

Estruturas de Repetição - Parte 1

Algoritmos e Programação de Computadores – ABI/LFI/TAI



Prof. Daniel Saad Nogueira
Nunes

IFB – Instituto Federal de Brasília,
Campus Taguatinga



Sumário

- 1 Introdução
- 2 While
- 3 Do While
- 4 For
- 5 Considerações



Sumário

1 Introdução



Introdução

- Até o momento sabemos realizar operações aritméticas em nossos programas.
- Também sabemos como testar uma determinada condição e realizar ações diferentes dependendo do valor desta condição: tudo isto usando operadores relacionais e lógicos.
- E se quisermos repetir um bloco de código diversas vezes até que uma condição seja atingida?



Introdução

- Um exemplo: vamos escrever um programa que imprima os números de 1 a 5.
- Com os conhecimentos que temos até agora, é possível fazer isso tranquilamente.



Introdução

```
1  #include <stdio.h>
2
3  int main(void) {
4      printf("1\n");
5      printf("2\n");
6      printf("3\n");
7      printf("4\n");
8      printf("5\n");
9      return 0;
10 }
```



Introdução

- E se quiséssemos fazer um programa que imprime números de 1 a 100?
- Poderíamos seguir a mesma estratégia, mas seria bem mais trabalhoso.



Introdução

```
1  #include <stdio.h>
2
3  int main(void) {
4      printf("1\n");
5      printf("2\n");
6      printf("3\n");
7      printf("4\n");
8      printf("5\n");
9      ...
10     printf("99\n");
11     printf("100\n");
12     return 0;
13 }
```




Introdução

- Se pudéssemos repetir o comando `printf` cem vezes, certamente nossa tarefa se tornaria mais fácil.
- Felizmente, podemos fazer isto, graças às **estruturas de repetição**.
- A linguagem C felizmente fornece para nós três estruturas de repetição: `while`, `do while` e `for`.
- Examinaremos agora cada uma delas.



Sumário

2 While



While

Fluxo: While

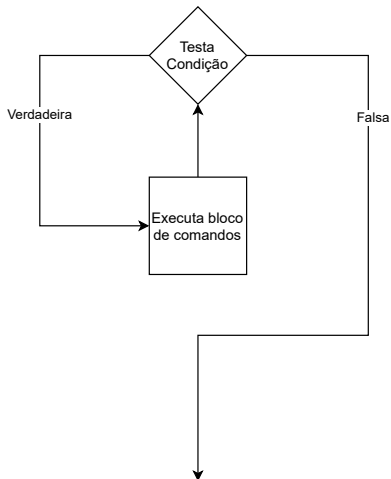
While (enquanto): enquanto a condição for verdadeira, faça.

A estrutura `while` segue o fluxo abaixo:

- 1 Testa uma condição, caso ela seja verdadeira, vá para o passo 2, caso contrário, vá para o passo 4.
- 2 Execute o bloco de comandos.
- 3 Volte para o passo 1.
- 4 Continue o fluxo normal de execução do programa.



While





While: Sintaxe

- Bloco com comando único:

```
1 while (condicao)
2     comando;
```

- Bloco com múltiplos comandos:

```
1 while (condicao) {
2     comando_1;
3     comando_2;
4     ...
5     comando_n;
6 }
```



While: Sintaxe

- A condicao é uma expressão e pode envolver operadores relacionais, aritméticos e lógicos.



While: Exemplo

- Vamos tomar o seguinte problema.
- O usuário deve digitar um valor n e o programa deverá imprimir todos os inteiros positivos até o valor n .



While: Exemplo

```
1  #include <stdio.h>
2
3  int main(void) {
4      int n;
5      scanf("%d", &n);
6      int i = 1;
7      while (i <= n) {
8          printf("%d\n", i);
9          i++;
10     }
11     return 0;
12 }
```




While: Exemplo

```
1  #include <stdio.h>
2
3  int main(void) {
4      int n;
5      scanf("%d", &n);
6      int i = 1;
7      while (i <= n) {
8          printf("%d\n", i);
9          i++;
10     }
11     return 0;
12 }
```

O que acontece quando o usuário digita 5?



