Computação eletrônica: Estruturas de repetição

Gurvan Huiban
ghuiban@cin.ufpe.br

24 de abril de 2014

Plano de aula

Estruturas de repetição

- Repetir a execução de um conjunto de instruções por um número de vezes:
  - Fixo;
  - Variável.
- Na linguagem C: `for`, `while`, `do-while`.

Exemplos

- Somar os *n* primeiros números ímpares;
- Repetir um menu até o usuário digitar `x`.

Exemplo: $\sum_{i=1}^{n} i$

Soma de 1 até 10

Mostrar a soma dos n primeiros inteiros, para n de 1 até 10.

De que precisamos?

Realizar um grande número de operações

Soma de 1 até 10: Solução força bruta

```
main()
{
  printf("Valor da soma ate 1: %d\n", 1);
  printf("Valor da soma ate 2: %d\n", 1+2);
  printf("Valor da soma ate 3: %d\n", 1+2+3);
  printf("Valor da soma ate 4: %d\n", 1+2+3+4);
  printf("Valor da soma ate 5: %d\n", 1+2+3+4+5);
  printf("Valor da soma ate 6: %d\n", 1+2+3+4+5+6);
  printf("Valor da soma ate 7: %d\n", 1+2+3+4+5+6+7);
  printf("Valor da soma ate 8: %d\n", 1+2+3+4+5+6+7+8);
  printf("Valor da soma ate 9: %d\n", 1+2+3+4+5+6+7+8+9);
  printf("Valor da soma ate 10: %d\n",1+2+3+4+5+6+7+8+9+10);
}
```

Problemas

- Realizamos varias vezes as mesmas somas (1+2+3...)
- Código muito extenso
- Código difícil “copiar-colar”

Soma de 1 até 10: Variável acumulando a soma

```
main()
{
  int soma = 0;
  soma = soma + 1;
  printf("Valor da soma ate 1: %d\n", soma);
  soma = soma + 2;
  printf("Valor da soma ate 2: %d\n", soma);
  soma = soma + 3;
  printf("Valor da soma ate 3: %d\n", soma);
  ...
  soma = soma + 8;
  printf("Valor da soma ate 8: %d\n", soma);
  soma = soma + 9;
  printf("Valor da soma ate 9: %d\n", soma);
  soma = soma + 10;
  printf("Valor da soma ate 10: %d\n", soma);
}
```

Problemas

- Código muito extenso
- Código difícil “copiar-colar”

Soma de 1 até 10: Código “copiável”

```c
int main(void)
{
  int soma = 0;
  int i = 1;
  soma = soma + i;
  printf("Valor da soma ate %d: %d\n", i, soma);
  i = i+1;
  soma = soma + i;
  printf("Valor da soma ate %d: %d\n", i, soma);
  i = i+1;
  soma = soma + i;
  printf("Valor da soma ate %d: %d\n", i, soma);
  i = i+1;
  soma = soma + i;
  printf("Valor da soma ate %d: %d\n", i, soma);
  i = i+1;
  soma = soma + i;
  printf("Valor da soma ate %d: %d\n", i, soma);
  i = i+1;
  soma = soma + i;
  printf("Valor da soma ate %d: %d\n", i, soma);
  i = i+1;
  ...
}
```

Problemas

- Código muito extenso

Soma de 1 até 10: Com estrutura for

Programa: Mostrar a soma dos *n* primeiros inteiros, para *n* de 1 até 10

```
int main(void)
{
  int soma = 0;
  int i;

  for (i = 1; i <= 10; i = i+1)
  {
    soma = soma + i;
    printf("Valor da soma ate %d: %d\n", i, soma);
  }
  return 0;
}
```

Sintaxe da estrutura de repetição for

Sintaxe

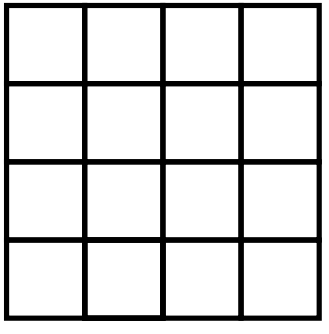
```
for (inicializacoes; condicoes; incrementos)
{
  sequencia de comandos
}
```

- Inicializações: valores iniciais das variáveis;
- Condições: condições para continuar a execução do laço (expressão lógica);
- Incrementos: incrementação das variáveis;
- Sequência de comandos: comandos a serem repetidos.

Soma de 1 até 10: Passo a passo (início)

```
int soma = 0;
int i;
for (i = 1; i <= 10; i = i+1)
{
  soma = soma + i;
  printf("Valor da soma ate %d: %d\n", i, soma);
}
printf("Fim do programa");
```

Tela

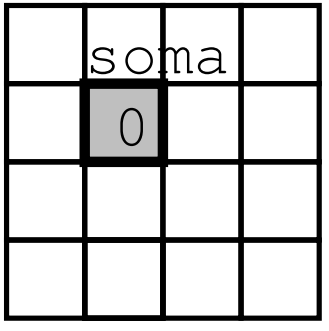


Memória

Soma de 1 até 10: Passo a passo (início)

```
int soma = 0;
int i;
for (i = 1; i <= 10; i = i+1)
{
  soma = soma + i;
  printf("Valor da soma ate %d: %d\n", i, soma);
}
printf("Fim do programa");
```

Tela

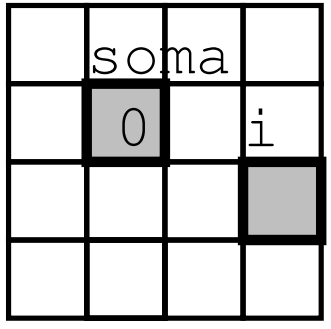


Memória

Soma de 1 até 10: Passo a passo (início)

```
int soma = 0;
int i;
for (i = 1; i <= 10; i = i+1)
{
  soma = soma + i;
  printf("Valor da soma ate %d: %d\n", i, soma);
}
printf("Fim do programa");
```

Tela

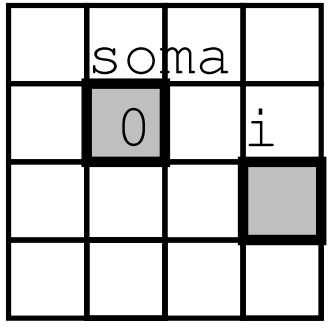


Memória

Soma de 1 até 10: Passo a passo (início)

```
int soma = 0;
int i;
for (i = 1; i <= 10; i = i+1)
{
  soma = soma + i;
  printf("Valor da soma ate %d: %d\n", i, soma);
}
printf("Fim do programa");
```

Tela

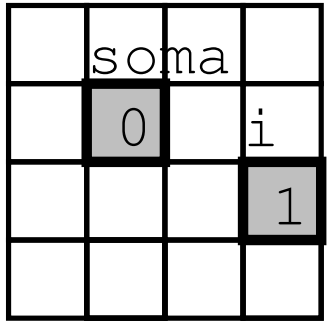


Memória

Soma de 1 até 10: Passo a passo (início)

```
int soma = 0;
int i;
for (i = 1; i <= 10; i = i+1) // Execucao da inicializacao
{
  soma = soma + i;
  printf("Valor da soma ate %d: %d\n", i, soma);
}
printf("Fim do programa");
```

Tela

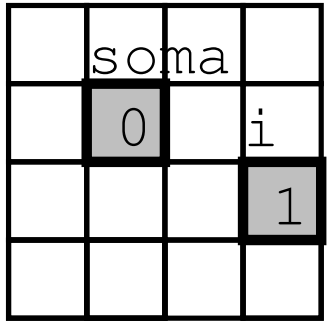


Memória

Soma de 1 até 10: Passo a passo (início)

```
int soma = 0;
int i;
for (i = 1; i <= 10; i = i+1)  // Verdadeiro => Execucao dos comandos
{
  soma = soma + i;
  printf("Valor da soma ate %d: %d\n", i, soma);
}
printf("Fim do programa");
```

Tela

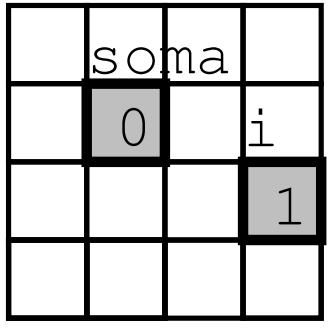


Memória

Soma de 1 até 10: Passo a passo (início)

```
int soma = 0;
int i;
for (i = 1; i <= 10; i = i+1)
{
  soma = soma + i;
  printf("Valor da soma ate %d: %d\n", i, soma);
}
printf("Fim do programa");
```

Tela

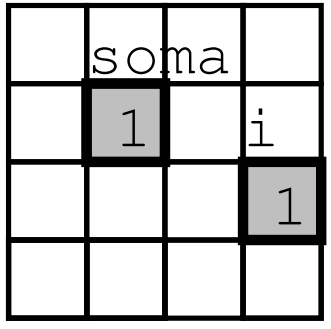


Memória

Soma de 1 até 10: Passo a passo (início)

```
int soma = 0;
int i;
for (i = 1; i <= 10; i = i+1)
{
  soma = soma + i;
  printf("Valor da soma ate %d: %d\n", i, soma);
}
printf("Fim do programa");
```

Tela



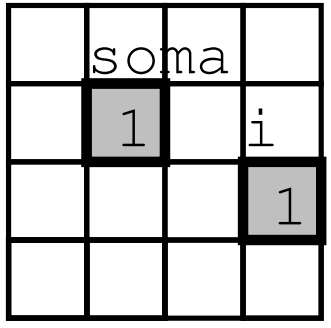
Memória

Soma de 1 até 10: Passo a passo (início)

```
int soma = 0;
int i;
for (i = 1; i <= 10; i = i+1)
{
  soma = soma + i;
  printf("Valor da soma ate %d: %d\n", i, soma);
}
printf("Fim do programa");
```

Tela

```
Valor da soma ate 1: 1
```



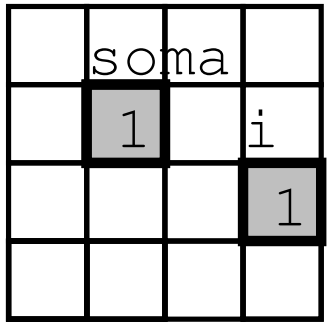
Memória

Soma de 1 até 10: Passo a passo (início)

```
int soma = 0;
int i;
for (i = 1; i <= 10; i = i+1)
{
  soma = soma + i;
  printf("Valor da soma ate %d: %d\n", i, soma);
} // Fim dos comandos associados ao for. Voltando no for
printf("Fim do programa");
```

Tela

```
Valor da soma ate 1: 1
```



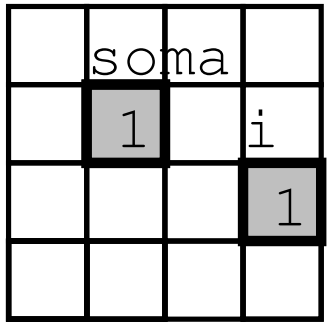
Memória

Soma de 1 até 10: Passo a passo (início)

```
int soma = 0;
int i;
for (i = 1; i <= 10; i = i+1)
{
  soma = soma + i;
  printf("Valor da soma ate %d: %d\n", i, soma);
}
printf("Fim do programa");
```

Tela

```
Valor da soma ate 1: 1
```



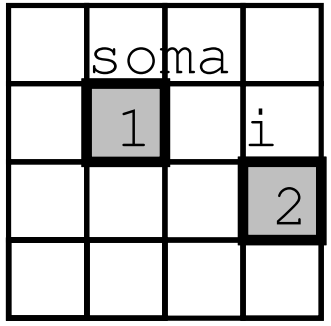
Memória

Soma de 1 até 10: Passo a passo (início)

```
int soma = 0;
int i;
for (i = 1; i <= 10; i = i+1)  // Execucao do incremento
{
  soma = soma + i;
  printf("Valor da soma ate %d: %d\n", i, soma);
}
printf("Fim do programa");
```

Tela

```
Valor da soma ate 1: 1
```



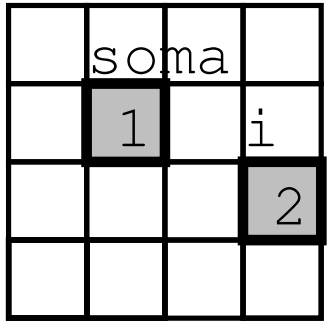
Memória

Soma de 1 até 10: Passo a passo (início)

```
int soma = 0;
int i;
for (i = 1; i <= 10; i = i+1)  // Verdadeiro => Execucao dos comandos
{
  soma = soma + i;
  printf("Valor da soma ate %d: %d\n", i, soma);
}
printf("Fim do programa");
```

Tela

```
Valor da soma ate 1: 1
```



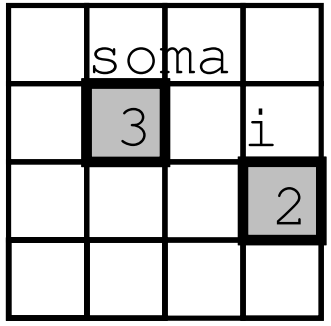
Memória

Soma de 1 até 10: Passo a passo (início)

```
int soma = 0;
int i;
for (i = 1; i <= 10; i = i+1)
{
  soma = soma + i;
  printf("Valor da soma ate %d: %d\n", i, soma);
} // Fim dos comandos associados ao for. Voltando no for
printf("Fim do programa");
```

Tela

```
Valor da soma ate 1: 1
Valor da soma ate 2: 3
```



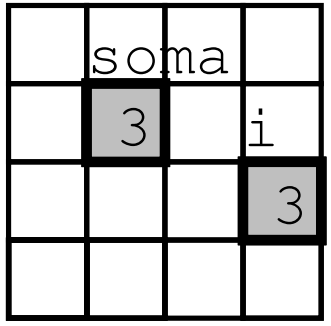
Memória

Soma de 1 até 10: Passo a passo (início)

```
int soma = 0;
int i;
for (i = 1; i <= 10; i = i+1)  // Execucao do incremento
{
  soma = soma + i;
  printf("Valor da soma ate %d: %d\n", i, soma);
}
printf("Fim do programa");
```

Tela

```
Valor da soma ate 1: 1
Valor da soma ate 2: 3
```



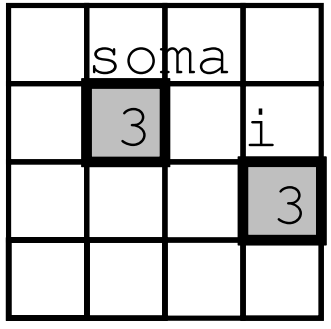
Memória

Soma de 1 até 10: Passo a passo (início)

```
int soma = 0;
int i;
for (i = 1; i <= 10; i = i+1)  // Verdadeiro => Execucao dos comandos
{
  soma = soma + i;
  printf("Valor da soma ate %d: %d\n", i, soma);
}
printf("Fim do programa");
```

Tela

```
Valor da soma ate 1: 1
Valor da soma ate 2: 3
```



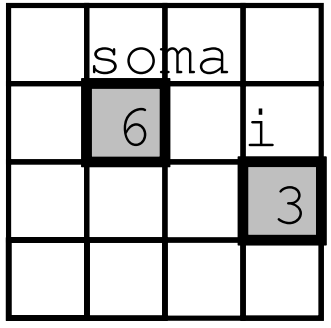
Memória

Soma de 1 até 10: Passo a passo (início)

```
int soma = 0;
int i;
for (i = 1; i <= 10; i = i+1)
{
  soma = soma + i;
  printf("Valor da soma ate %d: %d\n", i, soma);
} // Fim dos comandos associados ao for. Voltando no for
printf("Fim do programa");
```

Tela

```
Valor da soma ate 1: 1
Valor da soma ate 2: 3
Valor da soma ate 3: 6
```



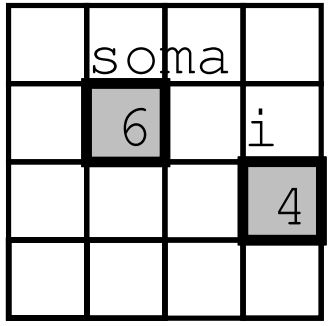
Memória

Soma de 1 até 10: Passo a passo (início)

```
int soma = 0;
int i;
for (i = 1; i <= 10; i = i+1)    // Etc.
{
  soma = soma + i;
  printf("Valor da soma ate %d: %d\n", i, soma);
}
printf("Fim do programa");
```

Tela

```
Valor da soma ate 1: 1
Valor da soma ate 2: 3
Valor da soma ate 3: 6
```



