MAC 2166 – Introdução à Computação

POLI - PRIMEIRO SEMESTRE DE 2007

Material Didático

Prof. Ronaldo Fumio Hashimoto

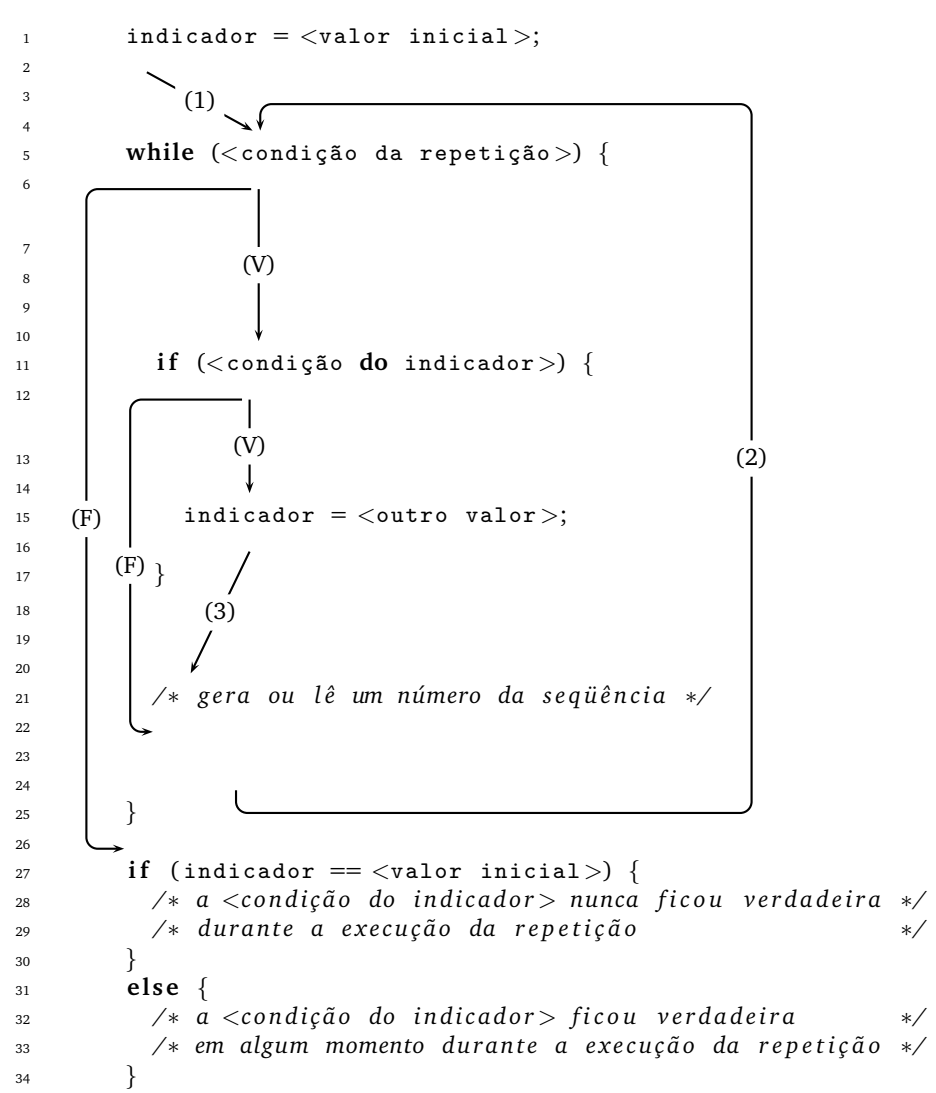
INDICADOR DE PASSAGEM

Introdução

Nesta aula, vamos falar de um tópico bastante importante para o curso de MAC2166.

Conceito de Indicador de Passagem

Considere o seguinte padrão de programação.



Na linha 5, temos uma repetição que trata de gerar ou ler pelo teclado uma seqüência de números. Observo mais uma vez que os exercícios que estamos lidando sempre há uma seqüência de números. Antes da repetição, na linha 1, existe uma inicialização de uma variável `indicador` com um certo valor inicial. Dentro da repetição, na linha 11, existe um comando de seleção simples (`if`) que testa uma propriedade da seqüência de números (por exemplo, seqüência crescente, seqüência com todos números positivos, seqüência com todos números pares, etc...). Se a condição `<condição do indicador>` ficar verdadeira em algum momento durante a execução da repetição, então o valor da variável `indicador` recebe outro valor diferente do valor inicial. No final da repetição, testa-se o conteúdo da variável `indicador`. Se conteúdo desta variável é o `<valor inicial>`, então a condição `<condição do indicador>` nunca foi satisfeita durante a execução da repetição. Agora, se o conteúdo é `<outro valor>`, então, em algum momento, durante a execução da repetição, a condição `<condição do indicador>` foi satisfeita.