While: Exemplo

```
1  #include <stdio.h>
2
3  int main(void) {
4      int n;
5      scanf("%d", &n);
6      int i = 1;
7      while (i <= n) {
8          printf("%d\n", i);
9          i++;
10     }
11     return 0;
12 }
```

O que acontece quando o usuário digita 0?



Sumário

3 Do While



Do While

- A estrutura `do while` em C primeiro executa os comandos para depois verificar se a condição é verdadeira.
- Enquanto a condição for verdadeira, volta a executar o bloco de comandos.
- Garante que o bloco de comandos é executado pelo menos uma vez.



Do While

Fluxo: Do While

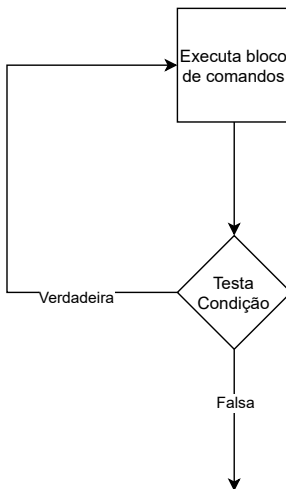
Do While: faça enquanto a condição for verdadeira.

A estrutura `do while` segue o fluxo abaixo.

- 1 Execute o bloco de comandos.
- 2 Verifique se a condição é verdadeira, em caso afirmativo, vá para o passo 1, caso contrário, vá para o passo 3.
- 3 Continue o fluxo normal de execução do programa.



Do While





Do While: Sintaxe

- Bloco com comando único:

```
1  do
2      comando;
3  while (condicao);
```

- Bloco com múltiplos comandos:

```
1  do {
2      comando_1;
3      comando_2;
4      ...
5      comando_n;
6  } while (condicao);
```



Do While: Sintaxe

- A condicao é uma expressão e pode envolver operadores relacionais, aritméticos e lógicos.



While: Exemplo

- Vamos tomar o mesmo problema anterior.
- O usuário deve digitar um valor n e o programa deverá imprimir todos os inteiros positivos até o valor n .
- Como resolvê-lo com `do while`?



Do While: Exemplo

```
1  #include <stdio.h>
2
3  int main(void) {
4      int n;
5      scanf("%d", &n);
6      int i = 1;
7      do{
8          printf("%d\n", i);
9          i++;
10     }while(i<=n);
11     return 0;
12 }
```



Do While: Exemplo

```
1  #include <stdio.h>
2
3  int main(void) {
4      int n;
5      scanf("%d", &n);
6      int i = 1;
7      do{
8          printf("%d\n", i);
9          i++;
10     }while(i<=n);
11     return 0;
12 }
```

O que acontece quando o usuário digita 5?



Do While: Exemplo

```
1  #include <stdio.h>
2
3  int main(void) {
4      int n;
5      scanf("%d", &n);
6      int i = 1;
7      do{
8          printf("%d\n", i);
9          i++;
10     }while(i<=n);
11     return 0;
12 }
```

O que acontece quando o usuário digita 0?



Sumário

3 Do While

- While vs Do While



While vs Do While

- Devemos empregar o `while` e o `do while` em situações adequadas.
- `while`: o teste da condição é feito antes da execução do bloco.
- `do while`: o teste da condição é feito apenas após a execução do bloco.
- Existem ocasiões em que o uso de um é mais apropriado que o uso do outro.
- Vamos examinar agora um problema cuja solução é mais natural com `do while`.



While vs Do While

- Neste problema, enquanto o usuário não digitar o valor 0, deverá ser lido um inteiro.
- Finalmente, quando o usuário digitar o valor 0, o programa deverá parar de ler os valores, apresentar a soma de todos os números lidos e encerrar.



Do While: Exemplo

```
1  #include <stdio.h>
2
3  int main(void) {
4      int numero;
5      int soma = 0;
6      do {
7          scanf("%d", &numero);
8          soma += numero;
9      } while (numero != 0);
10     printf("Soma total: %d\n", soma);
11     return 0;
12 }
```




While vs Do While

- O programa equivalente utilizando a estrutura `while` pode ser visto a seguir.



