computar termos

```
double seno(double x)
{
int cont; // conta termo
double numerador;
double termo; // termo corrente
double s; // valor do seno
int denominador;
     cont = 1;
     numerador = x;
     denominador = 1;
     s = 0.0;
     while (cont <= 10 )
     {
         termo = numerador / (double) fatorial(denominador);
         if (cont % 2 == 0 )
            s = s - termo;
         else
            s = s + termo;
         numerador = numerador * x * x;
         denominador = denominador + 2;
         cont = cont + 1;
     }
}
```

Série de taylor – o que faltava?

```
double seno(double x)
{
int cont; // conta termo
double numerador;
double termo; // termo corrente
double s; // valor do seno
int denominador;
     cont = 1;
     numerador = x;
     denominador = 1;
     s = 0.0;
     while (cont <= 10 )
     {
         termo = numerador / (double) fatorial(denominador);
         if (cont % 2 == 0 )
            s = s - termo;
         else
            s = s + termo;
         numerador = numerador * x * x;
         denominador = denominador + 2;
         cont = cont + 1;
     }
     return s;
}
```

Exercício

- Modificar a função do seno, de modo que o erro seja sempre menor do que 0.0001
- Modificar a função do seno de modo que o erro desejado seja um parâmetro da função

Whiles aninhados

- Podem aparecer diversos níveis de whiles

```
 while ( condição 1 )
{
    while ( condição 2 )
    {
       C1; C2;
    }
}
```

- Escrever um programa que mostra

```
*
**
***
****
*****
******
```

Whiles aninhados – exemplo

```
void print_arvore(int n)
{
int linha;
int coluna;
     linha = 1;
     while ( linha <= n )
     {
        // tem que imprimir cada uma
        // das linhas inteiras
     }
     return;
}
```

Whiles aninhados – exemplo

```
void print_arvore(int n)
{
int linha;
int coluna;
     linha = 1;
     while ( linha <= n )
     {
         coluna = 1;
         while ( coluna <= linha )
         {
            // imprimir cada *
         }
         printf("\n");
         linha++;
     }
     return;
}
```

Whiles aninhados – exemplo

```
void print_arvore(int n)
{
int linha;
int coluna;
     linha = 1;
     while ( linha <= n )
     {
         coluna = 1;
         while ( coluna <= linha )
         {
             printf("*");
             coluna++;
         }
         printf("\n");
         linha++;
     }
     return;
}
```

Para registro – void

- O tipo “**void**” indica que a função não retorna nenhum valor
- É util quando a função “faz alguma coisa” mas não computa nenhum valor que seja de interesse para quem chama a função
- Não se deve fazer

  ```
  k = print_arvore(x);
  ```

  se a função for do tipo `void`
- O comando `i++` adiciona 1 no valor da variável `i`
- O comando `i--` subtrai 1 do valor da variável `i`

Exercícios

- Escreva funções que mostrem as seguintes árvores, com qualquer número de linhas:
  - 
  - 

Exercício (2)

- Escreva um programa que leia um número inteiro N. Depois ele deve ler N números inteiros e dizer quantos são ímpares.

Solução (2)

```
int main()
{
int i, n, k, cont;
   printf("Quantos numeros serao digitados? ");
   scanf("%d", &n);
   cont = 0;
   i = 1;
   while ( i <= n )
   {
         printf("Digite ==> ");
         scanf("%d", &k);
         if ( k % 2 != 0 )
            cont++;
         i++;
   }
   printf("Foram %d impares\n", cont);
}
```

Recordando – definição de funções

- Podemos definir nossas próprias funções
- Cada função tem
  - Nome que deve ser único
  - Tipo, que o tipo do valor retornado pela função
  - Parametros
    - Que por sua vez têm tipos também
- Definir protótipo antes de usar ou definir a função

Recordando – variáveis

- Locais: só existem no escopo da função em que foi definida
  - Diferentes funções podem ter variáveis locais com mesmo nome
  - Referem-se a variáveis diferentes.
- Globais: são variáveis compartilhadas por todas as funções
  - Todas usam a mesma variável
  - Não deve haver variável local com o mesmo nome de uma variável global

Comandos repetidos

- Escrever todos os números de 1 a 100
- Escrever todos os números de 1 até um valor N
- Calcular o fatorial de um número inteiro
- Calcular o valor do seno pela série de Taylor, até que o erro seja menor do que um valor ϵ

