

Fundamentos de Programação

Repetições aninhadas

Dainf - UTFPR

Profa. Leyza B. Dorini

Prof. Bogdan T. Nassu

Repetições aninhadas

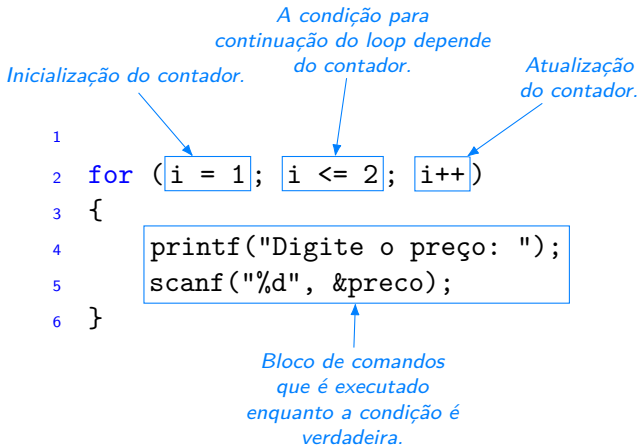
Blocos de comandos podem se tornar bastante complexos, pois a linguagem C permite que cada bloco contenha comandos com mais blocos, e assim sucessivamente. Agora, iremos estudar repetições dentro de repetições!

Caso 01: `for` como bloco de comandos de um `for`

Vamos começar analisando o que acontece quando temos um `for` dentro de um `for`.

Relembrando o for

Ao utilizar a estrutura de repetição for em laços determinados, em geral temos:



Repetições aninhadas em laços determinados com o for

Em repetições **aninhadas**, o bloco de comandos contém outra estrutura de repetição!

(1) *Teste da condição.*

(3) *Atualiza o contador do for mais externo.*

```
1  for (i = 1; i <= 2; i++)  
2  {  
3      printf ("i tem valor %d \n", i);  
4      for (j = 3; j <= 4; j++ )  
5      {  
6          printf ("%d %d \n", i, j);  
7      }  
8  }
```

(2) *Executa bloco de comandos - observe que ele possui outra estrutura de repetição.*

Repetições aninhadas em laços determinados com o for

Em repetições **aninhadas**, o bloco de comandos contém outra estrutura de repetição!

```
1 for (i = 1; i <= 2; i++)
2 {
3     printf ("i tem valor %d \n", i);
4     for (j = 3; j <= 4; j++ )
5     {
6         printf ("%d %d \n", i, j);
7     }
8 }
```

Para cada iteração do laço mais externo (ou seja, para cada novo valor de i) o bloco de comandos é executado uma vez.

Portanto, a cada iteração do laço mais externo, além de executar o printf da linha 3, o laço for mais interno é executado de forma completa.

Repetições aninhadas em laços determinados com o for

Desafio: você consegue determinar qual a saída para o exemplo?

```
1 for (i = 1; i <= 2; i++ )  
2 {  
3     printf ("i tem valor %d \n", i);  
4     for (j = 3; j <= 4; j++ )  
5     {  
6         printf ("%d %d \n", i, j);  
7     }  
8 }
```

—

```
i tem valor 1  
1 3  
1 4  
i tem valor 2  
2 3  
2 4
```

Repetições aninhadas em laços determinados com o for

Desafio: você consegue determinar qual a saída para o exemplo?

```
1 for (i = 1; i <= 2; i++ )  
2 {  
3     printf ("i tem valor %d \n", i);  
4     for (j = 3; j <= 4; j++ )  
5     {  
6         printf ("%d %d \n", i, j);  
7     }  
8 }
```

—

```
i tem valor 1  
1 3  
1 4  
i tem valor 2  
2 3  
2 4
```


Outras formas de repetições aninhadas

Nos slides anteriores, vimos como fazer repetições aninhadas com o `for`, mas também é possível fazer outras formas de repetições aninhadas (inclusive combinando loops determinados com indeterminados). Alguns exemplos a seguir.

Repetições aninhadas while com for

Mesmo exemplo anterior, mas implementando o laço determinado mais externo com o while.

Inicialização do contador.

