CI1055: Algoritmos e Estruturas de Dados I

Profs. Drs. Marcos Castilho Carmem Hara e Bruno Muller Jr

Departamento de Informática/UFPR

31 de julho de 2020

Aplicação das técnicas elementares

- Fatorial
- Fibonacci revisado
- Palídromos

Fatorial

Problema: Imprimir o valor do fatorial de todos os números entre 1 e n, sendo n fornecido pelo usuário.

- quais as semelhanças deste com os já vistos?
- (sub-problema) como calcula o fatorial de n?

$$fat(n) = n \times (n-1) \times (n-2) \times \cdots \times 3 \times 2 \times 1$$

- aplicar a técnica de acumuladores;
- idéia: colocar a solução para o sub-problema em um laço;
- primeiro passo: implementar o sub-problema:

Fatorial

Problema: Imprimir o valor do fatorial de todos os números entre 1 e n, sendo n fornecido pelo usuário.

- quais as semelhanças deste com os já vistos?
- (sub-problema) como calcula o fatorial de n?

$$fat(n) = n \times (n-1) \times (n-2) \times \cdots \times 3 \times 2 \times 1$$

  - aplicar a técnica de acumuladores;
- idéia: colocar a solução para o sub-problema em um laço;
- primeiro passo: implementar o sub-problema:

Fatorial

Problema: Imprimir o valor do fatorial de todos os números entre 1 e n, sendo n fornecido pelo usuário.

- quais as semelhanças deste com os já vistos?
- (sub-problema) como calcula o fatorial de n?

$$fat(n) = n \times (n-1) \times (n-2) \times \cdots \times 3 \times 2 \times 1$$

  - aplicar a técnica de acumuladores;
- idéia: colocar a solução para o sub-problema em um laço;
- primeiro passo: implementar o sub-problema:

Fatorial

Problema: Imprimir o valor do fatorial de todos os números entre 1 e n, sendo n fornecido pelo usuário.

- quais as semelhanças deste com os já vistos?
- (sub-problema) como calcula o fatorial de n?

$$fat(n) = n \times (n-1) \times (n-2) \times \cdots \times 3 \times 2 \times 1$$

  - aplicar a técnica de acumuladores;
- idéia: colocar a solução para o sub-problema em um laço;
- primeiro passo: implementar o sub-problema:

Fatorial

Problema: Imprimir o valor do fatorial de todos os números entre 1 e n, sendo n fornecido pelo usuário.

- quais as semelhanças deste com os já vistos?
- (sub-problema) como calcula o fatorial de n?

$$fat(n) = n \times (n-1) \times (n-2) \times \cdots \times 3 \times 2 \times 1$$

  - aplicar a técnica de acumuladores;
- idéia: colocar a solução para o sub-problema em um laço;
- primeiro passo: implementar o sub-problema:

Fatorial de n

$$fat(5) = 5 \times 4 \times 3 \times 2 \times 1 = 120$$

```
program fatorial ;
var i , n, fat : integer ;
begin
   read (n) ;
   fat:= 1; (* inicializacao do acumulador *)
   i:= n;
   write ('fat(' ,n,')= ') ;
   while i >= 1 do
   begin
      fat:= fat * i ;
      if i > 1 then
         write ( i ,'x')
      else
         write ( i ,'= ') ;
         i:= i - 1;
   end;
   writeln ( fat ) ;
end.
```

Fatorial de todos até n

```
program fatorial1_n ;
var cont, i, n, fat : integer ;
begin
   read (n);
   cont:= 1;
   while cont <= n do
   begin
      write ('fat(' ,cont ,')= ') ;
      fat:= 1; (* inicializacao do acumulador *)
      i:= cont;
      while i >= 1 do
      begin
         fat:= fat * i ;
         if i > 1 then
           write ( i ,'x')
         else
           write ( i ,'= ') ;
         i:= i - 1;
      end;
      writeln ( fat ) ;
      cont:= cont + 1;
   end;
end.
```

Fatorial de todos até n

```
Saıída do programa com entrada n=7:
fat(1)= 1= 1
fat(2)= 2x1= 2
fat(3)= 3x2x1= 6
fat(4)= 4x3x2x1= 24
fat(5)= 5x4x3x2x1= 120
fat(6)= 6x5x4x3x2x1= 720
fat(7)= 7x6x5x4x3x2x1= 5040
```

Uma novo ângulo

- Em muitos casos, é possível escrever programas mais simples e eficientes olhando para o problema por outro ângulo;
- O programa foi desenvolvido a partir da fórmula abaixo:

$$fat(n) = n \times (n-1) \times (n-2) \times \cdots \times 3 \times 2 \times 1$$

- Porém, outra forma de obter este mesmo resultado é:

$$fat(n) = fat(n-1) \times n$$

```
fat(1)= 1= 1
fat(2)= fat(1)x2= 2
fat(3)= fat(2)x3= 6
fat(4)= fat(3)x4= 24
fat(5)= fat(4)x5= 120
fat(6)= fat(5)x6= 720
fat(7)= fat(6)x7= 5040
```

Uma novo ângulo

- Em muitos casos, é possível escrever programas mais simples e eficientes olhando para o problema por outro ângulo;
- O programa foi desenvolvido a partir da fórmula abaixo:

$$fat(n) = n \times (n-1) \times (n-2) \times \cdots \times 3 \times 2 \times 1$$

- Porém, outra forma de obter este mesmo resultado é:

$$fat(n) = fat(n-1) \times n$$

```
fat(1)= 1= 1
fat(2)= fat(1)x2= 2
fat(3)= fat(2)x3= 6
fat(4)= fat(3)x4= 24
fat(5)= fat(4)x5= 120
fat(6)= fat(5)x6= 720
fat(7)= fat(6)x7= 5040
```

Uma novo ângulo

- Em muitos casos, é possível escrever programas mais simples e eficientes olhando para o problema por outro ângulo;
- O programa foi desenvolvido a partir da fórmula abaixo:

$$fat(n) = n \times (n-1) \times (n-2) \times \cdots \times 3 \times 2 \times 1$$

- Porém, outra forma de obter este mesmo resultado é:

$$fat(n) = fat(n-1) \times n$$

```
fat(1)= 1= 1
fat(2)= fat(1)x2= 2
fat(3)= fat(2)x3= 6
fat(4)= fat(3)x4= 24
fat(5)= fat(4)x5= 120
fat(6)= fat(5)x6= 720
fat(7)= fat(6)x7= 5040
```

Uma novo ângulo

- Este novo ângulo sugere reaproveitar os valores calculados, produzindo o programa abaixo:

```
rogram fatorial1_n_v2 ;
var cont , n, fat : integer ;
begin
   read (n) ;
   cont:= 1;
   fat:= 1; (* inicializacao do acumulador *)
   while cont <= n do
   begin
      fat:= fat * cont ;
      writeln ('fat(' ,cont ,')= ' , fat ) ;
      cont:= cont + 1;
   end;
end.
```

Fibonacci revisado

- Muitas vezes, o problema a ser resolvido é semelhante a um problema já implementado.
- Esta seção apresenta problemas que podem ser resolvidos ao modificar um programa já desenvolvido (neste caso fibonacci):
  - qual é o primeiro número de Fibonacci maior do que um determinado valor;
  - o número áureo;

Fibonacci revisado

- Muitas vezes, o problema a ser resolvido é semelhante a um problema já implementado.
- Esta seção apresenta problemas que podem ser resolvidos ao modificar um programa já desenvolvido (neste caso fibonacci):
  - qual é o primeiro número de Fibonacci maior do que um determinado valor;
  - o número áureo;

Fibonacci revisado

- Muitas vezes, o problema a ser resolvido é semelhante a um problema já implementado.
- Esta seção apresenta problemas que podem ser resolvidos ao modificar um programa já desenvolvido (neste caso fibonacci):
  - qual é o primeiro número de Fibonacci maior do que um determinado valor;
  - o número áureo;

