

Fundamentos de Programação

Loops indeterminados com o `while`

Dainf - UTFPR

Profa. Leyza B. Dorini
Prof. Bogdan T. Nassu

Estruturas de Repetição: caso 02 (com while)

Este material discute o uso do comando `while` para fazer *loops* indeterminados, ou seja, quando a quantidade repetições depende de algo que só será conhecido **dentro do bloco de comandos** da estrutura de repetição (resultado da análise de uma condição, cálculo, conteúdo de variável, ...).

Loops indeterminados com o while

A sintaxe do `while` é exatamente a mesma utilizada nos *loops* determinados:

```
1 while (condicao)
2 {
3     bloco de comandos
4 }
```

O que vai mudar é a forma como utilizamos esta estrutura: sabemos que o bloco de comandos é executado **enquanto** a condição for verdadeira, mas esta condição não necessariamente precisa estar ligada a um contador de iterações!

Loops determinados

Em um **loop indeterminado**, a quantidade de repetições depende de algo que só será conhecido dentro do bloco de comandos da estrutura de repetição:


```
1  int main () {  
2      int foo;  
3  
4      // Repete enquanto digitar valores positivos.  
5      scanf("%d", &foo);  
6      while (foo > 0)  
7      {  
8          printf ("ECO %d\n", foo);  
9          scanf ("%d", &foo);  
10     }  
11     return 0;  
12 }
```

O número de repetições do bloco das linhas 8-9 não é controlado por um contador de iterações, mas sim pelo **conteúdo de uma variável cujo valor é atualizado dentro do bloco**.

Loops indeterminados com o while

Embora não seja uma regra, inicialmente vamos utilizar a seguinte estrutura para *loops* indeterminados:

Antes do laço, é inicializada
a variável cujo conteúdo
é analisado na condição.



```
1  scanf ("%d ", &a);  
2  while (a <= 10)  
3  {  
4      printf ("Outros comandos...");  
5      scanf ("%d ", &a);  
6  }
```

Loops indeterminados com o while

Embora não seja uma regra, inicialmente vamos utilizar a seguinte estrutura para *loops* indeterminados:

Antes do laço, é inicializada
a variável cujo conteúdo
é analisado na condição.

1 scanf ("%d ", &a);

2 while (a <= 10)

3 {

4 printf ("Outros comandos...");

5 scanf ("%d ", &a);

6 }

A condição para
continuação
do loop depende
do valor desta variável.

Loops indeterminados com o while

Embora não seja uma regra, inicialmente vamos utilizar a seguinte estrutura para *loops* indeterminados:

Antes do laço, é inicializada
a variável cujo conteúdo
é analisado na condição.

1 `scanf ("%d ", &a);`

2 `while (a <= 10)`

3 `{`

4 `printf ("Outros comandos...");`

5 `scanf ("%d ", &a);`

6 `}`

A condição para
continuação
do loop depende
do valor desta variável.

Atualização do conteúdo da variável
que está controlando o laço é o
último comando do bloco. Dessa forma,
nenhum comando é executado entre a atribuição
do novo valor e o teste da condição!

Checkpoint

Faça um programa para ler números inteiros do teclado até que o usuário digite um valor ímpar (ou, em outras palavras, um programa que leia valores inteiros do teclado enquanto o conteúdo digitado for par)!

Loops indeterminados com o while

Inicialização da variável
cujo conteúdo
vai controlar o laço.

```
1 printf ("Digite o valor: ");  
2 scanf ("%d ", &num);  
3 while (num % 2 == 0)  
4 {  
5     printf ("Digite o valor: ");  
6     scanf ("%d ", &num);  
7 }
```

Loops indeterminados com o while

Inicialização da variável
cujo conteúdo
vai controlar o laço.

```
1 printf ("Digite o valor: ");
2 scanf ("%d ", &num);
3 while (num % 2 == 0)
4 {
5     printf ("Digite o valor: ");
6     scanf ("%d ", &num);
7 }
```

A condição para
continuação
do loop depende
do valor da variável
num.

Loops indeterminados com o while

Inicialização da variável
cujo conteúdo
vai controlar o laço.

```
1 printf ("Digite o valor: ");
2 scanf ("%d ", &num);
3 while (num % 2 == 0)
4 {
5     printf ("Digite o valor: ");
6     scanf ("%d ", &num);
7 }
```

A condição para
continuação
do loop depende
do valor da variável
num.

Atualização do conteúdo
da variável que está
controlando o laço é o
último comando do bloco.

Loops indeterminados

Note que não sabemos quantas vezes as instruções do bloco de comandos (linhas 9–10) serão repetidas, pois depende de quando o usuário vai digitar algum valor ímpar (pode ser na primeira iteração, na décima segunda...).

```
1  int main () {  
2      int num;  
3  
4      printf ("Digite o valor: ");  
5      scanf ("%d", &num);  
6  
7      while (num % 2 == 0)  
8      {  
9          printf ("Digite o valor: ");  
10         scanf ("%d", &num);  
11     }  
12     printf ("Acabou.");  
13     return 0;  
14 }
```

Como determinar a condição do `while`?

Cuidado ao determinar a condição

Quando estamos aprendendo a usar *loops* indeterminados, é muito comum criar *loops* infinitos (ou então fazer programas que nunca executam o bloco de comandos)!



Cuidado ao determinar a condição

Para determinar a condição corretamente, observe que ela consiste no critério para **CONTINUAR** a executar os comandos do bloco! No exemplo abaixo, os comandos das linhas 3 e 4 continuam a ser executados enquanto `preco >= 0`, concorda?

```
1  while (preco >= 0)
2  {
3      printf ("Digite o preco: ");
4      scanf ("%d",&preco);
5  }
```

Cuidado ao determinar a condição

Por outro lado, os enunciados dos exercícios normalmente mencionam o **critério de parada**: “faça um programa que leia valores do teclado até que o usuário digite um preço negativo”.

Note que o que foi enfatizado no enunciado foi o critério de parada do laço: `preco < 0`.

Cuidado ao determinar a condição

Por outro lado, os enunciados dos exercícios normalmente mencionam o **critério de parada**: “faça um programa que leia valores do teclado até que o usuário digite um preço negativo”.

Note que o que foi enfatizado no enunciado foi o critério de parada do laço: `preco < 0`.

Determinando a condição

Portanto, uma boa estratégia consiste em identificar no enunciado qual o critério de parada, e então “inverter” para determinar qual a condição para continuidade do laço!

Em exercícios mais complexos — e em problemas reais — você precisará ser capaz de identificar este tipo de situação sem que ela esteja explícita.

Checkpoint

Para o enunciado abaixo, determine (1) o critério de parada e (2) qual seria a condição do laço indeterminado!

Exercício

Escreva um programa que repita a leitura de uma senha até que ela seja válida.

(1) critério de parada :

(2) condição do laço :

Checkpoint

Para o enunciado abaixo, determine (1) o critério de parada e (2) qual seria a condição do laço indeterminado!

Exercício

Escreva um programa que repita a leitura de uma senha até que ela seja válida.

(1) critério de parada : `valorLido == senhaCorreta`

(2) condição do laço : `while(valorLido != senhaCorreta)`

Posso fazer *loops* indeterminados com `for`?

É possível, mas não é usual! Isso se explica pela própria estrutura do comando `for`, que inclui inicialização, condição e passo (que se adapta perfeitamente aos *loops* determinados).

```
for (; x != 0;) // Funciona, mas é muito feio.
```

for ou while?



Uma prática comum é usar:

- `for` para *loops* **determinados** e
- `while` para *loops* **indeterminados**

Por esta razão, nestes slides usaremos apenas `while`.

Por que usar essa estruturação?

Inicialização do conteúdo
da variável que
vai controlar o laço.

```
1  
2 scanf ("%d ", &preco);  
3 while (preco >= 0)  
4 {  
5     printf ("Digite o preco: ");  
6     scanf ("%d ", &preco);  
7 }
```

A condição para
continuação
do *loop* depende
do valor da variável
preco.

Atualização do conteúdo
da variável que está
controlando o laço é o
último comando do bloco.

Modificação 1

```
1  int main () {  
2      int preco, total=0;  
3  
4      //scanf ("%d", &preco);  
5      while (preco >= 0)  
6      {  
7          total = total + preco;  
8          scanf ("%d", &preco);  
9      }  
10     printf ("Total = %d ", total);  
11     return 0;  
12 }
```

Se omitirmos a leitura do conteúdo inicial da variável cujo conteúdo controla o loop...

...um valor inconsistente será considerado na condição que controla a estrutura de repetição.

Modificação 2

Embora seja possível fazer a inicialização da variável `preco` de tal forma que ela não interfira no valor final da variável `total`, na primeira passagem pelo laço a soma seria desnecessária.

```
1  int main () {  
2      int preco = 0, total=0;  
3  
4      //scanf("%d", &preco);  
5      while ( preco >= 0 )  
6      {  
7          total = total + preco;  
8          scanf("%d", &preco);  
9      }  
10     printf("Total = %d ", total);  
11     return 0;  
12 }
```

“Funciona”, mas é deselegante e pode levar a erros em certos contextos!

Modificação 3

Se a leitura do conteúdo da variável cujo conteúdo controla a estrutura de repetição não for o último comando do bloco...

... cuidado com as inconsistências: neste exemplo, além do valor digitado na linha 4 ser desconsiderado, o valor negativo será contabilizado na variável total.

```
1  int main () {  
2      int preco, total=0;  
3  
4      scanf ("%d", &preco);  
5      while (preco != 0)  
6      {  
7          scanf ("%d", &preco);  
8          total = total + preco;  
9      }  
10     printf ("Total = %d ", total);  
11     return 0;  
12 }
```

Portanto...

Até que você adquira mais experiência com o uso de *loops* indeterminados, considere a seguinte estruturação¹:

Inicialização da variável que
vai controlar o laço.

```
1  scanf ("%d ", &preco);  
2  while (preco >= 0)  
3  {  
4      //restante dos comandos  
5      scanf ("%d ", &preco);  
6  }
```

A condição para
continuação
do loop depende
do valor da variável.

Atualização da variável que está
controlando o laço é o **último comando do bloco**.

¹A inicialização e a atualização não necessariamente serão com `scanf`, a questão é **onde** estes eventos aparecem no código! A atualização poderia ser, por exemplo, um cálculo que muda o valor da variável.

Não é regra, mas...

Utilizando a estruturação sugerida, minimizamos os principais problemas encontrados por quem está começando a programar!

Como a variável que vai controlar
o laço é inicializada antes da
primeira comparação....


```
1  
2 scanf ("%d ", &preco);  
3 while (preco >= 0)  
4 {  
5     //restante dos comandos  
6     scanf ("%d ", &preco);  
7 }
```

...evitamos que a análise da condição use valores inconsistentes.

Não é regra, mas...

Utilizando a estruturação sugerida, minimizamos os principais problemas encontrados por quem está começando a programar!

```
1
2  scanf ("%d ", &preco);
3  while (preco >= 0)
4  {
5      //restante dos comandos
6      scanf ("%d ", &preco);
7  }
```



Como a atualização do conteúdo da variável que está controlando o laço é o **último comando** do bloco, o laço será interrompido assim que a variável tiver um valor que corresponde ao critério de parada.

Existem outras possibilidades

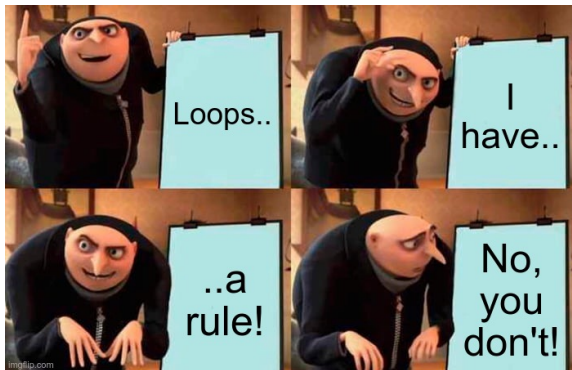
É claro que a atualização do conteúdo da variável que controla o laço não precisa ser exatamente o **último comando** do bloco! O que importa é que o seu conteúdo não interfira no resultado do processamento!

```
1      (...)
2      while (totalDoacoes < capacidade){
3          printf("\n\nNova doacao: (em kg): ");
4          scanf("%d", &doacaoAtual);
5
6          if (totalDoacoes+doacaoAtual <= capacidade)
7              totalDoacoes += doacaoAtual;
8          else
9              printf("Não coube no container! \n");
10
11         printf("\n---Relatorio parcial---\n");
12         printf("Doacoes recebidas: %d\n", totalDoacoes);
13         printf("Espaco: %d\n", capacidade-totalDoacoes);
14     }
15     (...)

```

Existem outras possibilidades

Para compreensão, dividimos as repetições em “determinadas” e “indeterminadas”. Com a prática, você vai perceber que os problemas não necessariamente se encaixam em somente uma definição, e nem que essas “receitas de bolo” necessariamente precisam ser seguidas. Mas nessa fase inicial de aprendizado elas vão te ajudar a compreender melhor estruturas de repetição!



Existem outras possibilidades

No código abaixo, por exemplo, usamos um `for` para fazer um laço indeterminado, aproveitando a estrutura da sua sintaxe para inicializar e incrementar um contador. Em resumo: muitas coisas são “possíveis”, mas podem nos confundir no início do aprendizado! ;-)

```
1  int main () {
2      int iteracoes, parcela;
3
4      printf ("Numero a ser somado (0 para sair): ");
5      scanf("%d", &parcela);
6
7      for (iteracoes=0; parcela != 0; iteracoes++) {
8          printf("Numero a ser somado (0 para sair): ");
9          scanf("%d", &parcela);
10     }
11
12     printf("Qtde de iteracoes: %d\n", iteracoes);
13     return 0;
14 }
```

Agora é contigo: faça a lista de exercícios!