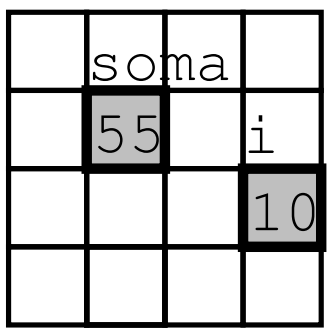
Memória

Soma de 1 até 10: Passo a passo (fim)

```
int soma = 0;
int i;
for (i = 1; i <= 10; i = i+1)
{
  soma = soma + i;
  printf("Valor da soma ate %d: %d\n", i, soma);
}
printf("Fim do programa");
```

Tela

```
Valor da soma ate 1: 1
Valor da soma ate 2: 3
Valor da soma ate 3: 6
Valor da soma ate 4: 10
Valor da soma ate 5: 15
Valor da soma ate 6: 21
Valor da soma ate 7: 28
Valor da soma ate 8: 36
Valor da soma ate 9: 45
```



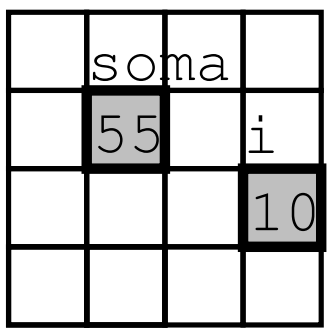
Memória

Soma de 1 até 10: Passo a passo (fim)

```
int soma = 0;
int i;
for (i = 1; i <= 10; i = i+1)
{
  soma = soma + i;
  printf("Valor da soma ate %d: %d\n", i, soma);
}
printf("Fim do programa");
```

Tela

```
Valor da soma ate 1: 1
Valor da soma ate 2: 3
Valor da soma ate 3: 6
Valor da soma ate 4: 10
Valor da soma ate 5: 15
Valor da soma ate 6: 21
Valor da soma ate 7: 28
Valor da soma ate 8: 36
Valor da soma ate 9: 45
Valor da soma ate 10: 55
```



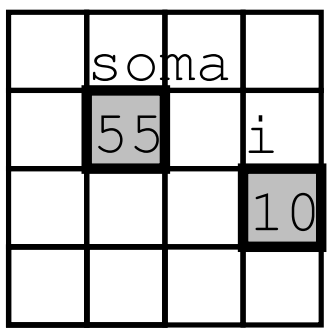
Memória

Soma de 1 até 10: Passo a passo (fim)

```
int soma = 0;
int i;
for (i = 1; i <= 10; i = i+1)
{
  soma = soma + i;
  printf("Valor da soma ate %d: %d\n", i, soma);
} // Fim dos comandos associados ao for. Voltando no for
printf("Fim do programa");
```

Tela

```
Valor da soma ate 1: 1
Valor da soma ate 2: 3
Valor da soma ate 3: 6
Valor da soma ate 4: 10
Valor da soma ate 5: 15
Valor da soma ate 6: 21
Valor da soma ate 7: 28
Valor da soma ate 8: 36
Valor da soma ate 9: 45
Valor da soma ate 10: 55
```



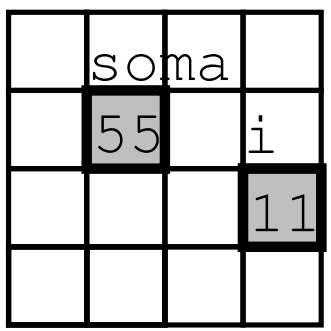
Memória

Soma de 1 até 10: Passo a passo (fim)

```
int soma = 0;
int i;
for (i = 1; i <= 10; i = i+1)  // Execucao do incremento
{
  soma = soma + i;
  printf("Valor da soma ate %d: %d\n", i, soma);
}
printf("Fim do programa");
```

Tela

```
Valor da soma ate 1: 1
Valor da soma ate 2: 3
Valor da soma ate 3: 6
Valor da soma ate 4: 10
Valor da soma ate 5: 15
Valor da soma ate 6: 21
Valor da soma ate 7: 28
Valor da soma ate 8: 36
Valor da soma ate 9: 45
Valor da soma ate 10: 55
```