Vamos chamar este padrão de programação de **padrão indicador de passagem**.

Exemplo

Considere o seguinte programa que lê uma seqüência de dez números inteiros:

```
1   # include <stdio.h>
2
3   int main () {
4     int pos, i, x;
5
6     pos = 0;
7     i = 0;
8     while (i<10) {
9       printf ("Entre com x: ");
10      scanf ("%d", &x);
11      if (x > 0) {
12        pos = 1;
13      }
14      i = i + 1;
15    }
16    if (pos == 0)
17      printf ("Todos elems menores ou iguais a zero\n");
18    else
19      printf ("Pelo menos um elem. maior do que zero\n");
20
21    return 0;
22  }
```

A variável `pos` é um **indicador de passagem**.

Na linha 8, temos uma repetição que trata de ler pelo teclado uma seqüência de dez números inteiros. Antes da repetição, na linha 6, existe uma inicialização da variável `pos` com valor inicial igual a um. Dentro da repetição, na linha 11, existe um comando de seleção simples (`if`) que testa se o número lido é maior do que zero. Se esta condição ficar verdadeira em algum momento durante a execução da repetição, então o valor da variável `pos` recebe outro valor diferente do valor inicial, que no caso é o valor um. No final da repetição, testa-se o conteúdo da variável `pos`. Se conteúdo da variável `pos` é o zero, então a condição `x>0` nunca foi satisfeita durante a execução da repetição, indicando que todos os elementos da seqüência são menores ou iguais a zero. Agora, se o conteúdo é um, então, em algum momento durante a execução da repetição, a condição `x>0` foi satisfeita, indicando que pelo menos um elemento da seqüência é maior do que zero.

É importante notar que o indicador de passagem tenta capturar uma propriedade da seqüência. No exemplo anterior, o indicador de passagem tenta capturar se a seqüência contém algum número positivo.

Além disso, note que a propriedade da seqüência que o indicador de passagem tenta capturar sempre coloca uma questão cuja resposta é “sim” (verdadeira) ou “não” (falsa). No exemplo anterior, o indicador tenta responder a questão: a seqüência tem algum número positivo?

Uso de Constantes

Na aula “Detalhes da Linguagem C” comentamos que é possível definir constantes. Nesta aula, vamos falar um pouco da sua utilidade.

Para responder a questão que o indicador de passagem tenta responder, poderíamos definir duas constantes: `TRUE` e `FALSE`. O indicador de passagem terminaria com valor `TRUE` se a reposta da questão for “sim” e com valor `FALSE` caso contrário.

Assim, o programa anterior ficaria:

```c
# include <stdio.h>

# define TRUE 1
# define FALSE 0

int main () {
  int pos, i, x;

  pos = FALSE;
  i = 0;
  while (i<10) {
    printf ("Entre com x: ");
    scanf ("%d", &x);
    if (x > 0) {
      pos = TRUE;
    }
    i = i + 1;
  }
  if (pos == FALSE)
    printf ("Todos elems menores ou iguais a zero\n");
  else
    printf ("Pelo menos um elem. maior do que zero\n");

  return 0;
}
```

Exemplo

Considere o seguinte problema:

Dado `n>1`, verificar se `n` tem dois dígitos adjacentes iguais. Exemplos: (1) `n = 21212 => Não` e (2) `n = 212212 => Sim`.

Observe:

- `n = 212123`.
- Seqüência Numérica deste exercício: os dígitos de `n`.
- Neste exemplo, `2, 1, 2, 1, 2, 3`.
- Isto significa que dado `n>1`, a seqüência é gerada.
- Usar a propriedade de divisão e resto por `10`.
- `n/10` $\Rightarrow$ quociente inteiro de `n` por `10`, ou seja, o número `n` sem o último dígito.
- Então `212123/10 = 21212`
- `n%10` $\Rightarrow$ o resto da divisão de `n` por `10`, ou seja, o último dígito de `n`.
- Então `212123%10 = 3`.
- Usando estas propriedades, vamos gerar a seqüência dos dígitos de `n` de trás para frente. Mas isto não tem importância, uma vez que a propriedade de adjacência não depende se a seqüência está de trás para frente e vice-versa. Veja, logo depois do código, por que a seqüência é inversa.
- Descascar o número `n` até que ele vire zero.
- Neste exercício, queremos verificar uma propriedade da seqüência gerada: se ela contém dois números adjacentes iguais.