*Teste da condição
depende do contador.*

```
1  i = 1;
2  while (i <= 2)
3  {
4      printf ("i tem valor %d \n", i);
5      for (j = 3; j <= 4; j++ )
6      {
7          printf ("%d %d \n", i, j);
8      }
9      i++;
10 }
```

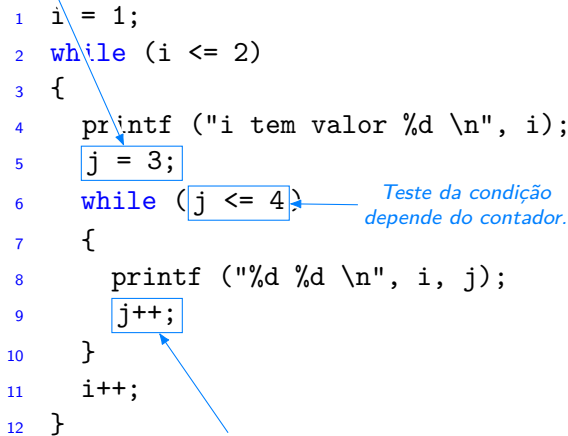
Atualização do contador.

Repetições aninhadas while com while

Mesmo exemplo anterior, mas implementando ambos os laços com o while.

Inicialização do contador.

```
1  i = 1;
2  while (i <= 2)
3  {
4      printf ("i tem valor %d \n", i);
5      j = 3;
6      while (j <= 4)
7      {
8          printf ("%d %d \n", i, j);
9          j++;
10     }
11     i++;
12 }
```



Atualização do contador.

Podemos ter diversos níveis...

```
1  int main()
2  {
3      int i, j, k;
4
5      for (i=1; i<=3; i++){
6          for (j=4; j<=5; j++)
7              for (k=1; k<=2; k++)
8                  printf ("%d %d %d\n", i, j, k);
9      }
10     return 0;
11 }
```

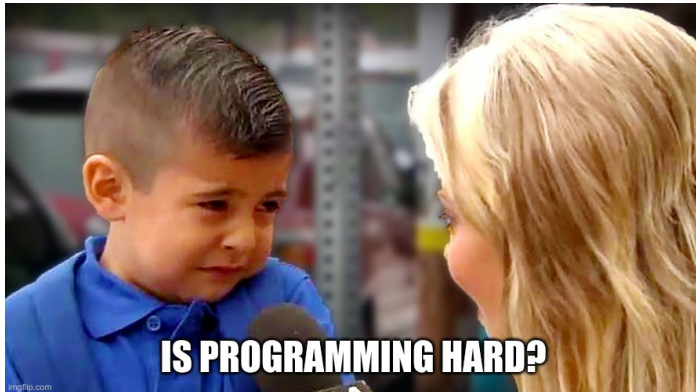
... mas tome cuidado

Embora seja possível utilizar vários níveis de repetições aninhadas, é preciso analisar ter cautela!



E tem muito mais...

Muitas outras combinações são possíveis! Entender os conceitos básicos vai permitir que você consiga resolver novos problemas, além daqueles contemplados aqui!



Erros comuns!

Preste atenção nas variáveis que controlam os laços!

Quando tiver repetições aninhadas, tome muito cuidado com mudanças em variáveis usadas em *loops* mais externos!!!

```
1
2
3 for (i = 0; i < 5; i++)
4 {
5     for (i = 0; i < 10; i++)
6     {
7         printf ("0i.\n");
8     }
9 }
10 printf ("Acabou");
```

*Observe que a variável i
está sendo utilizada
nos dois laços!*

*Neste código, o bloco de comandos entre as
linhas 5 e 8 será executado apenas uma vez, e não 5!
Isso porque ao final da primeira iteração do loop
mais interno o valor de i será 10,
fazendo com que a condição do
loop mais externo seja falsa.*