While: Exemplo

```
1  #include <stdio.h>
2
3  int main(void) {
4      int numero;
5      int soma = 0;
6      scanf("%d", &numero);
7      while (numero != 0) {
8          soma += numero;
9          scanf("%d", &numero);
10     }
11     printf("Soma total: %d\n", soma);
12     return 0;
13 }
```



Sumário

4 For

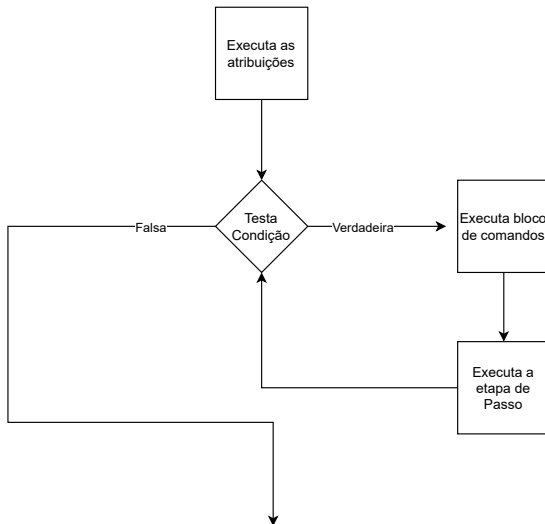


For

- A estrutura `for` em C tenta compactar um padrão muito comum observado em laços de repetição em C.
- Ele possui três mecanismos:
 - ▶ Atribuição: nesta etapa, todas as atribuições preliminares são feitas antes do laço propriamente dito. Estas atribuições não são executadas durante o laço, mas sim uma única vez.
 - ▶ Condição: Nesta etapa, a condição é verificada, se verdadeira, executa-se o bloco de comandos.
 - ▶ Passo: Os comandos descritos desta etapa só são executados caso o bloco de comandos seja executado da etapa anterior seja e sempre após ele.



For





For

Fluxo: For

A estrutura `for` (para) segue o fluxo abaixo:

- 1 Executa os comandos de atribuição.
- 2 Verifica a condição se verdadeira, vá para o passo 3, se falsa, vá para o passo 6.
- 3 Execute o bloco de comandos.
- 4 Execute os comandos de passo.
- 5 Vá para o passo 2.
- 6 Continue o fluxo normal do programa.



For: Sintaxe

- Bloco com comando único:

```
1  for (atribuicoes; condicao; passo)
2      comando;
```

- Bloco com múltiplos comandos:

```
1  for (atribuicoes; condicao; passo) {
2      comando_1;
3      comando_2;
4      ...
5      comando_n;
6  }
```



For: Sintaxe

- Os comandos de atribuição e passos devem ser separados por vírgulas.
- A condicao é uma expressão e pode envolver operadores relacionais, aritméticos e lógicos.



For: Exemplo

- Vamos tomar o problema de imprimir os n primeiros números inteiros positivos na tela, em que n é um número digitado pelo usuário.



For: Exemplo

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



For: Exemplo

- Agora, considere o problema de imprimir todos os pares de números inteiros positivos cuja soma é 100.



For: Exemplo

```
1  #include <stdio.h>
2
3  int main(void) {
4      int i,j;
5      for (i = 1,j=99; i <=j ; i++,j--) {
6          printf("%d + %d = 100\n", i,j);
7      }
8      return 0;
9  }
```



Sumário

4 For

- For vs While



For vs While

- É fácil ver que a estrutura `for` segue uma estrutura muito parecida a do `while`, mas de forma um pouco mais compacta.
- É possível escrever um código equivalente usando `while` da seguinte forma:

```
1  atribuicao_1;  
2  atribuicao_2;  
3  ...  
4  atribuicao_m;  
5  while (condicao) {  
6      comando_1;  
7      comando_2;  
8      ...  
9      comando_n;  
10     passo_1;  
11     passo_2;  
12     ...  
13     passo_l;  
14 }
```



For vs While

- Se não houver necessidade de executar comandos de atribuição ou passo, talvez seja mais interessante utilizar o `while`, por questões de legibilidade.



Sumário

5 Considerações



Considerações

- Vimos nesta aula as três estruturas de repetição suportadas pela Linguagem C.
- Cada uma delas possui uma determinada particularidade e deve ser usada sabiamente, especialmente no que tange o teste da condição no início ou no final.



Considerações

Boa Prática de Programação: Indentação

- Assim como as estruturas condicionais, é importante manter a indentação correta nas estruturas de repetição.
- A cada novo bloco de código, adicione um caractere de tabulação extra em relação ao bloco anterior.