- Para verificar esta propriedade, vamos usar um indicador de passagem de nome `adjacente` que começaria com valor `FALSE`. Se em algum momento, o programa encontrar dois números adjacentes iguais, então esta variável recebe valor `TRUE`.

Usando o padrão indicador de passagem, então temos o seguinte programa:

```
1   # include <stdio.h>
2
3   # define TRUE 1
4   # define FALSE 0
5
6   int main () {
7     int n;
8     int posterior, anterior;
9     int adjacente;
10
11    printf ("Entre com n > 0: ");
12    scanf ("%d", &n);
13
14    adjacente = FALSE;
15    posterior = -1;
16
17    while (n != 0) {
18      anterior  = posterior;
19      posterior = n % 10;
20
21      if (anterior == posterior) {
22        adjacente = TRUE;
23        n = 0;
24      }
25
26      n = n / 10;
27    }
28
29    if (adjacente == FALSE)
30      printf ("NAO\n");
31    else
32      printf ("SIM\n");
33
34    return 0;
35  }
```

Neste esquema de repetição, na linha `19`, a variável `posterior` recebe o último dígito de `n` e, na linha `26`, `n` fica sendo o número `n` sem o último dígito. Assim, para `n = 213`, teríamos a seguinte tabela de valores para `posterior` e `n`:

posterior	n
?	123
3	12
2	1
1	0

Assim, neste exercício, seqüência de dígitos de `n` é gerada de trás para frente. No exemplo acima, a seqüência para `n = 123` é então $3, 2, 1$.

Outro Exemplo

Considere o seguinte problema:

Dado um inteiro `n>0`, e uma seqüência de `n` inteiros, calcular a sua soma e verificar se a seqüência é estritamente crescente.

Neste exercício, além de calcular a soma, queremos verificar uma propriedade da seqüência lida: se ela é estritamente crescente ou não. Para verificar esta propriedade, vamos usar um indicador de passagem de nome `crescente` que começaria com valor `TRUE`. Se em algum momento a seqüência deixar de ser crescente, então esta variável recebe valor `FALSE`.

Usando o padrão indicador de passagem, então temos o seguinte programa:

```
1   # include <stdio.h>
2
3   # define TRUE 1
4   # define FALSE 0
5
6   int main () {
7     int cont; /* contador dos elementos da sequencia */
8     int n; /* numero de elementos da sequencia */
9     int soma;
10    int num; /* cada elemento da sequencia */
11    int ant; /* elemento anterior ao num */
12    int crescente;
13
14    printf ("Entre com n>0: ");
15    scanf ("%d", &n);
16
17    printf ("Entre com um num. inteiro da seq.: ");
18    scanf ("%d", &num);
19
20    crescente = TRUE;
21    soma = num;
22
23    cont = 2;
24    while (cont <= n) {
25
26      ant = num;
27
28      printf ("Entre com um num. inteiro da seq.: ");
29      scanf ("%d", &num);
30
31      if (ant >= num)
32        crescente = FALSE;
33
34      soma = soma + num;
35
36      cont = cont + 1;
37    }
38
39    printf ("soma = %d\n", soma);
40
41    if (crescente == TRUE)
42      printf ("Sequencia Estritamente Crescente\n");
43    else
44      printf ("Sequencia Nao Estritamente Crescente\n");
45
46    return 0;
47  }
```

