

Fundamentos de Programação

Loops determinados com o `while`

Dainf - UTFPR

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

Estruturas de Repetição: caso 01

(com while)

Este material discute o uso do comando `while` para fazer laços de repetição (*loops*)! A princípio, vamos explorar o caso em que a quantidade de vezes que um comando (ou bloco de comandos) deve ser executado é **pré-determinada**, ou seja, é conhecida antes do início da estrutura de repetição!

Variáveis contadoras

Antes, é preciso aprender um conceito importante: variáveis contadoras! Elas é que vão nos ajudar a *controlar quantas vezes um bloco de comandos será executado*¹!

¹Lembrando: cada execução do bloco é uma *iteração* (e não iNteração!!!).

Contadores

Um contador nada mais é do que uma variável utilizada para contabilizar quantas vezes um evento de interesse ocorre em um programa.

- Em termos de implementação, um “contador” é uma variável que recebe o seu próprio conteúdo incrementado (ou decrementado) de um valor constante. Por exemplo:

```
cont = cont + 1;
```

Inicialização dos contadores

Como um “contador” é uma variável que recebe o seu próprio conteúdo somado (ou subtraído) de um valor constante, o resultado será inconsistente caso não seja corretamente inicializada! Analise o exemplo abaixo e responda: o que será impresso na tela?

```
1 int main(){
2     int cont; //note que não foi inicializado
3
4     cont = cont + 1;
5     printf("Total: %d", cont);
6 }
```

```
1 Total: 2130567171
```

Este é um *bug* muito comum, e pode ser difícil de detectar, porque frequentemente este lixo vale 0. Mas não sempre!

Inicialização dos contadores

Como um “contador” é uma variável que recebe o seu próprio conteúdo somado (ou subtraído) de um valor constante, o resultado será inconsistente caso não seja corretamente inicializada! Analise o exemplo abaixo e responda: o que será impresso na tela?

```
1 int main(){
2     int cont; //note que não foi inicializado
3
4     cont = cont + 1;
5     printf("Total: %d", cont);
6 }
```

Na linha 4, a variável `cont` recebeu o próprio valor somado de um. Contudo, como ela não foi corretamente inicializada, essa soma considerou `cont = lixo + 1`, causando a seguinte saída:

```
1 Total: 2130567171
```

Este é um *bug* muito comum, e pode ser difícil de detectar, porque frequentemente este lixo vale 0. Mas não sempre!

Operador de incremento

Se usarmos um contador para contar iterações de um *loop*, ele quase sempre será atualizado de um em um! Portanto, podemos utilizar o operador de incremento!!

Lembre-se que

`i++;`

é equivalente a

`i=i+1;`

Loops determinados com o comando `while`

Loops determinados com o while

Na primeira estrutura de repetição que veremos, um bloco de comandos é executado **enquanto** uma condição for verdadeira.

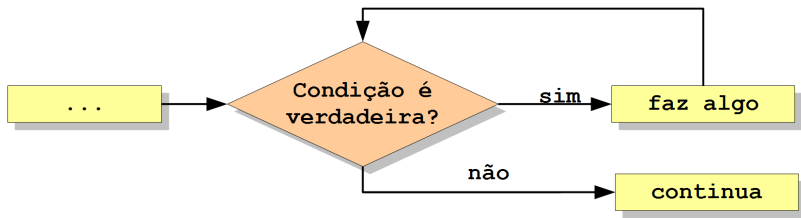
Sintaxe:

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

Assim como no if, não existe ; após a condição!

Assim como no if, se houver apenas um comando no bloco, as chaves são opcionais!

Fluxo do while



Loops determinados

Em um **loop determinado**, a quantidade de repetições é conhecida antes do início da estrutura de repetição. Por exemplo, segue um programa que lê e repete 10 inteiros do teclado:

```
1  int main ()
2  {
3      int i, num;
4
5      i = 0;
6      while (i < 10)
7      {
8          scanf ("%d", &num);
9          printf ("Digitou %d\n", num);
10         i++; //equivalente a i=i+1;
11     }
12
13     return 0;
14 }
```

O bloco de comandos das linhas 8-10 será executado 10 vezes, independentemente do número digitado pelo usuário.

Loops determinados com o while

Ao usar o `while` em *loops* determinados, em geral os seguintes pontos principais:

(0) Inicialização do contador.

1 `i = 0;`

(1) A condição para continuação do loop depende do contador.

2 `while (i < 10)`

3 `{`

4 `scanf ("%d", &num);`

5 `printf ("Digitou %d\n", num);`

6 `i++;`

7 `}`

(3) Atualização do contador (lembre-se que `i++`; é equivalente a `i=i+1`);

(2) bloco de comandos que é executado se a condição for verdadeira

Loops determinados com o while: inicialização (0)

Primeiro passo: inicializar o contador, que é a variável que vai nos ajudar a controlar quantas vezes o *loop* será executado.

Inicialização
do contador.

Observe que, como está “antes”
do while, vai ser executado apenas uma única vez
(ou seja, não pertence ao bloco que será repetido).

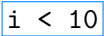
```
1  i = 0;
2  while (i < 10)
3  {
4      scanf ("%d", &num);
5      printf ("Digitou %d\n", num);
6      i++;
7  }
```

Loops determinados com o while: *loop* (1)

Agora entramos de fato no *loop*: a condição é analisada...

Note que o teste da condição depende do conteúdo da variável contadora naquela iteração.


```
1 i = 0;
2 while (i < 10)
3 {
4     scanf ("%d", &num);
5     printf ("Digitou %d\n", num);
6     i++;
7 }
```



Loops determinados com o while (2): loop

... se a condição for verdadeira, o bloco de comandos será executado...

```
1  int i = 0;
2  while (i < 10)
3  {
4      scanf ("%d", &num);
5      printf ("Digitou %d\n", num);
6      i++;
7  }
```



Bloco de comandos que será executado caso a condição seja verdadeira.

Loops determinados com o while (3): loop

Um ponto muito importante: o contador **precisa** ser atualizado dentro do bloco de comandos! Depois, a condição será testada novamente!

```
1  int i = 0;
2  while (i < 10)
3  {
4      scanf ("%d", &num);
5      printf ("Digitou %d\n", num);
6      i++;
7  }
```

Atualização do contador
faz parte do bloco de comandos.

Desta forma, ocorre o “controle”
que o bloco foi executado
mais uma vez

Em resumo....

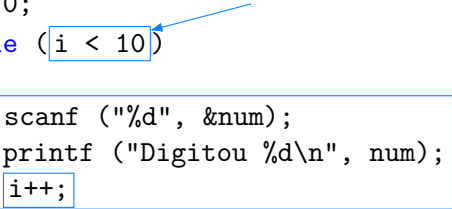
Loops determinados com o while: repassando

Como estamos em um laço de repetição (*loop*), **enquanto a condição for verdadeira** será feito:

- 1 Teste da condição.

```
1  i = 0;
2  while (i < 10)
3  {
4      scanf ("%d", &num);
5      printf ("Digitou %d\n", num);
6      i++;
7  }
```

1. Teste da condição.



Loops determinados com o while: repassando

Como estamos em um laço de repetição (*loop*), **enquanto a condição for verdadeira** será feito:

- ① Teste da condição.
- ② Execução do bloco de comandos.

```
1 i = 0;
2 while (i < 10)
3 {
4     scanf ("%d", &num);
5     printf ("Digitou %d\n", num);
6     i++;
7 }
```

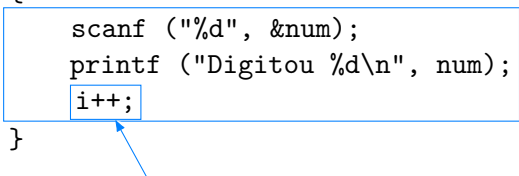
2. Bloco de comandos.

Loops determinados com o while: repassando

Como estamos em um laço de repetição (*loop*), **enquanto a condição for verdadeira** será feito:

- ① Teste da condição.
- ② Execução do bloco de comandos.
- ③ Atualização do contador (dentro do bloco de comandos).

```
1 i = 0;
2 while (i < 10)
3 {
4     scanf ("%d", &num);
5     printf ("Digitou %d\n", num);
6     i++;
7 }
```



3. Atualização do contador.

Loops determinados com o while: repassando

Como estamos em um laço de repetição (*loop*), **enquanto a condição for verdadeira** será feito:

- ① Teste da condição.
- ② Execução do bloco de comandos.
- ③ Atualização do contador (dentro do bloco de comandos).

→ Testa novamente a condição.
Se for verdadeira, executa bloco de comandos. Senão, sai do *loop* e continua a execução do programa.

```
1 i = 0;  
2 while (i < 10) {  
3     {  
4         scanf ("%d", &num);  
5         printf ("Digitou %d\n", num);  
6         i++;  
7     }  
}
```

E se ...

Na sequência, vamos discutir alguns erros comuns para quem está começando a programar!

Loops determinados com o while: perguntas...

Questões:

- O que aconteceria se a linha `i++` não estivesse dentro do bloco de comandos?
- O que aconteceria se a condição fosse `i > 10`?
- O que aconteceria se a condição fosse `i <= 10`?
- Por que começar a contagem em 0?

```
1 i = 0;
2 while (i < 10)
3 {
4     scanf ("%d", &num);
5     printf ("Digitou %d\n", num);
6     i++;
7 }
```

O que acontece ao retirar `i++` do bloco de comandos?

O valor de `i` nunca seria alterado, ou seja, seria sempre 0.
Portanto, ocorre um ***loop infinito***!

```
1 i = 0;
2 while (i < 10)
3 {
4     scanf ("%d", &num);
5     printf ("Digitou %d\n", num);
6     //i++;
7 }
```


O que aconteceria se a condição fosse $i > 10$?

Note que o teste da condição ocorre antes do bloco ser executado a primeira vez. Antes da primeira iteração, testaríamos se $0 > 10$, o que não é verdade. Logo, o bloco de comandos **nunca seria executado!**

```
1 i = 0;
2 while (i > 10)
3 {
4     scanf ("%d", &num);
5     printf ("Digitou %d\n", num);
6     i++;
7 }
```

O que aconteceria se a condição fosse $i \leq 10$?

Teríamos **11 iterações em vez de 10**. Isto acontece porque começamos a contar de 0!

```
1 i = 0;
2 while (i <= 10)
3 {
4     scanf ("%d", &num);
5     printf ("Digitou %d\n", num);
6     i++;
7 }
```

Por que começar a contagem em 0?



Three programmers walk into a bar...

Isto é uma tradição. No futuro, também será útil quando trabalharmos com vetores. Mas é uma questão estranhamente profunda! Depois procure por “*why do programmers start counting from 0*” no Google e verá!

Loops determinados com o while

Lembre-se também que:

Chaves são
opcionais
se o bloco
tiver apenas um
comando.

```
1 i = 0;
2 while (i < 10)
3 {
4     scanf ("%d", &num);
5     printf ("Digitou %d\n", num);
6     i++;
7 }
```

Checkpoint 1

Antes de seguir, responda: qual o papel de cada um dos seguintes pontos?

(1) ??

1 `int i = 1;`

(2) ??

2 `while (i <= 10)`

3 `{`

4 `printf("Digite o preço");`

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

6 `i++;`

7 `}`

(3) ??

Checkpoint 1

Antes de seguir, responda: qual o papel de cada um dos seguintes pontos?

(1) Inicialização do contador.

(2) Condição que determina se o bloco será executado ou não.

```
1  int i = 1;
2  while ( i <= 10 )
3  {
4      printf("Digite o preço");
5      scanf("%d", &preco);
6      i++;
7  }
```

(3) Atualização do contador.

Checkpoint 2

O que faz o seguinte programa?

```
1  #define N 5
2
3  int main ()
4  {
5      int i = 0;
6      while (i <= N)
7      {
8          printf("%d \n", i);
9          i++;
10     }
11     return 0;
12 }
```

Checkpoint 2

O que faz o seguinte programa?

```
1  #define N 5
2
3  int main ()
4  {
5      int i = 0;
6      while (i <= N)
7      {
8          printf("%d \n", i);
9          i++;
10     }
11     return 0;
12 }
```

Imprime na tela os valores de 0 até 5, um em cada linha.

Boas práticas...

Você percebeu que estamos usando a variável `i` como contadora? Embora não seja uma “regra”, é um padrão comum para os programadores.

Choosing an iterator for your for loop like



Nada impede que você use uma variável chamada `h` ou `contador_da_leyza`. Entretanto, como `i` é a notação mais utilizada, é uma boa prática a seguir (lembre-se que o seu código será manipulado por outros programadores).

Atualização do contador


Em alguns casos, além de ajudar a contabilizar a quantidade de iterações, o contador também faz parte da solução do problema. Nestes casos, nem sempre ele vai ser *incrementado em uma unidade* a cada iteração!!!

Atualização do contador

No exemplo abaixo, onde desejamos imprimir uma contagem regressiva (de 10 a 1), o contador é **decrementado** em um a cada iteração!

```
1 printf ("Imprimindo contagem regressiva ... ");
2 i = 10;
3 while (i >= 1)
4 {
5     printf ("%d \n", i);
6     i--;
7 }
```

3. atualiza o contador



Lembre-se

`i--`; é equivalente a `i=i-1`;

Atualização do contador

Cuidado com a posição do incremento / decremento. Diferente do caso anterior, que mostrava 10, 9, ..., 3, 2, 1, o programa abaixo mostrará 9, 8, ..., 2, 1, 0.

```
1 printf ("Imprimindo contagem regressiva ... ");
2 i = 10;
3 while (i >= 1)
4 {
5     i--;
6     printf("%d \n", i);
7 }
```

Dica

Deixe a atualização do contador de iterações como último comando do bloco, a não ser que tenha um ótimo motivo para fazer diferente!

Atualização do contador

Neste exemplo, onde precisamos imprimir os pares entre 10 e 20, o contador é incrementado de dois em dois.

```
1 printf ("Os pares entre 10 e 20 são: ");
2 i = 10;
3 while (i <= 20)
4 {
5     printf ("%d \n", i);
6     i += 2; // i += 2 equivale a i = i+2
7 }
```

3. atualiza o contador

Cuidado ao alterar o valor do contador!

Para alterar o conteúdo é preciso fazer `i = i+2;` ou `i += 2;`. Se você fizer apenas `i+2;` o programa vai entrar em loop infinito (pois o valor nunca será atualizado).

Exemplos - while

Vejamos mais algumas situações interessantes...

Ex. 01 - Imprimindo os números inteiros de 1 a n

```
1  int main () {  
2      int i=1, n;  
3  
4      scanf ("%d",&n);  
5      while (i<=n) {  
6          printf("%d ",i);  
7          i++;  
8      }  
9      return 0;  
10 }
```

Note que, embora o *loop* seja executado n vezes, **é um loop determinado**. Isso pode confundir quem está começando a programar, pois “executar n vezes” nos induz a pensar que é indeterminado. Mas o n é conhecido antes do *loop* começar!

Ex. 02 - Ler 10 números e contar quantos são pares

```
1  int main ()
2  {
3      int i = 0, aux, contPares = 0;
4
5      while (i < 10)
6      {
7          scanf ("%d", &aux);
8          if (aux % 2 == 0)
9              contPares++;
10             i++;
11     }
12     printf("Qtde de pares: %d.", contPares);
13
14     return 0;
15 }
```

Podemos usar sem problemas um
if dentro do bloco do while!
Repare na indentação.

Neste exemplo, temos **dois contadores** — um para as iterações e outro para contar quantos números pares foram digitados.

Tarefa: fazer exercícios da lista!

Se você entendeu os exemplos até aqui, já entendeu a parte “mecânica” do uso do `while` para realizar *loops* determinados. Daqui para a frente, só praticando! E não desanime se surgirem dificuldades na resolução dos exercícios!