Alterando o critério de parada

Problema: imprimir o primeiro número de Fibonacci maior do que um determinado valor

```
program Fibonacci ;
var ultimo , penultimo , soma, cont , n: integer ;
begin
   read (n) ; (* define quantos numeros serao impressos *)
   ultimo:= 1; (* inicializacao das variaveis principais *)
   penultimo:= 1;
   writeln (penultimo) ; (* imprime os dois primeiros valores *)
   writeln (ultimo) ;
   cont:= 3 ; (* calcula do terceiro em diante *)
   while cont <= n do
   begin
      soma:= penultimo + ultimo ;
      writeln (cont , ' ' ,soma) ;
      penultimo:= ultimo ; (* a ordem destes dois comandos *)
      ultimo:= soma;       (* eh relevante no codigo *)
      cont:= cont + 1;
   end;
end.
```

Alterando o critério de parada

Problema: imprimir o primeiro número de Fibonacci maior do que um determinado valor

```
program Fibonacci_2 ;
const max=1000;
var ultimo , penultimo , soma: integer ;
begin
    ultimo:= 1;
    penultimo:= 1;
    soma:= penultimo + ultimo ;
    while soma <= max do (* termina quando atinge o valor desejado *)
    begin
        penultimo:= ultimo ;
        ultimo:= soma;
        soma:= penultimo + ultimo ;
    end;
    writeln (soma) ;
end.
```

O número áureo

- razão entre dois termos consecutivos converge para um número irracional conhecido como número áureo, de notado pela letra grega φ, e é aproximadamente 1.6180339887499:

$$\frac{1}{1} = 1 \, , \frac{2}{1} = 2 \, , \frac{3}{2} = 1.5 \, , \frac{5}{3} = 1.66 \, ,$$

$$\frac{8}{5} = 1.60 \, , \frac{13}{8} = 1.625 \, , \frac{21}{13} = 1.615 \, \ldots$$

- Convergência do número: os número intermediários obtidos nos cálculos estão cada vez mais próximos uns dos outros, o que nos leva a ter a oportunidade de definirmos, por exemplo, com quantas casas decimais estaremos satisfeitos.

O número áureo

- razão entre dois termos consecutivos converge para um número irracional conhecido como número áureo, de notado pela letra grega φ, e é aproximadamente 1.6180339887499:

$$\frac{1}{1} = 1 \, , \frac{2}{1} = 2 \, , \frac{3}{2} = 1.5 \, , \frac{5}{3} = 1.66 \, ,$$

$$\frac{8}{5} = 1.60 \, , \frac{13}{8} = 1.625 \, , \frac{21}{13} = 1.615 \, \ldots$$

- Convergência do número: os número intermediários obtidos nos cálculos estão cada vez mais próximos uns dos outros, o que nos leva a ter a oportunidade de definirmos, por exemplo, com quantas casas decimais estaremos satisfeitos.

O número áureo

Problema: utilizar a sequencia de fibonacci para imprimir o número áureo com a precisão desejada.

- Precisão desejada?
- Convergência: $|aureo(n) - aureo(n-1)| \leq precisao$
- Pode não convergir, por exemplo precisão 10^{-10000};
- Incluir regra alternativa, por exemplo: se o laço executar um certo número de vezes, digamos, mil, e não chegar no erro estabelecido, então o programa deve encerrar com uma mensagem de erro informando esta situação (exercício).

O número áureo

Problema: utilizar a sequencia de fibonacci para imprimir o número áureo com a precisão desejada.

- Precisão desejada?
- Convergência: $|aureo(n) - aureo(n-1)| \leq precisao$
- Pode não convergir, por exemplo precisão 10^{-10000};
- Incluir regra alternativa, por exemplo: se o laço executar um certo número de vezes, digamos, mil, e não chegar no erro estabelecido, então o programa deve encerrar com uma mensagem de erro informando esta situação (exercício).

O número áureo

Problema: utilizar a sequencia de fibonacci para imprimir o número áureo com a precisão desejada.