Função print_100

```
void print_100();

void print_100()
{
    printf("1\n");
    printf("2\n");
    printf("3\n");
    printf("4\n");
      ...
    printf("100\n");
}
```

Função print_n

```
void print_n(int k);

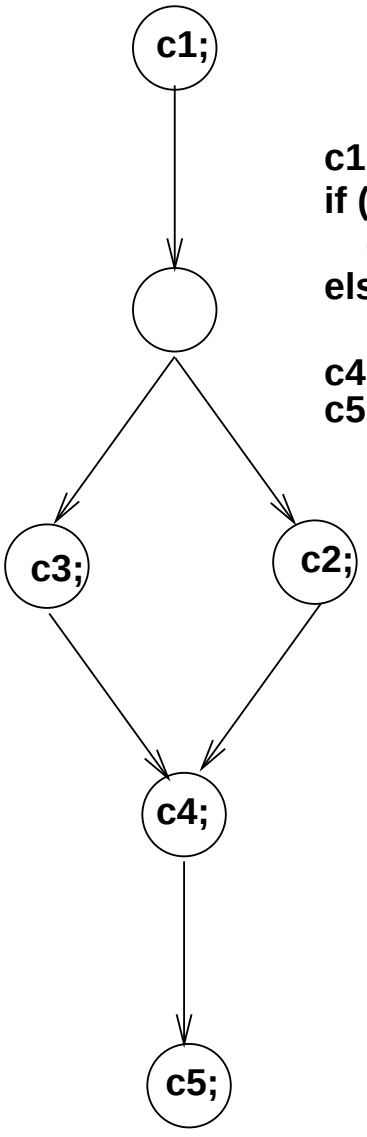
void print_n(int k)
{
    printf("1\n");
    printf("2\n");
    printf("3\n");
    printf("4\n");
   ???
}
```

Função fatorial

```
int fatorial(int);

int fatorial(int k)
{
   ???
}
```

Comando `while`

- O comando *while* permite repetir a execução de alguns comandos enquanto uma condição for verdadeira
- ```
  comando antes;
  while ( condição )
  {
      C1;
      C2;
      C3;
  }
  comando depois;
  ```
- Executa comando antes; C1-C2-C3 (0 ou mais vezes); comando depois
  ```

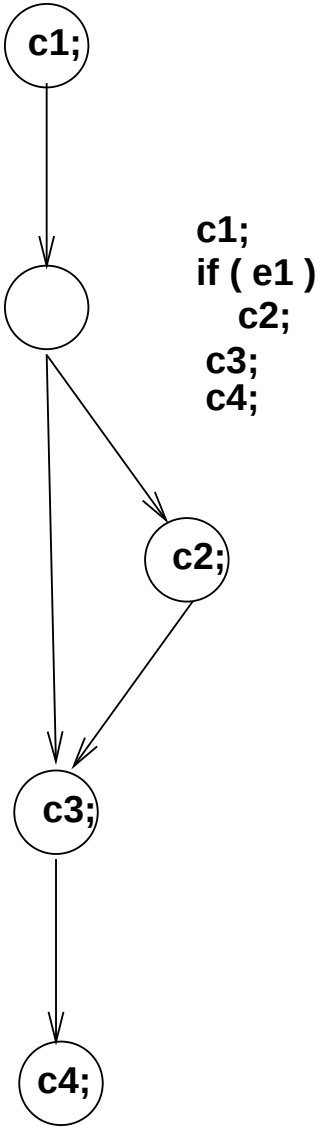
Representação diagramática – sequência

```
c1;
c2;
c3;
c4;
```

Representação diagramática – if

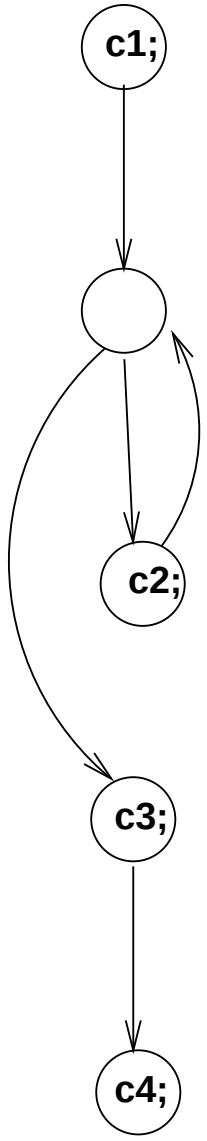



