LINGUAGEM C - COMANDOS ESPECIAIS E DE TELA

CORES

Podemos alterar as cores de fundo e de texto de nossas janelas de programas em C, existem mais de um comando para esta função.

Um comando que pode ser usado para trocar a cor de fundo e de texto de um programa em C é o:

system("color corfundocortexto);

Esse comando altera para todo um programa em C, a cor de fundo da janela e do texto mostrado, essa alteração vale para toda a execução do comando.

Para usá-lo devemos ter incluído a biblioteca stdlib.h.

Exemplo:

```
#include <stdlib.h>
#include <stdio.h>

int main()
{
	system("color 01"); //coloca a cor de fundo como preto e de texto em azul

	printf("Meu texto na cor azul");

	system("pause");

	return(0);
}
```

Cores disponíveis

0 = Preto	8 = Cinza
1 = Azul	9 = Azul claro
2 = Verde	10 = Verde claro
3 = Verde-água	11 = Verde-água claro
4 = Vermelho	12 = Vermelho claro
5 = Roxo	13 = Lilás
6 = Amarelo	14 = Amarelo claro
7 = Branco	15 = Branco brilhante

As cores devem ser definidas pelos seu números em Hexadecimal (ex. Verde claro = A)

Outros comandos disponíveis para trabalhar com cores estão em uma biblioteca chamada conio.c, nela temos os comandos textbackground(corfundo); e textcolor(cortexto)

A vantagem destes comandos em relação ao system(“color”) é que com eles podemos mudar a cor do texto e de fundo várias vezes durante o mesmo programa.

Aceitam as mesmas variações de cores, e elas podem ser definidas por números de 0 a 15 ou pelo nome da cor.

Exemplo:

```
#include <stdlib.h>
#include <stdio.h>
#include <conio.c>

int main()
{
        textbackground(3); //define a cor de fundo como verde água

        textcolor(1);

        printf(“Meu texto na cor azul ”);

        textcolor(4);

        printf(“Meu texto na cor vermelho ”);

        system(“pause”);

        return(0);
}
```

Podemos usar ao invés de um número fixo para a cor, uma variável do tipo inteira.

Exemplo:

```
#include <stdlib.h>
#include <stdio.h>
#include <conio.c>

int cor=0;

int main()
{
        textbackground(3); //define a cor de fundo como verde água
        textcolor(cor);
        printf(“Meu texto”);
        cor++;
        textcolor(cor);
        printf(“Meu texto 2 ”);
        system(“pause”);
        return(0);
}
```

LIMPAR TELA

Outro comando importante em se tratando de tela, é o comando para limpar a tela, existem 2 comandos que pode ser usados para tal. No momento que o compilador encontra um destes comandos ele limpa a tela e coloca o que vem após ele em uma tela nova.

system (“cls”); // biblioteca <stdlib.h>

clrscr();// biblioteca <conio.c>

POSICIONAMENTO TEXTO

A biblioteca conio.c possui um comando para definir a posição de um texto na janela. A janela em que os comandos são executados tem um tamanho máximo de 80 colunas por 24 linhas.

O comando usado é:
gotoxy(x,y); //posiciona em determinada coluna (x) e determinada linha (y)

Exemplo:

```
#include <stdlib.h>
#include <stdio.h>
#include <conio.c>

int main()
{
	gotoxy(20,5); //posiciona o texto na coluna 20 e linha 5
	printf(“Meu texto”);

	gotoxy(20,7); //posiciona o texto na coluna 20 e linha 7
	printf(“Meu texto 2 ”);

	system(“pause”);
	return(0);
}
```

Temos que usar um comando gotoxy para cada texto a ser posicionado, o texto escrito no printf não poderá ter o \n para que o posicionamento funcione.

Podemos usar além de valores fixos para o gotoxy variáveis do tipo inteira;

Exemplo:

```
#include <stdlib.h>
#include <stdio.h>
#include <conio.c>

int x=20,y=8;

int main()
{
	gotoxy(x,y);
	printf(“Meu texto”);
```

```
        x++;
        y++;

        gotoxy(x,y);
        printf(“Meu texto 2 ”);

        system(“pause”);
        return(0);
}
```

RETARDO NA EXECUÇÃO DO PROGRAMA

Um comando bastante útil em alguns programas é o comando Sleep (com S maiúsculo), ele causa um retardo na execução do programa em alguns milisegundos.

Podemos usá-lo sempre que precisarmos parar por algum tempo a execução.

Exemplo:

```
#include <stdlib.h>
#include <stdio.h>
#include <conio.c>

int i;

int main()
{
        textbackground(1);
        clrscr();
        textcolor(15);
        printf("\nAguarde carregando o programa ");
        for (i=1;i<50;i++)
        {
                Sleep(100); //retarda em 100 milisegundos
                printf(".");
        }

        textbackground(4);
        clrscr();
        textcolor(15);
        printf("\nPrograma carregado com sucesso!!!\n\n");

        system("pause");
        return(0);
}
```

DATA/HORA

Existem comandos específicos para pegar a data e a hora do sistema.

Exemplo

```
printf("DATA :%s HORA:%s",__DATE__,__TIME__);
```

```
//__DATE e __TIME sao constantes do sistema ...

ou

system ( "date /t" );
system ("time /t");
```

RANDOM

Em alguns problemas precisamos gerar números aleatórios, ou seja, que não sejam definidos por usuário e que mudem cada vez que o programa for executado, para isso podemos usar o comando rand().

O comando funciona da seguinte forma:

// Primeiro definimos o random:

```
srand(time(NULL));
```

// Depois atribuímos ele a uma variável (ele randomiza de acordo com o horário do pc). E podemos colocar um mod, de modo que ele defina um limite.

```
variavel1 = rand() % 10;   // até 10.
variavel2 = rand() % 30;  // até 30
```

Exemplo:

```
#include <stdlib.h>
#include <stdio.h>
#include<time.h>

int num, variavel1;

int main()
{
    printf("\nDigite um número de 1 a 10 ");
    scanf("%d",&num);

    srand(time(NULL));
    variavel1 = rand() % 10;   // até 10.

    if (num==variavel1)
       printf("\n Número %d. Parabéns você acertou!!",variavel1);
    else
       printf("\n Ops você errou!! Número %d. ",variavel1);

    system("pause");
    return(0);
}
```