CI1055: Algoritmos e Estruturas de Dados I

Profs. Drs. Marcos Castilho, Bruno Müller Jr e Carmem Hara

Departamento de Informática/UFPR

23 de julho de 2020

Resumo

Técnicas elementares que envolvem a combinação de comandos repetitivos e condicionais.

Técnicas Elementares

- Definir a priori e depois corrigir
- Lembrar de mais de uma informação
- Processar parte dos dados de entrada
- Processar dados de entrada de formas distintas
- Múltiplos acumuladores

Definir a priori e depois corrigir

Determinar o menor número de uma sequência de inteiros terminada em zero.

Entrada: 7 3 12 45 2 5 0

num	**menor**	
7	7	Define como valor de **menor** a priori
3	3	Corrige
12	3	
45	3	
2	2	Corrige novamente
5	2	

Definir a priori e depois corrigir

Determinar o menor número de uma sequência de inteiros terminada em zero.

Entrada: 7 3 12 45 2 5 0

Como definir e corrigir o valor de **menor**?

num	**menor**	
7	7	Inicialização
3	3	Corrige SE o valor de **num** for menor que **menor**
12	3	
45	3	
2	2	Corrige SE o valor de **num** for menor que **menor**
5	2	

Definir a priori e depois corrigir

Determinar o menor número de uma sequência de inteiros terminada em zero.

Entrada: 7 3 12 45 2 5 0

Como definir e corrigir o valor de **menor**?

num	**menor**	
7	7	read(num); menor:= num
3	3	if num $<$ menor then menor:= num;
12	3	if num $<$ menor then menor:= num;
45	3	if num $<$ menor then menor:= num;
2	2	if num $<$ menor then menor:= num;
5	2	if num $<$ menor then menor:= num;

Solução 1

```
1  program menorValor;
2  var
3      n, menor : integer;
4  begin
5      read( n );                  // inicializacao
6      menor:= n;
7      while n <> 0 do
8      begin
9          if n < menor then       // padrao repetitivo
10             menor:= n;
11         read( n );              // incremento
12     end;
13     writeln( menor );
14 end.
```

Solução 2

Mas a sequência pode ser vazia.

```
1  program menorValor;
2  var
3      n, menor : integer;
4
5  begin
6      read( n );
7      if n <> 0 then
8      begin
9          menor:= n;
10         while n <> 0 do
11         begin
12             if n < menor then
13                 menor:= n;
14             read( n );
15         end;
16         writeln( menor );
17     end;
18 end.
```

Lembrar de mais de uma informação

Fibonacci: fib(0) = 0
fib(1)=1
fib(n)=fib(n-1)+fib(n-2)

Gera a sequência 0,1,1,2,3,5,8,13,...
Escrever um programa que leia $n \geq 1$ e escreva os valores de $fib(0)$ a $fib(n)$.

i	**fib**			
0	0			
1	1			
2	1	fib(1)	+	fib(0)
3	2	fib(2)	+	fib(1)
4	3	fib(3)	+	fib(2)
5	5	fib(4)	+	fib(3)
6	8	fib(5)	+	fib(4)

"Lembrar" é guardar em uma variável

Para calcular fib(n-1)+fib(n-2) é preciso LEMBRAR destas informções.

i	**fib**	**ult**		**penult**			
0	0						
1	1						
2	1	1	+	0	fib(1)	+	fib(0)
3	2	1	+	1	fib(2)	+	fib(1)
4	3	2	+	1	fib(3)	+	fib(2)
5	5	3	+	2	fib(4)	+	fib(3)
6	8	5	+	3	fib(5)	+	fib(4)

"Lembrar" é guardar em uma variável

i	fib	ult		penult	
0	0				
1	1				
2	1	1	+	0	fib:= ult+penult;
3	2	1	+	1	fib:= ult+penult;
4	3	2	+	1	fib:= ult+penult;
5	5	3	+	2	fib:= ult+penult;
6	8	5	+	3	fib:= ult+penult;

Para funcionar, os valores de **ult** e **penult** tem que mudar a cada repetição.