```
c1;
if ( e1 )
    c2;
else
    c3;
c4;
c5;
```




```
c1;
if ( e1 )
    c2;
c3;
c4;
```

Representação diagramática – while




```
c1;
while ( e1 )
    c2;
 c3;
 c4;
```

Voltando ao fatorial

```
int fatorial(int k)
{
int r;

}
```

Voltando ao fatorial

```
int fatorial(int k)
{
int r;

   r = k;
}
```

Voltando ao fatorial

```
int fatorial(int k)
{
int r;

   r = k;
   while ( k > 1 )
   {

   }
   return r;
}
```

Voltando ao fatorial

```
int fatorial(int k)
{
int r;

   r = k;
   while ( k > 1 )
   {
       k = k - 1;
       r = r * k;
   }
   return r;
}
```

Executando o fatorial

```
int fatorial(int k)
{
int r;

   r = k;
   while ( k > 1 )
   {
       k = k - 1;
       r = r * k;
   }
   return r;
}
```

- Valor de k: 4
- Valor de r: 4
- Valor da condição: ???

Executando o fatorial

```
int fatorial(int k)
{
int r;

    r = k;
    while ( k > 1 )
    {
        k = k - 1;
        r = r * k;
    }
    return r;
}
```

- Valor de k: 4
- Valor de r: 4
- Valor da condição: verdadeira (1)

Executando o fatorial

```
int fatorial(int k)
{
int r;

   r = k;
   while ( k > 1 )
   {
       k = k - 1;
       r = r * k;
   }
   return r;
}
```

- Valor de k: ~~4~~ 3
- Valor de r: 4
- Valor da condição:

Executando o fatorial

```
int fatorial(int k)
{
int r;

   r = k;
   while ( k > 1 )
   {
       k = k - 1;
       r = r * k;
   }
   return r;
}
```

- Valor de k: ~~4~~ 3
- Valor de r: ~~4~~ 12
- Valor da condição:

Executando o fatorial

```
int fatorial(int k)
{
int r;

   r = k;
   while ( k > 1 )
   {
        k = k - 1;
        r = r * k;
   }
   return r;
}
```

- Valor de k: ~~4~~ 3
- Valor de r: ~~4~~ 12
- Valor da condição: verdadeira

Executando o fatorial

```
int fatorial(int k)
{
int r;

    r = k;
    while ( k > 1 )
    {
        k = k - 1;
        r = r * k;
    }
    return r;
}
```

- Valor de k: ~~4~~ ~~3~~ 2
- Valor de r: ~~4~~ ~~12~~ 24
- Valor da condição:

Executando o fatorial

```
int fatorial(int k)
{
int r;

   r = k;
   while ( k > 1 )
   {
        k = k - 1;
        r = r * k;
   }
   return r;
}
```

- Valor de k: ~~4~~ ~~3~~ 2
- Valor de r: ~~4~~ ~~12~~ 24
- Valor da condição: verdadeira

Executando o fatorial

```
int fatorial(int k)
{
int r;

   r = k;
   while ( k > 1 )
   {
        k = k - 1;
        r = r * k;
   }
   return r;
}
```

- Valor de k: ~~4~~ ~~3~~ ~~2~~ 1
- Valor de r: ~~4~~ ~~12~~ ~~24~~ 24
- Valor da condição:

Executando o fatorial

```
int fatorial(int k)
{
int r;

    r = k;
    while ( k > 1 )
    {
        k = k - 1;
        r = r * k;
    }
    return r;
}
```

- Valor de k: ~~4~~ ~~3~~ ~~2~~ 1
- Valor de r: ~~4~~ ~~12~~ ~~24~~ 24
- Valor da condição: falsa

Executando o fatorial

```
int fatorial(int k)
{
int r;

    r = k;
    while ( k > 1 )
    {
        k = k - 1;
        r = r * k;
    }
    return r;
}
```

- Valor de k: ~~4~~ ~~3~~ ~~2~~ 1
- Valor de r: ~~4~~ ~~12~~ ~~24~~ 24
- Valor da condição: falsa
- retorna o valor de r

O que vai no while

- Dentro do “corpo” do while pode ir qualquer comando
-
```
while ( condição 1 )
{
    if ( condição 2 )
       C1;
    else
       C2;
}
```

Exemplo - série de Taylor

- Computar o seno pela série de Taylor
- Usar a função fatorial
- Utilizar os 10 primeiros termos da série
-

$$sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} ...$$

Série de taylor

```
double seno(double x)
{
int cont; // conta termo
double numerador;
double termo; // termo corrente
double s; // valor do seno
int denominador;
}
```

Série de taylor – inicialização

```
double seno(double x)
{
int cont; // conta termo
double numerador;
double termo; // termo corrente
double s; // valor do seno
int denominador;
     cont = 1;
     numerador = x;
     denominador = 1;
     s = 0.0;
}
```

Série de taylor – computar termo

```
double seno(double x)
{
int cont; // conta termo
double numerador;
double termo; // termo corrente
double s; // valor do seno
int denominador;
     cont = 1;
     numerador = x;
     denominador = 1;
     s = 0.0;
     while (cont <= 10 )
     {
         // computar termos
     }
}
```

Série de taylor – computar termos

```
double seno(double x)
{
int cont; // conta termo
double numerador;
double termo; // termo corrente
double s; // valor do seno
int denominador;
     cont = 1;
     numerador = x;
     denominador = 1;
     s = 0.0;
     while (cont <= 10 )
     {
         termo = numerador / (double) fatorial(denominador);
         if (cont % 2 == 0 )
            s = s - termo;
         else
            s = s + termo;
         numerador = numerador * x * x;
         denominador = denominador + 2;
         cont = cont + 1;
     }
}
```

Série de taylor – o que faltava?

```
double seno(double x)
{
int cont; // conta termo
double numerador;
double termo; // termo corrente
double s; // valor do seno
int denominador;
     cont = 1;
     numerador = x;
     denominador = 1;
     s = 0.0;
     while (cont <= 10 )
     {
         termo = numerador / (double) fatorial(denominador);
         if (cont % 2 == 0 )
            s = s - termo;
         else
            s = s + termo;
         numerador = numerador * x * x;
         denominador = denominador + 2;
         cont = cont + 1;
     }
     return s;
}
```

Exercício

- Modificar a função do seno, de modo que o erro seja sempre menor do que 0.0001
- Modificar a função do seno de modo que o erro desejado seja um parâmetro da função

Whiles aninhados

- Podem aparecer diversos níveis de whiles

```
 while ( condição 1 )
{
    while ( condição 2 )
    {
       C1; C2;
    }
}
```

- Escrever um programa que mostra

```
*
**
***
****
*****
******
```

Whiles aninhados – exemplo

```
void print_arvore(int n)
{
int linha;
int coluna;
     linha = 1;
     while ( linha <= n )
     {
        // tem que imprimir cada uma
        // das linhas inteiras
     }
     return;
}
```

Whiles aninhados – exemplo

```
void print_arvore(int n)
{
int linha;
int coluna;
     linha = 1;
     while ( linha <= n )
     {
         coluna = 1;
         while ( coluna <= linha )
         {
            // imprimir cada *
         }
         printf("\n");
         linha++;
     }
     return;
}
```

Whiles aninhados – exemplo

```
void print_arvore(int n)
{
int linha;
int coluna;
     linha = 1;
     while ( linha <= n )
     {
         coluna = 1;
         while ( coluna <= linha )
         {
             printf("*");
             coluna++;
         }
         printf("\n");
         linha++;
     }
     return;
}
```

Para registro – void

- O tipo “**void**” indica que a função não retorna nenhum valor
- É util quando a função “faz alguma coisa” mas não computa nenhum valor que seja de interesse para quem chama a função
- Não se deve fazer

  ```
  k = print_arvore(x);
  ```

  se a função for do tipo `void`
- O comando `i++` adiciona 1 no valor da variável `i`
- O comando `i--` subtrai 1 do valor da variável `i`

Exercícios

- Escreva funções que mostrem as seguintes árvores, com qualquer número de linhas:
  - 
  - 

Exercício (2)

- Escreva um programa que leia um número inteiro N. Depois ele deve ler N números inteiros e dizer quantos são ímpares.

Solução (2)

```
int main()
{
int i, n, k, cont;
   printf("Quantos numeros serao digitados? ");
   scanf("%d", &n);
   cont = 0;
   i = 1;
   while ( i <= n )
   {
         printf("Digite ==> ");
         scanf("%d", &k);
         if ( k % 2 != 0 )
            cont++;
         i++;
   }
   printf("Foram %d impares\n", cont);
}
```