Analise se não é preciso reinicializar variáveis

Muitas vezes é preciso **reinicializar** variáveis a cada iteração do bloco de comandos do laço mais externo (principalmente acumuladoras e flags).

```
1
2 #define N_PROVAS 2
3 #define N_ALUNOS 3
4 int main () {
5     int i, j;
6     float soma, nota;
7
8     for (i=0; i < N_PROVAS; i++)
9     {
10         soma = 0;
11         printf ("Digite as notas da prova %d ", i);
12         for (j=0; j < N_ALUNOS; j++)
13         {
14             scanf ("%f", &nota);
15             soma = soma + nota;
16         }
17         printf ("Media da prova %d: %f\n", i, soma/N_ALUNOS);
18     }
19 }
```

Como precisamos calcular a soma das notas de todos os alunos (loop mais interno) para cada prova (loop mais externo), é necessário reinicializar a variável soma a cada iteração do loop mais externo.

Exemplos

Exemplo: quadrado 1-N

Faça um programa que defina uma macro N com valor igual a 5 e imprima N linhas na tela com o seguinte formato:


```
1 1 2 3 4 5
2 1 2 3 4 5
3 1 2 3 4 5
4 1 2 3 4 5
5 1 2 3 4 5
```

Neste contexto, você pode explorar a decomposição de problemas, identificando que é preciso repetir 5 vezes uma estrutura de repetição que imprime os valores de 1 até 5.

Exemplo: quadrado 1-N

```
1  #include<stdio.h>
2  #define N 5
3  int main()
4  {
5      int i, j;
6
7      for(i=1; i<=N; i++)
8      {
9          for(j=1; j<=N; j++)
10             printf("%d ", j);
11         printf("\n"); //observe onde está este \n
12     }
13
14     return 0;
15 }
```

*O bloco de comandos (o qual
é repetido N vezes) imprime
os valores entre 1 e N e depois
quebra a linha.*



Exemplo: triângulo 1-N

Faça um programa que defina uma macro N com valor igual a 5 e imprima N linhas na tela com o seguinte formato:


```
1 1
2 1 2
3 1 2 3
4 1 2 3 4
5 1 2 3 4 5
```

Aqui você precisa identificar o seguinte padrão: na primeira linha imprime até 1, na segunda linha até 2, ..., na N-ésima linha até N. Como adaptar o programa do slide anterior?

Exemplo: triângulo 1-N

```
1  #include<stdio.h>
2  #define N 5
3  int main()
4  {
5      int i, j;
6
7      for(i=1; i<=N; i++)
8      {
9          for(j=1; j<=i; j++)
10             printf("%d ", j);
11         printf("\n");
12     }
13     return 0;
14 }
```

*Basta variar o conteúdo de j
entre 1 e o valor de i
naquela iteração.*



Exemplo: imprimindo se **UM** número é primo

Adapte o programa abaixo de tal forma informar se um valor n é primo enquanto o valor digitado seja diferente de -1.

```
1  int main()
2  {
3      int div, candidato, eh_primo, nPrimos;
4
5      printf("Qual numero testar? ");
6      scanf("%d", &candidato);
7
8      eh_primo = 1;
9      for (div=2; div<=candidato-1; div++){
10         if (candidato%div == 0)
11             eh_primo = 0; // se teve divisor, altera flag
12     }
13     if(eh_primo==1)
14         printf("%d eh primo ", candidato);
15
16     return 0;
17 }
```

Exemplo: imprimindo quem é primo para n valores

```
1  #define N 5
2  int main ()
3  {
4      int n, div, eh_primo;
5
6      printf("Valor a testar: ");
7      scanf("%d", &n);
8      while(n != -1)
9      {
10         eh_primo = 1; //A flag é REINICIALIZADA para cada novo n.
11         for (div = 2; div < n && eh_primo; div++)
12             if (n%div == 0)
13                 eh_primo = 0; // Achou, inverte a flag.
14
15         if (eh_primo) // Verifica o status da flag.
16             printf ("%d eh primo.\n", n);
17
18         printf("Valor a testar: ");
19         scanf("%d", &n);
20     }
21     return 0;
22 }
```

Observe se flags e/ou acumuladores não precisam ser reinicializados a cada iteração.

Tarefa: fazer todos os exercícios da lista disponibilizada



Só praticando (e errando, consequentemente) iremos aprender programação!