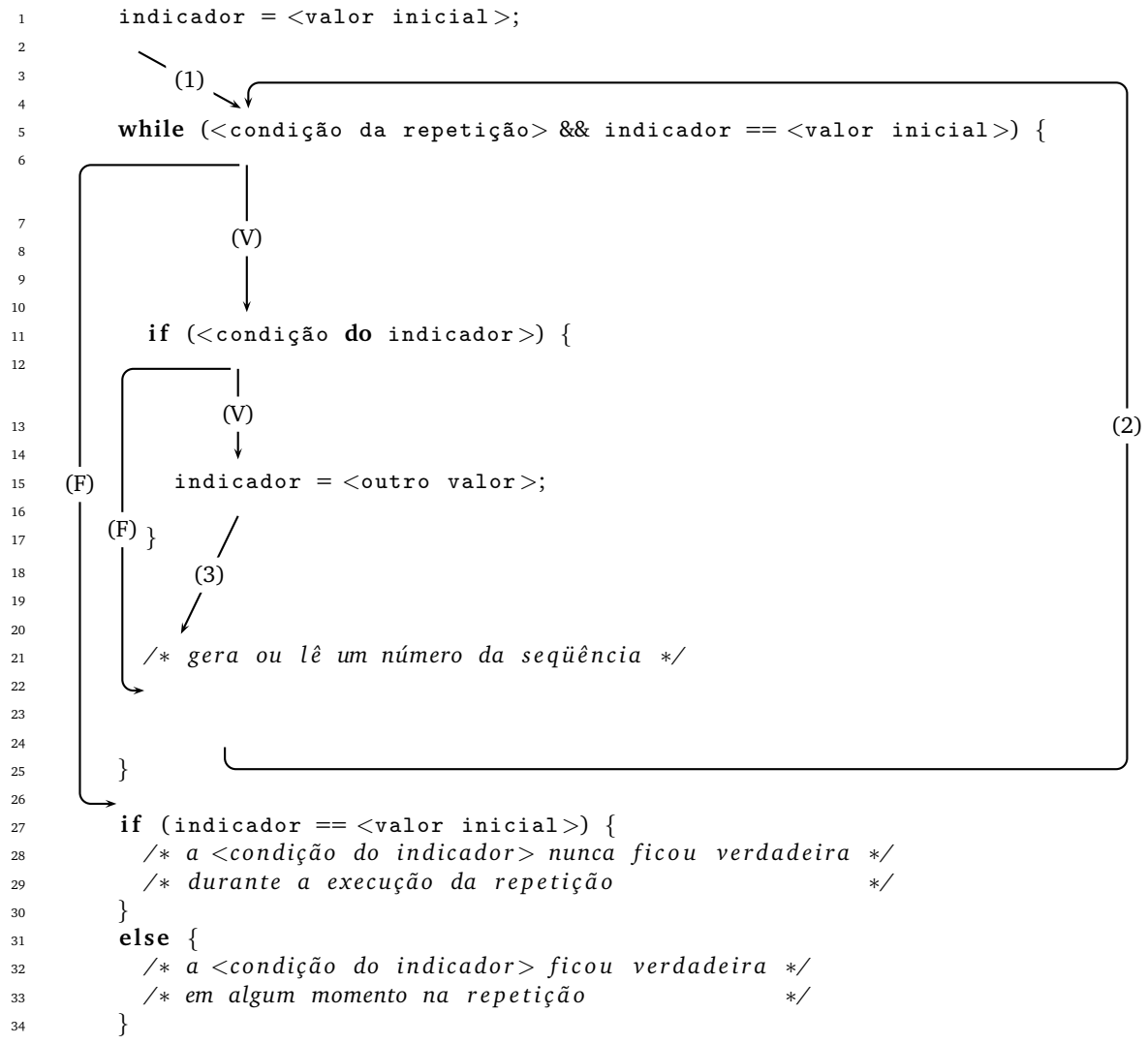
Repetição Interrompida Condicionada

Existem situações em que precisamos verificar uma propriedade de uma seqüência. Para isto, fazemos uso de um indicador de passagem.

Agora, pode acontecer que, no momento em que o indicador de passagem recebe outro valor (diferente do valor inicial), não é mais necessário testar os outros números da seqüência. Neste caso, podemos interromper a repetição.

Por exemplo, considere o problema dos dígitos adjacentes. Considere `n=12345678990`. Lembre-se que a seqüência gerada é composta pelos dígitos de `n` de trás para frente. A seqüência gerada é então: $0, 9, 9, 8, 7, 6, 5, 4, 3, 2, 1$. Quando o programa encontrar os dígitos adjacentes `9` e `9`, não é mais necessário verificar o restante da seqüência.

Nestes casos, podemos utilizar o seguinte padrão de programação que usa o operador lógico `&&` e o indicador de passagem dentro da condição da repetição



A primeira condição `<condição da repetição>` garante a geração e/ou leitura da seqüência. A segunda condição `indicador == <valor inicial>` garante que quando o indicador de passagem trocar de valor, a repetição é interrompida no momento da verificação da condição `<condição da repetição> && indicador == <valor inicial>`, uma vez que estamos usando o operador lógico `&&` e a condição `indicador == <valor inicial>` é falsa.

Exercício que Usa este Conceito

1. Dado um inteiro `p>1`, verificar se `p` é primo.
   Use indicador de passagem e o padrão repetição interrompida condicionada.

Dúvidas

Dúvidas deste material pode ser enviadas para o “Fórum para assuntos específicos da turma WEB”.