- Precisão desejada?
- Convergência: $|aureo(n) - aureo(n-1)| \leq precisao$
- Pode não convergir, por exemplo precisão 10^{-10000};
- Incluir regra alternativa, por exemplo: se o laço executar um certo número de vezes, digamos, mil, e não chegar no erro estabelecido, então o programa deve encerrar com uma mensagem de erro informando esta situação (exercício).

O número áureo

Problema: utilizar a sequencia de fibonacci para imprimir o número áureo com a precisão desejada.

- Precisão desejada?
- Convergência: $|aureo(n) - aureo(n-1)| \leq precisao$
- Pode não convergir, por exemplo precisão 10^{-10000};
- Incluir regra alternativa, por exemplo: se o laço executar um certo número de vezes, digamos, mil, e não chegar no erro estabelecido, então o programa deve encerrar com uma mensagem de erro informando esta situação (exercício).

O número áureo

Problema: utilizar a sequencia de fibonacci para imprimir o número áureo com a precisão desejada

```
program Fibonacci ;
var ultimo , penultimo , soma, cont , n: integer ;
begin
   read (n) ; (* define quantos numeros serao impressos *)
   ultimo:= 1; (* inicializacao das variaveis principais *)
   penultimo:= 1;
   writeln (penultimo) ; (* imprime os dois primeiros valores *)
   writeln (ultimo) ;
   cont:= 3 ; (* calcula do terceiro em diante *)
   while cont <= n do
   begin
      soma:= penultimo + ultimo ;
      writeln (cont , ' ' ,soma) ;
      penultimo:= ultimo ; (* a ordem destes dois comandos *)
      ultimo:= soma;       (* eh relevante no codigo *)
      cont:= cont + 1;
   end;
end.
```

O número áureo

```
program numero aureo ;
const PRECISAO=0.00000000000001;
var ultimo , penultimo , soma: integer ;
   naureo , naureo_anterior : real ;
begin
   ultimo:= 1;     (* inicializacao das variaveis principais *)
   penultimo:= 1;
   naureo_anterior:= -1; (* para funcionar o primeiro teste *)
   naureo:= 1;
   writeln (naureo:15:14) ;
   (* calcula do terceiro em diante *)
   while abs(naureo - naureo_anterior ) >= PRECISAO do
   begin
      soma:= penultimo + ultimo ;
      naureo_anterior:= naureo ;
      naureo:= soma/ultimo ;
      writeln (naureo:15:14) ;
      penultimo:= ultimo ;
      ultimo:= soma;
   end;
end.
```

Palídromos

- Palíndromos são lidos da direita para a esquerda da mesma maneira que da esquerda para a direita. Por exemplo o número 12321 é palíndromo, enquanto que 123 não é;
- Método: separar os dígitos do número, reconstruir o número ao contrário, comparar com o número original.
- generalização do problema de se inverter um número de três dígitos. Dicas:

$$d_n d_{n-1} \ldots d_2 d_1 d_0 \Rightarrow d_0 d_1 d_2 \ldots d_{n-1} d_n$$

$$(12345)\ \textbf{div}\ 10 = 1234 \quad (1)$$

$$(12345)\ \textbf{mod}\ 10 = 5 \quad (2)$$

$$(1234) * 10 + 5 = 12345 \quad (3)$$

Palídromos

- Palíndromos são lidos da direita para a esquerda da mesma maneira que da esquerda para a direita. Por exemplo o número 12321 é palíndromo, enquanto que 123 não é;
- Método: separar os dígitos do número, reconstruir o número ao contrário, comparar com o número original.
- generalização do problema de se inverter um número de três dígitos. Dicas:

$$d_n d_{n-1} \ldots d_2 d_1 d_0 \Rightarrow d_0 d_1 d_2 \ldots d_{n-1} d_n$$

$$(12345)\ \textbf{div}\ 10 = 1234 \quad (1)$$

$$(12345)\ \textbf{mod}\ 10 = 5 \quad (2)$$

$$(1234) * 10 + 5 = 12345 \quad (3)$$

Palídromos

- Palíndromos são lidos da direita para a esquerda da mesma maneira que da esquerda para a direita. Por exemplo o número 12321 é palíndromo, enquanto que 123 não é;
- Método: separar os dígitos do número, reconstruir o número ao contrário, comparar com o número original.
- generalização do problema de se inverter um número de três dígitos. Dicas:

