break e continue

Módulo 5 Aula 3

Linguagem C, o Curso Definitivo WR Kits

Autor: Dr. Eng. Wagner Rambo

break

Você já viu a instrução *break* em conjunto com *switch* aqui no curso. Lá aprendemos que a mesma pode ser utilizada para encerrar o *switch* ou laços de repetição. Pois bem, este é o propósito da mesma. Considere o código do Box 1.

```
main()
{
  char i = 0;

  while(i < 15)
  {
     if(i==10) break;
     i++;
     printf("%d\n",i);

  } /* end while */


} /* end main */
```

Box 1 - Exemplo de utilização da instrução break.

Observe que o laço será interrompido antes da condição ser invalidada pelos incrementos de i. Quando o valor chegar em 10, o laço é interrompido. Em outras palavras, serão impressos os números de 1 a 10 na tela, apesar da condição se manter verdadeira. Logo, você poderá utilizar um *break* para desviar para a instrução imediatamente após o final do laço.

O programa a seguir vai imprimindo números inteiros na tela em sequência até que seja pressionada uma tecla pelo usuário.

```
main()
{
  unsigned long i=0;

  while(1)
  {
    if(kbhit()) break;  /* a função kbhit retorna 0 se uma tecla
                           não for pressionada*/
    i++;
    printf("%d\n",i);

  }


} /* end main */
```

Box 2 - Encerrando um laço com pressionar de tecla.

continue

O comando *continue* tem o funcionamento semelhante ao *break*. A diferença é que ao invés de forçar o encerramento do laço, *continue* força o laço a executar a próxima iteração, desviando de códigos intermediários.

O programa exemplo a seguir realiza a codificação de mensagens, onde o caractere original é substituído por um caractere duas posições acima. Exemplo, se digitado A, terá como codificação C.

```
main()
{
  char ok, chr;

  ok = 0;

  while(!ok)
  {
    chr = getchar();     /* armazena os caracteres digitados */

    if(chr == '@')       /* se igual a @ */
    {
      ok = 1;            /* ok é verdadeiro */
      continue;          /* força ao próximo teste condicional */

    } /* end if */

    putchar(chr+2);      /* imprime na tela codificado */

  } /* end while */


} /* end main */
```

Box 3 - Programa que codifica caracteres.

O polêmico ***goto***

A instrução *goto* determina um desvio incondicional no código, em uma tradução livre significa “vá para”. A sua sintaxe no C, para gerar um loop infinito no código é conforme o Box 4. A sintaxe do *goto* para um desvio à frente é a mesma, porém nesse caso a label estará situada depois da instrução.

```
label:
  instrução;
  goto label;
```

Box 4 - Sintaxe do goto para loop infinito.

Como pode-se observar, precisamos utilizar *goto* associado a uma label, que basicamente é uma palavra que você escolhe e segue as mesmas regras de declaração de variáveis: não pode iniciar com número, não pode ter espaços, não pode ser uma palavra reservada do C e não pode conter caracteres especiais e/ou acentuação. Programadores acostumados com a linguagem BASIC, provavelmente têm a tendência de utilizar o *goto* em seus códigos, pois nesta linguagem a instrução é corriqueira.

No entanto em C, recomenda-se o uso do *goto* com cautela, pelo fato do mesmo provocar desvios no programa, inclusive podendo facilmente provocar bugs inesperados em seu software ou torna-lo difícil de interpretar. O fato é que não existe nenhuma situação específica na programação em C, que não possa ser resolvida sem o uso de *goto*.

Só para ilustrar, o programa do Box 3 poderia ser escrito conforme o Box 5.

```
main()
{
  char ok, chr;

  ok = 0;

  while(!ok)
  {
    chr = getchar();     /* armazena os caracteres digitados */

    if(chr == '@')       /* se igual a @ */
      goto exit_while;

    putchar(chr+2);      /* imprime na tela codificado */

  } /* end while */

exit_while:


} /* end main */
```

Box 5 - Utilizando goto para um loop infinito.

Mas não há motivos para uso de *goto* sendo que temos o *break* e o *continue* que são considerados mais elegantes.

Em resumo, você deve evitar o uso de *goto*. No entanto, temos a obrigação de apresenta-lo pois consiste em uma das palavras reservadas do C.

Exercício resolvido: faça um projeto em C que solicita a entrada de um número pelo usuário. Quando o número digitado for ímpar, o software encerra com mensagem de erro. Enquanto forem digitados números pares, o software segue solicitando entradas do usuário.

Exercício proposto: projete um código em Linguagem C que gere a Série de Fibonacci de 0 até no máximo 1000 na tela. A Série inicia com a sequência 0, 1 e o próximo número sempre será o resultado da soma dos dois anteriores: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34...

Bibliografia: DAMAS, Luís; Linguagem C, décima edição.

Disponível em: https://amzn.to/3nGdIbN