Aula 18: Laços aninhados e desvios

Introdução a Programação

Túlio Toffolo & Puca Huachi
http://www.toffolo.com.br

BCC201 – 2019/2
Departamento de Computação – UFOP

Aula Anterior

- Comandos de Repetição (Partes 2 e 3)
- Exercícios no *moodle*

Aula de Hoje

1. Exercícios
2. Laços Aninhados
3. Comando `continue`
4. Comando `break`
5. Exercício
6. Próxima aula

Aula de Hoje

1. Exercícios
2. Laços Aninhados
3. Comando `continue`
4. Comando `break`
5. Exercício
6. Próxima aula

Exercício da aula passada

Exercício 1

A Sequência de **Fibonacci** é uma sequência de números inteiros iniciando por 0, seguido por 1 e depois pela soma dos dois anteriores: (0, 1, 1, 2, 3, 5, 8, ...).

Escreva um programa (utilizando o comando de repetição `for`) que imprime os n primeiros números da sequência de Fibonacci. O usuário deve informar o valor de n.

Exemplo:

```
1 Qual o valor de n? 7
2
3 0 1 1 2 3 5 8
```

Exercícios

Série de Fibonacci

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, 6765, 10946, ...

$$F(n) = \begin{cases} 0 & \text{se } n = 0, \\ 1 & \text{se } n = 1, \\ F(n-1) + F(n-2) & \text{caso contrário.} \end{cases}$$

```c
#include <stdio.h>

int main()
{
    int n, termo1 = 0, termo2 = 1, aux;
    printf("Qual o valor de n?: ");
    scanf("%d", &n);
    printf("\n");

    if (n >= 1)
        printf("%d ", termo1);

    if (n >= 2)
        printf("%d ", termo2);

    for (int i = 3; i <= n; i++) {
        aux = termo1 + termo2;
        printf("%d ", aux);
        termo1 = termo2;
        termo2 = aux;
    }

    printf("\n");
    return 0;
}
```

Aula de Hoje

1. Exercícios
2. Laços Aninhados
3. Comando `continue`
4. Comando `break`
5. Exercício
6. Próxima aula

Laços Aninhados

```
1
22
333
4444
55555
666666
7777777
88888888
999999999
```

Repetição 1: temos nove repetições de linhas ($1 \leq n \leq 9$).

Repetição 2: temos, em cada linha, a repetição de n caracteres que identificam a própria linha, sendo $1 \leq n \leq 9$. Assim, temos $n = 1$ na linha 1, $n = 2$ na linha 2, e assim sucessivamente, até a linha 9.

- Para obter a saída acima, realizamos a **Repetição 2** dentro da **Repetição 1**.

Laços Aninhados

```c
#include <stdio.h>

int main()
{
    // Repetição 1
    for (int linha = 1; linha <= 9; linha++) {

        // Repetição 2
        for (int coluna = 1; coluna <= linha; coluna++) {
            printf("%d", linha);
        }

        printf("\n");
    }

    return 0;
}
```

Laços Aninhados

```
1  Contador externo (linha): 1
2
3          Contador interno (coluna): 1
4          Contador interno (coluna): 2
5          Contador interno (coluna): 3
6          Contador interno (coluna): 4
7
8  Contador externo (linha): 2
9
10         Contador interno (coluna): 1
11         Contador interno (coluna): 2
12         Contador interno (coluna): 3
13         Contador interno (coluna): 4
14
15 Contador externo (linha): 3
16
17         Contador interno (coluna): 1
18         Contador interno (coluna): 2
19         Contador interno (coluna): 3
20         Contador interno (coluna): 4
```

Laços Aninhados

```c
#include <stdio.h>

int main()
{
    // Repetição variando a <linha>
    for (int linha = 1; linha <= 3; linha++) {
        printf("Contador externo (linha): %d\n\n", linha);

        // Repetição variando a <coluna>
        for (int coluna = 1; coluna <= 4; coluna++) {
            printf("\t\tContador interno (coluna): %d\n", coluna);
        }

        printf("\n");
    }

    return 0;
}
```

Exemplo 1

Faça um programa que imprime a tabuada de x até y (valores de x e y devem ser digitados pelo usuário).

```
Digite os valores para x e y: 5 15

Tabuada de multiplicação!

   |   5   6   7   8   9  10  11  12  13  14  15
-----------------------------------------------
 5 |  25  30  35  40  45  50  55  60  65  70  75
 6 |  30  36  42  48  54  60  66  72  78  84  90
 7 |  35  42  49  56  63  70  77  84  91  98 105
 8 |  40  48  56  64  72  80  88  96 104 112 120
 9 |  45  54  63  72  81  90  99 108 117 126 135
10 |  50  60  70  80  90 100 110 120 130 140 150
11 |  55  66  77  88  99 110 121 132 143 154 165
12 |  60  72  84  96 108 120 132 144 156 168 180
13 |  65  78  91 104 117 130 143 156 169 182 195
14 |  70  84  98 112 126 140 154 168 182 196 210
15 |  75  90 105 120 135 150 165 180 195 210 225
```

```c
int main()
{
    int x, y;
    printf("Digite os valores para x e y: ");
    scanf("%d %d", &x, &y);

    // imprimindo o cabecalho
    printf("\nTabuada de multiplicação!\n\n");
    printf("   | ");
    for (int j = x; j <= y; j++)
        printf("%3d ", j);
    printf("\n----");
    for (int j = x; j <= y; j++)
        printf("----");
    printf("\n");

    // calculando (e imprimindo) a tabuada
    for (int i = x; i <= y; i++) {
        printf("%2d | ", i);
        for (int j = x; j <= y; j++)
            printf("%3d ", i*j);
        printf("\n");
    }
    return 0;
}
```

Exemplo 2

Bart Simpson está aprendendo a jogar xadrez, mas tem dificuldade em saber para qual direção ele pode mover sua **Torre**.

Sabemos que um tabuleiro de xadrez é composto por 8 linhas e 8 colunas, e que a **Torre** se move ortogonalmente, ou seja, pelas linhas (horizontais) e pelas colunas (verticais).

- Escreva um programa que solicite ao *Bart* o número da linha e da coluna que indicam a posição de sua **Torre**. O programa deve imprimir quais são os possíveis movimentos da **Torre**.
- Utilize `"-"` para indicar uma casa para a qual a Torre não pode ser movida e `"x"` para indicar uma casa para a qual ela pode ser movida.

Exemplo 2

Exemplo de saída:

```
Movimentos de uma Torre no xadrez!
Digite a linha  em que a Torre se encontra: 6
Digite a coluna em que a Torre se encontra: 3

Movimentos possíveis:

      1  2  3  4  5  6  7  8
    ------------------------
 1 |  -  -  x  -  -  -  -  -
 2 |  -  -  x  -  -  -  -  -
 3 |  -  -  x  -  -  -  -  -
 4 |  -  -  x  -  -  -  -  -
 5 |  -  -  x  -  -  -  -  -
 6 |  x  x  x  x  x  x  x  x
 7 |  -  -  x  -  -  -  -  -
 8 |  -  -  x  -  -  -  -  -
```

```c
int main()
{
    int linha, coluna;

    printf("Movimentos de uma Torre no xadrez!\n");
    printf("Digite a linha  em que a Torre se encontra: ");
    scanf("%d", &linha);
    printf("Digite a coluna em que a Torre se encontra: ");
    scanf("%d", &coluna);

    // Imprime o cabeçalho da tabela antes do loop
    printf("\nMovimentos possíveis:\n\n");
    printf("      1  2  3  4  5  6  7  8  \n");
    printf("    ------------------------\n");

    // Imprime a tabela
    for (int l = 1; l <= 8; l++) {
        printf(" %d | ", l);
        for (int c = 1; c <= 8; c++) {
            if (l == linha || c == coluna) {
                printf(" x ");
            }
            else {
                printf(" - ");
            }
        }
        printf("\n");
    }
    return 0;
}
```

Exemplo 3

O valor de π pode ser calculado pelo seguinte algoritmo:

Passo 1) Calcula-se a série com n termos:

$$S = \frac{1}{1^3} - \frac{1}{3^3} + \frac{1}{5^3} - \frac{1}{7^3} + \cdots$$

Passo 2) O valor aproximado de π é dado pela expressão:

$$\pi = \sqrt[3]{S \times 32}$$

Faça um programa para calcular e imprimir o valor aproximado de π (use 20 casas decimais). O programa deve ler o número de termos. Exemplo:

```
1 Digite o número de termos: 10
2 Valor aproximado de pi: 3.14152608792950616134
```

```c
#include <stdio.h>
#include <math.h>

int main()
{
    // lendo o nro de termos (n)
    int n;
    printf("Digite o número de termos: ");
    scanf("%d", &n);

    // calculando o valor de s
    double s = 0, sinal = +1;
    for (int i = 0; i < n; i++) {
        s += sinal / pow(1.0 + 2*i, 3);
        sinal *= -1;
    }

    // calculando e imprimindo o valor de pi
    double pi = cbrt(s * 32);
    printf("Valor aproximado de pi: %.20lf\n", pi);

    return 0;
}
```

Exemplo 4

Crie um programa que calcula o valor da expressão a seguir. Considere que os valores de n e m serão fornecidos pelo usuário.

$$x = \sum_{i=1}^{n} \sum_{j=1}^{m} (i + j)$$

Exemplo de execução:

```
1 Digite os valores de n e m: 10 5
2
3 x = 425.000000
```

Exemplo 4

```c
#include <stdio.h>

int main()
{
    // lendo os valores de n e m
    int n, m;
    printf("Digite os valores de n e m: ");
    scanf("%d %d", &n, &m);

    // calculando o valor de x
    double x = 0;
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= m; j++) {
            x += i + j;
        }
    }

    // imprimindo o valor de x na saída
    printf("\nx = %lf\n", x);
    return 0;
}
```

Aula de Hoje

1. Exercícios
2. Laços Aninhados
3. Comando `continue`
4. Comando `break`
5. Exercício
6. Próxima aula

Desvio em laço

Em diversos momentos queremos **alterar o fluxo** ou mesmo encerrar a execução de um laço de repetição.

