

MAC 2166 – Introdução à Computação

POLI - PRIMEIRO SEMESTRE DE 2007

Material Didático

Prof. Ronaldo Fumio Hashimoto

REPETIÇÕES ENCAIXADAS**Introdução**

O conceito de repetições encaixadas nada mais é que repetição dentro de uma outra repetição. Por exemplo:

```

1      while (<condição 1>) {
2          while (<condição 2>) {
3              while (<condição 3>) {
4                  }
5              }
6          }

```

Até o momento, nossos programas foram simples, e os problemas não exigiam muitos comandos dentro de uma repetição. Mas os comandos dentro de uma repetição podem se tornar bastante complexos. Nessa aula, vamos ver como alguns problemas exigem a criação de repetição dentro de repetição (ou repetições encaixadas). A forma mais simples de apresentar esse recurso é através de um problema.

Exemplo de um Problema que usa Repetições Encaixadas

Problema: Dados dois naturais $n > 0$ e $m > 0$, determinar entre todos os pares de números inteiros (x, y) tais que $0 \leq x \leq n$ e $0 \leq y \leq m$, um par para o qual o valor da expressão $x \times y - x^2 + y$ seja máximo e calcular também este máximo. Em caso de empate, imprima somente um valor.

Exemplo: para $n=2$ e $m=3$, a saída do programa deve ser $(1, 3)$ com valor máximo igual a 5.

Solução:

- Tomando o exemplo $n = 2$ e $m = 3$;
- Gerar todos os pontos (x, y) tais que $0 \leq x \leq n$ e $0 \leq y \leq m$.
- Para cada x fixo, temos que gerar os valores para y de 0 a $m=3$. Assim, temos:

$x=0$

$(0, 0) (0, 1) (0, 2) (0, 3) \Rightarrow$ Para $x=0$, um laço para gerar os valores de y de 0 a 3.

$x=1$

$(1, 0) (1, 1) (1, 2) (1, 3) \Rightarrow$ Para $x=1$, um laço para gerar os valores de y de 0 a 3.

$x=2$

$(2, 0) (2, 1) (2, 2) (2, 3) \Rightarrow$ Para $x=2$, um laço para gerar os valores de y de 0 a 3.

- Assim, para cada x fixo, temos uma repetição para gerar os valores para y de 0 a 3.

```

1      /* para um valor fixo de x */
2      y = 0;
3      while (y <= m) {
4          printf ("%d, %d\n", x, y);
5          y = y + 1;
6      }

```

- Agora é necessário ter uma repetição para gerar os valores de x de 0 a 2.

```

1      x = 0;
2      while (x <= n) {
3          /* gera um valor para x */
4          x = x + 1;
5      }

```

- Note que a repetição que gera a seqüência para y deve estar dentro da repetição que gera a seqüência para x

- Assim, juntando os códigos, temos

```

1      x = 0;
2      while (x <= n) {
3          /* gera um valor para x */
4          /* para um valor fixo de x */
5          y = 0;
6          while (y <= m) {
7              printf ("%d, %d\n", x, y);
8              y = y + 1;
9          }
10         x = x + 1;
11     }

```

- Agora, temos que detectar um par (x, y) para o qual o valor da expressão $x \times y - x^2 + y$ seja máximo e calcular também este máximo.

- Para cada par (x, y) gerado pelo código anterior, calcular o valor da expressão $x \times y - x^2 + y$.

$x=0$

$(0, 0) \Rightarrow 0$ $(0, 1) \Rightarrow 1$ $(0, 2) \Rightarrow 2$ $(0, 3) \Rightarrow 3$

$x=1$

$(1, 0) \Rightarrow -1$ $(1, 1) \Rightarrow 1$ $(1, 2) \Rightarrow 3$ $(1, 3) \Rightarrow 5$

$x=2$

$(2, 0) \Rightarrow -4$ $(2, 1) \Rightarrow -1$ $(2, 2) \Rightarrow 2$ $(2, 3) \Rightarrow 5$

- Assim, basta ter uma variável \max que, para cada par gerado, guarda o valor máximo até o presente momento. Assim, teríamos:

$x=0$

$(0, 0) \Rightarrow 0, \max=0$ $(0, 1) \Rightarrow 1, \max=1$ $(0, 2) \Rightarrow 2, \max=2$ $(0, 3) \Rightarrow 3, \max=3$

$x=1$

$(1, 0) \Rightarrow -1, \max=3$ $(1, 1) \Rightarrow 1, \max=3$ $(1, 2) \Rightarrow 3, \max=3$ $(1, 3) \Rightarrow 5, \max=5$

$x=2$

$(2, 0) \Rightarrow -4, \max=5$ $(2, 1) \Rightarrow -1, \max=5$ $(2, 2) \Rightarrow 2, \max=5$ $(2, 3) \Rightarrow 5, \max=5$

- Para cada par gerado, calcula-se o valor da expressão $v = x \times y - x^2 + y$ e testa se $v > \max$. Em caso afirmativo, o valor de \max deve ser atualizado para v fazendo $\max = v$;

- Temos então o seguinte código:

```

1      x = 0;
2      while (x <= n) {
3          /* gera um valor para x */
4          y = 0;
5          while (y <= m) {
6              v = x*y - x*x + y;
7              if (v > max)
8                  max = v;
9              y = y + 1;
10         }
11         x = x + 1;
12     }
13     printf ("O valor maximo = %d\n", max);

```

- O código anterior consegue determinar o valor máximo, mas não consegue detectar qual par (x,y) tem este valor.
- Para detectar qual par (x,y) tem o valor máximo, temos que guardá-lo toda vez que a variável max é atualizada:

```

1      x = 0;
2      while (x <= n) {
3          /* gera um valor para x */
4          y = 0;
5          while (y <= m) {
6              v = x*y - x*x + y;
7              if (v > max) {
8                  max = v;
9                  x_max = x;
10                 y_max = y;
11             }
12             y = y + 1;
13         }
14         x = x + 1;
15     }
16     printf ("O valor maximo = %d em x = %d e y = %d\n", max, x_max, y_max);

```

- Falta ainda resolver um problema: quais devem ser os valores iniciais para max, x_max e y_max?
- O problema de iniciarmos max com 0 é que se os valores calculados de v forem todos negativos, a variável max nunca será atualizada.
- O problema de iniciarmos max com um número negativo (digamos -1000) é que se os valores calculados de v forem todos menores que -1000, a variável max nunca será atualizada.
- Uma boa idéia é inicializarmos max com um valor que v assume. Poderia ser qualquer valor da seqüência 0, 1, 2, 3, -1, 1, 3, 5, -4, -1, 2, 5.
- Assim, podemos escolher um par (x,y) que é gerado. Por exemplo, o par (0,0) e calcular o valor da expressão $v = x \times y - x^2 + y$ para este par. Neste caso, $v = 0$ e colocamos para (x_max, y_max) o par (0, 0).
- Assim, a solução final seria:

```

1      # include <stdio.h>
2
3      int main () {
4          int x, y, n, m, v, x_max, y_max;
5
6          printf ("Entre com n>0: ", &n);
7          scanf ("%d", &n);
8
9          printf ("Entre com m>0: ", &m);
10         scanf ("%d", &m);
11
12         x = 0; max = x_max = y_max = 0;
13         while (x <= n) {
14             /* gera um valor para x */
15             y = 0;
16             while (y <= m) {
17                 v = x*y - x*x + y;
18                 if (v > max) {
19                     max = v;
20                     x_max = x;
21                     y_max = y;
22                 }
23                 y = y + 1;
24             }
25             x = x + 1;
26         }
27         printf ("O valor maximo = %d em x = %d e y = %d\n", max, x_max, y_max);
28         return 0;
29     }

```

Exercício

1. Dados $n > 0$ números inteiros positivos, calcular a soma dos que são primos.
 - (a) Uma repetição com contador para ler n números pelo teclado.
 - (b) Para cada número lido, testar se ele é primo (usando uma outra repetição com indicador de passagem). Em caso afirmativo, acumular em uma soma.

Dúvidas

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