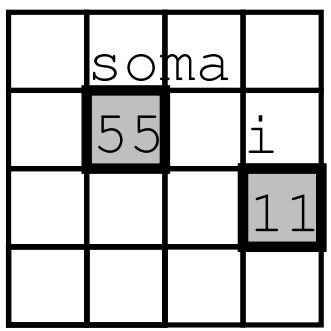
Memória

Soma de 1 até 10: Passo a passo (fim)

```
int soma = 0;
int i;
for (i = 1; i <= 10; i = i+1)  // False => Pular os comandos e continuar
{
  soma = soma + i;
  printf("Valor da soma ate %d: %d\n", i, soma);
}
printf("Fim do programa");
```

Tela

```
Valor da soma ate 1: 1
Valor da soma ate 2: 3
Valor da soma ate 3: 6
Valor da soma ate 4: 10
Valor da soma ate 5: 15
Valor da soma ate 6: 21
Valor da soma ate 7: 28
Valor da soma ate 8: 36
Valor da soma ate 9: 45
Valor da soma ate 10: 55
```



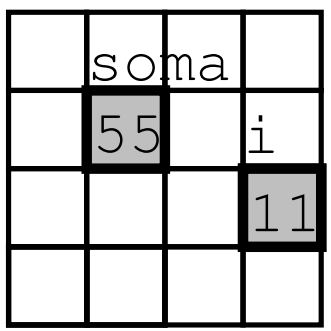
Memória

Soma de 1 até 10: Passo a passo (fim)

```
int soma = 0;
int i;
for (i = 1; i <= 10; i = i+1)
{
  soma = soma + i;
  printf("Valor da soma ate %d: %d\n", i, soma);
}
printf("Fim do programa");
```

Tela

```
Valor da soma ate 1: 1
Valor da soma ate 2: 3
Valor da soma ate 3: 6
Valor da soma ate 4: 10
Valor da soma ate 5: 15
Valor da soma ate 6: 21
Valor da soma ate 7: 28
Valor da soma ate 8: 36
Valor da soma ate 9: 45
Valor da soma ate 10: 55
Fim do programa
```



Memória

for: Passos de execução

```
for (inicializacoes; condicoes; incrementos)
{
  sequencia de comandos
}
```

1. Inicializações;
2. Verificação das condições;
3. Se verdadeiras: sequência de comandos;
4. Incrementos;
5. Verificação das condições;
6. Se verdadeiras: sequência de comandos;
7. ...

Exercícios

- Entrar com um inteiro *n*, e imprimir na tela os números ímpares menores ou iguais a *n*;
- Faça um programa que imprima em ordem decrescente todos os valores inteiros maiores que zero a partir de um número fornecido pelo usuário;

Observação: for

Observações

- Não tem ; no fim da linha do for;
- Se tiver uma instrução só, as chaves { e } são opcionais;
- É possível usar caracteres no lugar de inteiros:

```
for(c='a'; c <= 'z'; c++)
  printf("O valor ASCII de %c e:%d\n",c,c);
```

Recomendação: `for`

- Usar a estrutura `for` quando a quantidade de laços é previsível no momento da execução;
- Nos outros casos, usar uma estrutura `while` ou `do-while`.

Estrutura while

Sintaxe

```
while (expressao_de_teste)
{
  sequencia de comandos
}
```

Repetição com a estrutura while

- Usa apenas uma expressão de teste;
- Se tiver uma instrução só, as chaves { } são opcionais;
- Mais apropriado que a estrutura for em situações em que o laço pode ser terminado inesperadamente, em consequência das operações do corpo do laço;
- O corpo do laço será executado sempre que a expressão for verdadeira.

while: Passos de execução

```
while (expressao_de_teste)
{
  sequencia de comandos
}
```

1. Verificação da `expressão_de_teste`;
2. Se verdadeira: `sequência de comandos`;
3. Verificação da `expressão_de_teste`;
4. Se verdadeira: `sequência de comandos`;
5. ...

while: Exemplo

Programa: Imprimir a tecla digitada até o usuário digitar x:

```
printf("Entre com um caracter (x para sair): ");
scanf(" %c", &c);
while(c != 'x')
{
  printf("Digitou o caracter %c\n",c);
  printf("Entre com outro caracter (x para sair): ");
  scanf(" %c", &c);
}
```

while: Exercícios

Exercício: simular um for

Faça um programa usando a estrutura while que imprima em ordem decrescente todos os valores inteiros maiores que zero a partir de um número fornecido pelo usuário.

Exercício: Algarismos

Escreva um programa para determinar o número de algarismos de um número inteiro positivo dado.

Estrutura do-while

Sintaxe

```
do
{
   sequencia de comandos
}
while (expressao de teste)
```

do-while: Descrição

Repetição com a estrutura do-while

- Como a estrutura `while`, é usado quando a quantidade de repetições não é previsível no momento da execução;
- Se tiver uma instrução só, as chaves `{ }` são opcionais;
- Diferente do `while`, a expressão de teste fica no fim da estrutura;
- Repetição da `sequência de comandos` até a `expressão_de_teste` ser false;
- Consequentemente, o laço é sempre executado pelo menos uma vez.

do-while: Passos de execução

```
do
{
  sequencia de comandos
}
while (expressao de teste)
```

1. Sequência de comandos;
2. Verificação da expressão_de_teste;
3. Se verdadeira: sequência de comandos;
4. Verificação da expressão_de_teste;
5. ...

Exercício: Jogo de sorte

Escrever um programa em C que peça ao jogador para adivinhar o *número da sorte* (entre 0 e 100) gerado aleatoriamente pelo programa. O jogador vai entrando com números, e o programa vai informando se o número do jogador é maior ou menor que o número da sorte.
Quando o jogador acertar o número, o programa deve imprimir 'ACERTOU' e informar o número de tentativas do jogador.
Para gerar um número entre 0 e 100, use o comando:

```
num = rand()%100;
```

Estruturas aninhadas

Obviamente, estas estruturas podem ser aninhadas...
Por exemplo:

```
do
{
  sequencia de comandos (1)
  for (i=n1; i <= n2; i++)
  {
    sequencia de comandos (1.1)
  }
  sequencia de comandos (2)
while (expressao_de_teste);
```

Repetições aninhadas: ordem de execução

```
do
  sequencia de comandos (1)
  for (i=n1; i <= n2; i++)
  {
    sequencia de comandos (1.1)
  }
  sequencia de comandos (2)
while (expressao_de_teste);
```

1. `Sequência de comandos (1);`
2. Execução da estrutura `for`:
   1. `i = n1`
   2. Verificação se `i` $\leq$ `n2`;
   3. Se verdadeira: `sequência de comandos (1.1);`
   4. `...`
3. `Sequência de comandos (2);`
4. Verificação da `expressão_de_teste`;
5. Se verdadeira: `sequência de comandos(1);`
6. Execução da estrutura `for`:

Exercício: Desenho

Exercício: Estrelas

Usando apenas uma vez os comandos `printf(".")`, `printf("*")` e `printf("\n")` e usando laços aninhados, faça o seguinte aparecer na tela.

```
*****
.****
..***
...**
....*
.....
```

Exercício: Tabuada

Tabuadas de 1 até 5

Imprimir na tela as tabuadas de 1 até 5 da forma seguinte:

```
Tabuada de 1
 1 x  1 =  1
 1 x  2 =  2
 1 x  3 =  3
 1 x  4 =  4
 1 x  5 =  5
 1 x  6 =  6
 1 x  7 =  7
 1 x  8 =  8
 1 x  9 =  9
 1 x 10 = 10

Tabuada de 2
 2 x  1 =  2
 2 x  2 =  4
 2 x  3 =  6
 2 x  4 =  8
```

`break` e `continue`

O comando `break`

O comando `break` causa a saída imediata do laço.

O comando `continue`

Pula o código que estiver abaixo e força a próxima iteração do laço.

Observações

- Podem ser usado no corpo de qualquer estrutura (`for`, `while`, `do-while`);
- Se o `break` ou o `continue` estiver em laços aninhados, afetará somente o laço mais interno onde ele está.

`break` e `continue`: Recomendação

Evitar o uso dos comandos `break` e `continue`

Deve ser evitado, pois pode causar dificuldade de leitura e confusão na manutenção o programa.

Exemplo

```
while (1)
{
   printf("Digite um numero maior que zero: ");
   scanf("%d",&num);
   if(num<0)
   {
      printf("numero errado\n");
      continue;
   }
   printf("Numero correto\n");
}
```