$$d_n d_{n-1} \ldots d_2 d_1 d_0 \Rightarrow d_0 d_1 d_2 \ldots d_{n-1} d_n$$

$$(12345)\ \textbf{div}\ 10 = 1234 \quad (1)$$

$$(12345)\ \textbf{mod}\ 10 = 5 \quad (2)$$

$$(1234) * 10 + 5 = 12345 \quad (3)$$

Palídromos

Problema: Imprimir todos os números palíndromos entre 1 e 1000.

```
program todospalindromos ;
const max=1000;
var i , invertido , n: integer ;
begin
   i:= 1;
   while i <= max do (* laco que controla os numeros entre 1 e max *)
   begin
      invertido:= 0; (* inicializa aculumador *)
      n:= i ;
      while n <> 0 do
      begin
         invertido:= invertido*10 + n mod 10;
         n:= n div 10;
      end;
      (* imprime se for palindromo , senao nao faz nada *)
      if invertido = i then
      writeln ( i ) ;
      i:= i + 1;
   end;
end.
```

Palídromos

Problema: Gerar todos os números palíndromos entre 1 e 1000.

- Uma outra abordagem ao problemabaseia-se no fato de o maior palíndromo ocupar 3 dígitos:
  1. gerar todos os palíndromos de um dígito (muito fácil);
  2. gerar todos os palíndromos de dois dígitos (muito fácil);
  3. gerar todos os palíndromos de três dígitos. Dica:

$$d[0-9]d$$

Palídromos

Problema: Gerar todos os números palíndromos entre 1 e 1000.

- Uma outra abordagem ao problemabaseia-se no fato de o maior palíndromo ocupar 3 dígitos:
  1. gerar todos os palíndromos de um dígito (muito fácil);
  2. gerar todos os palíndromos de dois dígitos (muito fácil);
  3. gerar todos os palíndromos de três dígitos. Dica:

$$d[0-9]d$$

Palídromos

Problema: Gerar todos os números palíndromos entre 1 e 1000.

- Uma outra abordagem ao problemabaseia-se no fato de o maior palíndromo ocupar 3 dígitos:
  1. gerar todos os palíndromos de um dígito (muito fácil);
  2. gerar todos os palíndromos de dois dígitos (muito fácil);
  3. gerar todos os palíndromos de três dígitos. Dica:

$$d[0-9]d$$

Palídromos

Problema: Gerar todos os números palíndromos entre 1 e 1000.

- Uma outra abordagem ao problemabaseia-se no fato de o maior palíndromo ocupar 3 dígitos:
  1. gerar todos os palíndromos de um dígito (muito fácil);
  2. gerar todos os palíndromos de dois dígitos (muito fácil);
  3. gerar todos os palíndromos de três dígitos. Dica:

$$d[0-9]d$$

Palídromos

Problema: Gerar todos os números palíndromos entre 1 e 1000.

```
program gerandopalindromos ;
var i , j , pal : integer ;
begin
    ...                         (* gerando todos de um digito *)
    pal:= 11;                   (* gerando todos de 2 digitos *)
    i:= 2;
    while i <= 9 do
    begin
        writeln (pal) ;
        pal:= i * 11;
        i:= i + 1;
    end;                        (* ------------------------------ *)
    i:= 1;                      (* gerando todos os de 3 digitos *)
    while i <= 9 do
    begin
        j:= 0;
        while j <= 9 do
        begin
            pal:= i *100 + j*10 + i ;
            writeln (pal) ;
            j:= j + 1;
        end;
        i:= i + 1;
    end;
end.
```

Fim

- este material está no livro no capítulo 7, seções de 7.5 até 7.7