Fundamentos de Programação Controle de fluxo com break e continue

Dainf - UTFPR

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

break e continue

Os comandos break e continue servem para modificar o fluxo de um *loop* durante a sua execução.

break: interrompe imediatamente o loop.

continue: inicia imediatamente a próxima iteração do loop, pulando o restante do bloco.

Nenhum dos dois é obrigatório. Tudo o que se pode fazer com estes comandos pode ser obtido por outros meios...

- ... às vezes usar break e continue é menos trabalhoso!
- ... frequente fazer com break e continue é menos elegante!

break

Lembrando: usando flag para controlar o fluxo

Este programa informa se um dado número n é primo. Usamos a flag para interromper o loop quando um divisor de n é encontrado.

```
int n, div, eh_primo;
1
2
        scanf("%d", &n);
3
        eh_primo = 1;
5
        for (div = 2; div < n && eh_primo; div++)</pre>
6
            if (n%div == 0)
                 eh_primo = 0;
8
q
        if (eh_primo)
10
            printf ("Eh primo.\n");
11
       else
12
            printf ("Nao eh primo.");
13
```

Usando break para controlar o fluxo

Nesta versão, usamos o break para interromper o *loop* quando um divisor de n é encontrado. O resto do programa não muda!

```
for (div = 2; div < n; div++)
    if (n%div == 0)

{
    eh_primo = 0;
    break;
}</pre>
```

As duas soluções não são **exatamente** equivalentes. Se um divisor for encontrado, na versão somente com a *flag*, o valor de div vai ser incrementado antes do *loop* terminar. Já com o break, o *loop* será interrompido imediatamente, sem incrementar div mais uma vez! Então, se quisermos saber qual foi o divisor encontrado, na versão com o break é div. mas na versão sem o break é div-1.

Com break \times sem break

```
for (div = 2; div < n && eh_primo; div++)</pre>
1
           if (n%div == 0)
2
               eh_primo = 0;
3
           else
               printf ("%d nao eh divisor.\n", div);
5
       for (div = 2; div < n; div++)
1
           if (n%div == 0)
3
               eh_primo = 0;
               break;
5
           }
6
           printf ("%d nao eh divisor.\n", div);
```

A parte abaixo do bloco do if jamais será atingida após o break, então não precisamos "protegê-la" com o else.

Outro exemplo (com o break)

O programa abaixo mostra o menor divisor > 1 comum a v1 e v2. Note que usamos o valor de div para saber se o loop foi até o final.

```
int v1, v2, div;
1
2
       scanf ("%d %d", &v1, &v2);
3
       for (div = 2; div <= v1 && div <= v2; div++)
5
            if (v1 % div == 0 && v2 % div == 0)
6
                break;
8
       if (div > v1 \mid | div > v2)
q
           printf ("Nenhum divisor maior que 1.\n");
10
       else
11
           printf ("O menor divisor eh %d\n", div);
12
```

Outro exemplo (sem o break)

Usando uma *flag*, o teste final é mais claro, mas precisamos lembrar que o incremento ocorrerá antes do *loop* ser interrompido!

```
int v1, v2, div, achou = 0;
1
2
       scanf ("%d %d", &v1, &v2);
3
       for (div = 2; div <= v1 && div <= v2 && !achou; div++)</pre>
5
            if (v1 % div == 0 && v2 % div == 0)
6
7
                achou = 1;
8
                div--: // Porque logo depois vem um div++
9
10
11
       if (!achou)
12
            printf ("Nenhum divisor maior que 1.\n");
13
       else
14
            printf ("O menor divisor eh %d\n", div);
15
```

continue

Usando continue para controlar o fluxo

O continue pula todo o restante de um bloco e inicia imediatamente a próxima iteração de um *loop*. Ele **executa** o incremento e o teste da condição!

```
// Somando numeros de 0 a N, pulando os divisores de 3.
2
        int i, total = 0;
3
        for (i = 0; i < N; i++)</pre>
5
6
            if (i\%3 == 0)
                 continue;
8
            total += i;
        }
10
11
        printf ("%d\n", total);
12
```

Com continue \times sem continue

Na maior parte dos casos, podemos evitar o uso do continue...

```
for (i = 0; i < N; i++)
{
    if (i%3 == 0)
        continue;
    total += i;
}

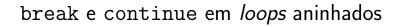
for (i = 0; i < N; i++)
    if (i%3 != 0)
        total += i;</pre>
```

break e continue: como não usar?

Código feio!

As construções abaixo, apesar de funcionarem costumam ser confusas ou inúteis. Evite-as!!! (Mas, como sempre, não existe uma regra absoluta!)

```
while (1)
2
       (\ldots)
       if (aconteceu alguma coisa)
           break;
  }
   while (alguma condicao)
  {
2
       if (aconteceu alguma coisa)
           continue;
       else
           faz alguma coisa interessante
  }
```



O break interrompe o loop interno!

```
1 (...)
while (foo)
4 while (bar)
6 (...)
7 if (alguma_condicao)
      break;
      (...) // Pula esta parte se entrar no if.
10
11
    (...) // O break manda o fluxo para ca.
14 (...)
```

O continue vai para a próxima iteração do loop interno!

```
1 (...)
while (foo)
    while (bar) // O continue volta para ca.
       (\ldots)
      if (alguma_condicao)
         continue;
       (...) // Pula esta parte se entrar no if.
10
11
     (...) // O continue nao afeta esta parte.
14 (...)
```