Padrão repetitivo

```
1 fib:= ult + penult;
2 writeln( fib );
3 penult:= ult;
4 ult:= fib;
```

Completando o programa

Inicialização

```
1  penult:= 0;
2  ult:= 1;
3  writeln( 0 );
4  writeln( 1 );
5  while ... do
6  begin
7      fib:= ult + penult;
8      writeln( fib );
9      penult:= ult;
10     ult:= fib;
11 end;
```

Completando o programa

Controle e Teste de parada

```
1  read( n );
2  i:= 2;
3  penult:= 0;
4  ult:= 1;
5  writeln( 0 );
6  writeln( 1 );
7  while i <= 2 do
8  begin
9      fib:= ult + penult;
10     writeln( fib );
11     penult:= ult;
12     ult:= fib;
13     i:= i+1;
14 end;
```

Programa Fibonacci

```
1  program fibonacci;
2  var
3     fib, ult, penult, n, i: integer;
4  begin
5     read( n );
6     i:= 2;
7     penult:= 0;
8     ult:= 1;
9     writeln( 'fib(0)= 0' );
10    writeln( 'fib(1)= 1' );
11    while i <= n do
12    begin
13       fib:= ult+penult;
14       writeln( 'fib(', i, ')= ', fib );
15       penult:= ult;
16       ult:= fib;
17       i:= i+1;
18    end;
19 end.
```

Processar parte dos dados de entrada

Ler 8 números inteiros e escrever na tela apenas os positivos.
Ignorar valores negativos e zero.

Exemplo de entrada: 10 -5 -3 12 33 87 -7 -10

cont	**num**	**ação**
1	10	escrever num
2	-5	
3	-3	
4	12	escrever num
5	33	escrever num
6	87	escrever num
7	-7	
8	-10	

Montagem da solução

cont	num	
1	10	if num > 0 then writeln(num);
2	-5	if num > 0 then writeln(num);
3	-3	if num > 0 then writeln(num);
4	12	if num > 0 then writeln(num);
5	33	if num > 0 then writeln(num);
6	87	if num > 0 then writeln(num);
7	-7	if num > 0 then writeln(num);
8	-10	if num > 0 then writeln(num);

Padrão repetitivo

```
1  read( num );
2  if num > 0 then
3     writeln( num );
```

Programa completo

```
1  program escrevePositivo;
2  var
3      cont, num : integer;
4  begin
5      cont:= 1;
6      while cont <= 8 do
7      begin
8          read( num );
9          if num > 0 then
10             writeln( num );
11         cont:= cont+1;
12     end;
13 end.
```

Processar parte de um modo e outra parte de outro

Ler 8 números inteiros e escrever na tela apenas os positivos.
Imprimir o quadrado dos que não são, incluindo o zero.

Exemplo de entrada: 10 -5 -3 12 33 87 -7 -10

cont	**num**	**ação**
1	10	escrever num
2	-5	escrever num*num
3	-3	escrever num*num
4	12	escrever num
5	33	escrever num
6	87	escrever num
7	-7	escrever num*num
8	-10	escrever num*num

Montagem da solução

cont	num	
1	10	if num $>$ 0 then writeln(num) else writeln(num*num);
2	-5	if num $>$ 0 then writeln(num) else writeln(num*num);
3	-3	if num $>$ 0 then writeln(num) else writeln(num*num);
4	12	if num $>$ 0 then writeln(num) else writeln(num*num);
5	33	if num $>$ 0 then writeln(num) else writeln(num*num);
6	87	if num $>$ 0 then writeln(num) else writeln(num*num);
7	-7	if num $>$ 0 then writeln(num) else writeln(num*num);
8	-10	if num $>$ 0 then writeln(num) else writeln(num*num);

Padrão repetitivo

```
1  read( num );
2  if num > 0 then
3    writeln( num )
4  else
5    writeln( num*num );
```

Programa completo

```
1  program escreveNumQuadrado;
2  var
3      cont, num : integer;
4  begin
5      cont:= 1;
6      while cont <= 8 do
7      begin
8          read( num );
9          if num > 0 then
10             writeln( num )
11         else
12             writeln( num * num );
13         cont:= cont+1;
14     end;
15 end.
```

Múltiplos Acumuladores

Ler uma sequência de inteiros que correspondem à idade dos pacientes. A sequência termina com o número -1, que não deve ser processado. Escrever na tela a quantidade de pacientes separados por faixa etária:
- bebes: 0-2 anos
- criancas: 3-12 anos
- adolescentes: 13-19 anos
- adultos: 20-59 anos
- idosos: 60 anos ou mais

