NCURSES e CONIO

NCURSES e CONIO são duas bibliotecas de funções em C voltadas para o desenvolvimento de aplicações que utilizam o console (janela de texto e teclado) como interface de entrada e saída.

No entanto, nenhuma das duas bibliotecas existe como padrão da linguagem C (ANSI C). A biblioteca NCURSES está disponível em ambientes que geram código executável para Linux, ao passo que conio é comumente disponibilizada por ambientes e pacotes de programação C para Windows e/ou DOS.

O objetivo deste documento é apresentar funcionalidades comuns de NCURSES e CONIO, e exemplificando e explicando quais são as funções que implementam essas funcionalidades para cada biblioteca.

1. Inclusão no projeto e inicialização

A inclusão/inicialização de CONIO é feita da seguinte maneira:

- codificando-se `#include <conio.h>` no início do código-fonte. Notar que alguns ambientes de desenvolvimento (por exemplo, Dev-C++) não oferecem CONIO na forma de biblioteca e sim como um arquivo fonte separado; nesse caso deve-se utilizar `#include <conio.c>`

2. Posicionamento e saída de texto

Utiliza-se a função *cprintf* da biblioteca padrão para saída de texto. Esta função é diferente de *printf* porque utiliza as configurações do console para imprimir (por exemplo, posicionamento e cores).

A posição do cursor pode ser configurada através de uma chamada à função *gotoxy(x, y)*, onde *x* é o número da coluna (começando em **1** para a coluna à esquerda e crescendo para a direita) e *y* é o número da linha (começando em **1** para a linha no alto do console e crescendo para baixo). O exemplo a seguir mostra o mesmo posicionamento utilizado no exemplo de NCURSES, porém levando em consideração que o sistema de coordenadas e a ordem dos argumentos de *gotoxy* são diferentes dos utilizados por *move.*

Exemplo:

```
#include <conio.h>
int main()
{
   gotoxy(31, 11);   /* mais ou menos no meio do console */
   cprintf("Texto no meio");     /* mostra a partir da posição (31,11) */
   return 0;
}
```

3. Configuração de entrada via teclado

Podem ser configurados a priori:

- exibição ou não do cursor, através da chamada a *_setcursortype(tipo)*, onde *tipo* é *_NOCURSOR* (para esconder o cursor), *_NORMALCURSOR* (cursor normal) e *_SOLIDCURSOR* (cursor sólido).

Exemplo:

```
#include <conio.h>
int main()
{
   _setcursortype(_NOCURSOR);    /* a partir de agora não mostra o cursor nas
                                    funções de impressão de texto */
   . . .
   return 0;
}
```

4. Entrada via teclado

A entrada de caracter pode ser feita utilizando-se *getch()* (sem eco) ou *getche()* (com eco). Estas funções retornam o código ASCII do caracter correspondente à tecla pressionada.

Para teclas especiais, como as teclas de seta, *getch()* retorna o número 0 e, se for chamada novamente, retorna um código que pode ser testado para identificar a tecla. Os códigos das teclas de seta são os seguintes:

- Para cima: 72
- Para baixo: 80
- Para esquerda: 75
- Para direita: 77

Considerando o fato que ambas *getch()* e *getche()* aguardam indefinidamente pela tecla, é necessário verificar a priori se existe uma tecla disponível para ser lida. Isso é feita através da chamada à função *kbhit()* , que retorna diferente de zero se uma tecla está disponível e 0 caso contrário. Como não existe o conceito de tempo máximo de espera nesse caso, o exemplo mostra um código-fonte no qual a função *clock()* de `time.h` está sendo utilizada para contar o tempo.

Exemplo:

```
#include <conio.h>
#include <stdio.h>
#include <time.h>
int main()
{
   int t0, t1;
   char carac;
   printf(“Por quantos segundos vc quer esperar por tecla?”);
   scanf(“%d”, &tempo);
   t0 = clock();
   do
   {
         t1 = clock();
   }
   while (t1 < t0 + CLOCKS_PER_SEC*tempo);     /* atraso */

   if (kbhit() != 0)
   {
         carac = getch();  /* só lê a tecla se ela está disponível */
         printf(“A tecla digitada foi %c”, carac);
   }
   else
   {
         printf(“Demorou demais”);
   }
   return 0;
}
```

5. Utilização de cores no texto

Fornece as funções *textcolor(cor)* e *textbackground(cor)* para ajustar as cores do texto e do fundo do texto, respectivamente. O argumento *cor* é uma das constantes pré-definidas, normalmente o nome da cor em inglês em letras maiúsculas (p. ex., RED, CYAN, WHITE, BLACK).

Lembrar que o texto só será exibido na cor configurada se for utilizado *cprintf*.

Exemplo:

```
#include <conio.h>
#include <stdio.h>
int main()
{
   textcolor(RED);              /* cor do texto para vermelho */
   textbackground(BLACK);       /* cor de fundo para preto */
   cprintf("Esse texto possui cor vermelha sobre fundo preto\n");
   textcolor(YELLOW);           /* cor do texto para amarelo */
   textbackground(BLUE);        /* cor do fundo para azul */
   cprintf("Esse texto possui cor amarela sobre fundo azul\n");
   return 0;
}
```