Um dos comandos utilizados para isso é o `continue`

- Este comando permite alterar o fluxo do laço, fazendo-o retornar ao início.
- É particularmente útil para evitar `if` aninhados em alguns casos.

Comando `continue`

Exemplo de uso em laço `while`:

```
1  while (<condição>) {
2      <comando_1>;
3      ...
4      continue;
5      ...
6      <comando_n>;
7  }
```

Comando `continue`

Exemplo de uso em laço `do-while`:

```
1  do {
2      <comando_1>;
3      ...
4      continue;
5      ...
6      <comando_n>;
7  } while (<condição>);
```

Comando `continue`

Exemplo de uso em laço `for`:

```
1  for (<inicialização>; <condição>; <incremento/decremento>) {
2      <comando_1>;
3      ...
4      continue;
5      ...
6      <comando_n>;
7  }
```

Exemplo

Faça um aplicativo em C que some todos os números, de 1 até 100, exceto os múltiplos de 5.

Precisamos do comando `continue` para criar tal programa?

- Definitivamente **não**!
- Mas o `continue` é uma alternativa válida que (em alguns casos) simplifica o código.

Exemplo

Usando o `continue`:

```c
#include <stdio.h>

int main()
{
    int soma = 0;

    for (int cont = 1; cont <= 100; cont++) {
        if (cont % 5 == 0)
            continue;
        soma += cont;
    }
    printf("Soma = %d\n", soma);
    return 0;
}
```

Exemplo

Alternativa sem o `continue`:

```c
#include <stdio.h>

int main()
{
    int soma = 0;

    for (int cont = 1; cont <= 100; cont++) {
        if (cont % 5 != 0) {
            soma += cont;
        }
    }
    printf("Soma = %d\n", soma);
    return 0;
}
```

Aula de Hoje

1. Exercícios
2. Laços Aninhados
3. Comando `continue`
4. Comando `break`
5. Exercício
6. Próxima aula

Desvio em laço

Em diversos momentos queremos alterar o fluxo ou mesmo **encerrar** a execução de um laço de repetição.

Um dos comandos utilizados para isso é o `break`

- Este comando permite encerrar o laço imediatamente.
- Assim como o continue, é particularmente útil para evitar uma quantidade excessiva de `if` aninhados.

Comando `break`

Exemplo de uso em laço `while`:

```
1  while (<condição>) {
2      <comando_1>;
3      ...
4      break;
5      ...
6      <comando_n>;
7  }
```

Comando `break`

Exemplo de uso em laço `do-while`:

```
1  do {
2      <comando_1>;
3      ...
4      break;
5      ...
6      <comando_n>;
7  } while (<condição>);
```

Comando `break`

Exemplo de uso em laço `for`:

```
1  for (<inicialização>; <condição>; <incremento/decremento>) {
2      <comando_1>;
3      ...
4      break;
5      ...
6      <comando_n>;
7  }
```

Exemplo

Faça um programa que imprime o primeiro número, entre 1 e 1 milhão, que é divisível por 11, 13 e 17.

Precisamos do comando `break` para criar tal programa?

- Definitivamente **não**!
- Mas o `break` neste caso simplifica o código.

Exemplo

```c
#include <stdio.h>

int main()
{
    for (int cont = 1; cont <= 1000000; cont++) {
        if (cont % 11 == 0 && cont % 13 == 0 && cont % 17 == 0) {
            printf("O número é %d\n", cont);
            break;
        }
    }
    return 0;
}
```

Aula de Hoje

1. Exercícios
2. Laços Aninhados
3. Comando `continue`
4. Comando `break`
5. Exercício
6. Próxima aula

Exercício

Exercício 1

Apresente um programa em C que imprime uma tabela contendo a tabuada de multiplicação de 1 a 20 **ignorando os números pares**.

Exemplo:

```
          1   3     ...     19
-----------------------------
  1 |     1   3             19
  3 |     3   9     ...     57
  5 |     5  15     ...     95
                      ...
 19 |    19  57     ...    361
```

Dica: use `"%3d "` para ficar bonito! :)

Aula de Hoje

1. Exercícios
2. Laços Aninhados
3. Comando `continue`
4. Comando `break`
5. Exercício
6. Próxima aula

Próxima Aula

Aula teórica:

- Vetores – estruturas de dados homogêneas unidimensionais

Perguntas?