Exemplo de entrada: 10 25 1 3 12 33 87 7 10 45 -1

Exemplo de saída:

```
bebes: 1
criancas: 5
adolescentes: 0
adultos: 3
idosos: 1
```

Um Acumulador por Faixa

idade	bebe	crianca	adolescente	adulto	idoso
10		+1			
25				+1	
1	+1				
3		+1			
12		+1			
33				+1	
87					+1
7		+1			
10		+1			
45				+1	

Padrão Repetitivo - Versão 1

```
1  if idade <= 2 then
2      bebe:= bebe + 1;
3  if (idade > 2) and (idade <= 12) then
4      crianca:= crianca + 1;
5  if (idade > 12) and (idade <= 19) then
6      adolescente:= adolescente + 1;
7  if (idade > 19) and (idade <= 59) then
8      adulto:= adulto + 1;
9  if idade > 59 then
10     idoso:= idoso + 1;
```

Isso é muito ineficiente. Por que?

Padrão Repetitivo - Versão 2

```
1  if idade <= 2 then
2      bebe:= bebe + 1
3  else if idade <= 12 then
4           crianca:= crianca + 1
5       else if idade <= 19 then
6                adolescente:= adolescente + 1
7            else if idade <= 59 then
8                     adulto:= adulto + 1
9                 else
10                    idoso:= idoso + 1;
```

- a Linha 3 só é avaliada SE a condição na Linha 1 for falsa.
- a Linha 5 só é avaliada SE as condições das Linhas 1 e 3 forem falsas.
- a Linha 7 só é avaliada SE as condições das Linhas 1, 3 e 5 forem falsas.
- a Linha 10 só é executada se TODAS as condições das linhas anteriores forem falsas.

Arrumando a indentação

Vamos deixar claro que são alternativas **mutamente excludentes** usando uma indentação diferenciada.

```
1  if idade <= 2 then
2      bebe:= bebe + 1
3  else if idade <= 12 then
4      crianca:= crianca + 1
5  else if idade <= 19 then
6      adolescente:= adolescente + 1
7  else if idade <= 59 then
8      adulto:= adulto + 1
9  else idoso:= idoso + 1;
```

Como tem um único comando em cada alternativa:

```
1  if idade <= 2 then bebe:= bebe + 1
2  else if idade <= 12 then crianca:= crianca + 1
3  else if idade <= 19 then adolescente:= adolescente + 1
4  else if idade <= 59 then adulto:= adulto + 1
5  else idoso:= idoso + 1;
```

O Programa Completo

```
1  program histograma;
2  var idade, bebe, crianca, adolescente, adulto, idoso : integer;
3  begin
4      bebe:= 0; crianca:= 0; adolescente:= 0; adulto:= 0; idoso:= 0;
5      read( idade );
6      while idade <> -1 do
7      begin
8          if idade <= 2 then bebe:= bebe + 1
9          else if idade <= 12 then crianca:= crianca + 1
10         else if idade <= 19 then adolescente:= adolescente + 1
11         else if idade <= 59 then adulto:= adulto + 1
12         else idoso:= idoso + 1;
13         read( idade );
14     end;
15     writeln( 'bebe: ', bebe );
16     writeln( 'crianca: ', crianca );
17     writeln( 'adolescente: ', adolescente );
18     writeln( 'adulto: ', adulto );
19     writeln( 'idoso: ', idoso );
20 end.
```

Exercícios

- Fazer os exercícios 1 a 9 da seção 6.10 do livro [1]

[1] `http://www.inf.ufpr.br/cursos/ci055/livro_alg1.pdf`

Fim do tópico

- o conteúdo desta aula está no livro no capítulo 6, seções 6.5-6.9

Licença

- Slides feitos em LaTeX usando beamer
- Licença

  Creative Commons Atribuição-Uso Não-Comercial-Vedada a Criação de Obras Derivadas 2.5 Brasil License.`http://creativecommons.org/licenses/by-nc-nd/2.5/br/`

  Creative Commons Atribuição-Uso Não-Comercial-Vedada a Criação de Obras Derivadas 2.5 Brasil License.`http://creativecommons.org/licenses/by-nc-nd/2.5/br/`