Introdução à Programação #10

Interrupção do Fluxo

Os fluxos são algo (quase) indispensável em programas minimamente complexos que, por vezes, precisamos ter um maior controle sobre os mesmos. Já conhecemos o *loop while* que nos permite repetir um determinado trecho de código enquanto uma condição for verdadeira.

Mas e se necessitarmos de interromper a continuidade de um fluxo por algum motivo? Se por exemplo, efetuarmos um teste lógico *if* para saber se é necessário efetuar algo mais dentro de um *loop while*. Se sim, realiza, mas caso contrário, deverá passar para a próxima iteração (repetição).

break

O primeiro comando que nos permite ter um maior controle sobre os fluxos de repetição é o comando *break* permite que terminemos o *loop* atual sem quaisquer problemas.

Exemplo

Precisamos de encontrar o primeiro número divisível por 17, 18 e 20 entre 0 e 1000000. Para o fazermos, bastaria utilizar o seguinte código:

```
1.  #include <stdio.h>
2.
3.  int main()
4.  {
5.          int count,
6.              num;
7.
8.          count = 1;
9.          num = 0;
10.
11.         while(count <= 1000000) {
12.             if(num== 0) {
13.                 if((count%17==0) && (count%18==0) && (count%20==0)) {
14.                     num=count;
15.                 }
16.             }
17.             count++;
18.         }
19.
20.         printf("O número divisível por 17, 18 e 20 entre 1 e 1000000 é: %d", n
    um);
21.         return 0;
22. }
```

Este trecho de código, depois de encontrar o número, que é 3060, continuaria a efetuar testes lógicos até o contador chegar a 1000000. Mas, se já encontrou o número, qual a necessidade de gastar tempo e poder de processamento? Bastaria adicionar um *break* para colmatar esta situação da seguinte forma:

```
1.  #include <stdio.h>
2.
3.  int main()
4.  {
5.          int count,
6.              num;
7.
8.          count = 1;
```

```
9.          num = 0;
10.
11.         while(count <= 1000000) {
12.             if(num== 0) {
13.                 if((count%17==0) && (count%18==0) && (count%20==0)) {
14.                     num=count;
15.                     break;
16.                 }
17.             }
18.             count++;
19.         }
20.
21.         printf("O número divisível por 17, 18 e 20 entre 1 e 1000000 é: %d", n
    um);
22.         return 0;
23. }
```

continue

O comando *continue* é o segundo comando deste género que interrompe a iteração atual passando logo para a seguinte iteração.

Exemplo

Precisamos de somar todos os números entre 0 e 1000 que não são múltiplos de 2 nem de 3. Para isso utilizamos o seguinte código:

```
1.  #include <stdio.h>
2.
3.  int main()
4.  {
5.      int count,
6.          sum;
7.
8.      count = 1;
9.      sum = 0;
10.
11.     while(count <= 1000) {
12.
13.         if(count%2 == 0 || count%3 == 0) {
14.             count++;
15.             continue;
16.         }
17.
18.         sum += count; //sum = sum + count
19.         count++;
20.     }
21.
22.     printf("Soma %d", sum);
23.     return 0;
24. }
```

O que este código faz é percorrer os números de 1 a 1000 e, se for divisível por 2 ou por 3, passa para a próxima iteração (adicionando um valor ao número atual). Caso a condição não se verifique, o número é acrescentado à soma, adicionado um número a *count* passando assim para a próxima iteração.

Teste lógico *switch*

O teste condicional *switch* é extremamente útil quando precisamos de, por exemplo, efetuar um comando dependendo da opção, entre 0 e 10, que o utilizador escolheu.

Claro que poderíamos utilizar os *if* porém o código ficaria deveras extenso e um pouco complicado de ler. Veja o seguinte código com um *switch*:

```
1.  #include <stdio.h>
2.
3.  int main()
4.  {
5.      int option;
6.
7.      printf("Insira a opção:\n");
8.      scanf("%d", &option);
9.
10.     switch(option) {
11.         case 1:
12.             printf("Escolheu a opção 1");
13.             break;
14.         case 2:
15.             printf("Escolheu a opção 2");
16.             break;
17.         case 3:
18.             printf("Escolheu a opção 3");
19.             break;
20.         case 4:
21.             printf("Escolheu a opção 4");
22.             break;
23.         case 5:
24.             printf("Escolheu a opção 5");
25.             break;
26.         default:
27.             printf("Opção inexistente.");
28.             break;
29.     }
30.
31.     return 0;
32. }
```

Como pode visualizar, o código executado para cada opção pode ser diferente. Caso o inserido não corresponda a nenhuma das opções, o que está em *default* é executado. Veja o código anterior com *if else*:

```
1.  #include <stdio.h>
2.
3.  int main()
4.  {
5.      int option;
6.
7.      printf("Insira a opção:\n");
8.      scanf("%d", &option);
9.
10.     if (option == 1) {
11.         printf("Escolheu a opção 1");
12.     } else if (option == 2) {
13.         printf("Escolheu a opção 2");
14.     } else if (option == 3) {
15.         printf("Escolheu a opção 3");
16.     } else if (option == 4) {
```

```
17.         printf("Escolheu a opção 4");
18.     } else if (option == 5) {
19.         printf("Escolheu a opção 5");
20.     } else {
21.         printf("Opção inexistente.");
22.     }
23.
24.     return 0;
25. }
```

Comparando as duas, é possível verificar que utilizando o *switch* o código fica mais legível e fácil de ser editado.

Deve ter reparado ainda que no final de cada instrução do *switch* utilizámos um *break*. Isto acontece porque, caso não o utilizemos, o programa continuaria a correr os comandos seguintes que existem no *switch*.

http://pplware.sapo.pt/tutoriais/vamos-programar-introducao-a-programacao-11