Qualquer que seja o problema, sempre poderemos optar pela solução com WHILE ou DO... WHILE e teremos apenas essas duas diferenças citadas?

Resposta: nem sempre.

Tomemos o exemplo já visto anteriormente:

Faça um programa que leia uma sequência de números inteiros, terminada em zero e mostre cada número lido (exceto o zero).

Já estudamos a solução em C, usando o comando WHILE:

Portugol Studio	Linguagem C

inteiro num
leia(num)
escreva("Digite um número: ")
enquanto (num!=0)
{
  escreva (num)
escreva("Digite um número: ")
leia (num(
}
	int num;
	printf ("Digite um número: ");
	scanf("%d",&num);
	while (num!=0)
	{
	  printf ("O número lido foi = %d\n\n ",num);
	  printf ("Digite um número: ");
	  scanf("%d",&num);
	}

Como ficaria a solução com o comando DO... WHILE?

Em tese, bastaria:
- Na linha 04, adicionar o DO.
- Na linha 10 (nova), deslocar o **while (num!=0).**

Solução em C usando WHILE	Solução em C usando DO... WHILE
01 int num; **02 printf ("Digite um número: ");** 03 scanf("%d",&num); 04 while (num!=0) 05 { 06 printf ("O número lido foi = %d\n\n ",num); 07 **printf ("Digite um número: ");** 08 scanf("%d",&num); 09 }	01 int num; **02 printf ("Digite um número: ");** 03 scanf("%d",&num); 04 do 05 { 06 printf ("O número lido foi = %d\n\n ",num); 07 **printf ("Digite um número: ");** 08 scanf("%d",&num); 09 } 10 while (num!=0)

O que aconteceria se fizéssemos apenas essas 2 mudanças?

Analise o código da coluna **Solução C usando DO... WHILE**:

1) Antes do DO (linha 04), na linha 02 temos o comando **printf** e na linha 03 o comando **scanf**; temos novamente na linha 07 o mesmo comando **printf** da linha 03 e na linha 08 o mesmo comando **scanf** da linha 04.

   Isso é mesmo necessário?

   Vamos voltar um pouco quando fizemos o código com comando WHILE.

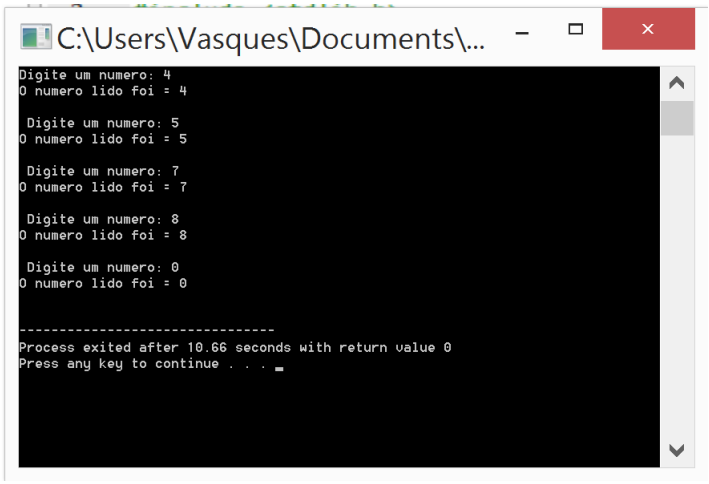
   Por que inserimos a leitura antes do WHILE?

   a. Resposta: para que tivéssemos um conteúdo válido para a variável **num** antes de realizar o teste, que é no início, e de executar a sequência de comandos a ser repetida.
   b. Mas com o comando DO... WHILE, o teste é no final da sequência a ser repetida, logo não precisamos dessa leitura inicial (printf e scanf das linhas 02 e 03, respectivamente) e podemos otimizar esse código com DO... WHILE, que ficaria assim:

Solução em C usando WHILE	**Solução em C usando DO... WHILE**
01 int num; **02 printf ("Digite um número: ");** 03 scanf("%d",&num); 04 while (num!=0) 05 { 06 printf ("O número lido foi = %d\n\n ",num); 07 **printf ("Digite um número: ");** 08 scanf("%d",&num); 09 }	01 int num; 02 do 03 { 04 **printf ("Digite um número: ");** 05 scanf("%d",&num); 06 printf ("O número lido foi = %d\n\n ",num); 07 } 08 while (num!=0);

2) Se você digitar o código acima no ambiente Dev-C++, vai perceber que a saída do programa escrito com DO... WHILE será distinta da do programa escrito com WHILE, quando for lido o número 0 (que determina o fim da sequência):
   a. Na solução com o comando WHILE, quando **num** for 0, nada será exibido.

b. Na solução com o comando DO... WHILE, quando **num** for 0, será exibido “O número lido foi 0”, conforme mostrado a seguir:



A solução para que o programa se comporte tal qual no programa escrito com WHILE, ou seja, sem mostrar “O número lido foi” quando **num** for zero, é simples: adicionar um teste de condição antes da linha 06 **printf ("O número lido foi = %d\n\n ",num);**

Dessa forma, o código do programa final ficaria assim (atente ao comando **if**, inserido logo após o comando **scanf**):

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
  int num;
  do
  {
    printf ("Digite um número: ");
    scanf("%d",&num);
    if (num!=0)
      printf ("O número lido foi = %d\n\n ",num);
  }
  while (num!=0);
  return 0;
}
```

Veja a execução do código acima, onde não é exibida a mensagem de “O número lido foi”, quando **num** for zero: